Introducing Watch Connectivity
Communication between devices
Session 713

Chris Jensen  watchOS Engineer
Alex Ledwith  watchOS Engineer
watchOS 1

iOS App
WatchKit Extension
Data Store

WatchKit App
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
WatchConnectivity
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
Always set your apps up to receive incoming WatchConnectivity content
Setup

Always set your apps up to receive incoming WatchConnectivity content

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

Always set your apps up to receive incoming WatchConnectivity content.

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

Always set your apps up to receive incoming WatchConnectivity content

```swift
if (WCSession.isSupported()) {
    let session = WCSession.defaultSession()
    session.delegate = self // conforms to WCSessionDelegate
}
```
Always set your apps up to receive incoming WatchConnectivity content

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

News App
let session = WCSession.defaultSession()
session.delegate = self
session.activateSession()
Session State

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

News App
Session State

News App

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

Open the Apple Watch app on your iPhone and tap Start Pairing.

Start Pairing
Session State

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

Open the Apple Watch app on your iPhone and tap Start Pairing.

Start Pairing
Session State

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

Open the Apple Watch app on your iPhone and tap Start Pairing.

Start Pairing
Session State

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

Open the Apple Watch app on your iPhone and tap Start Pairing.

Start Pairing
Session State

```javascript
session.paired == false
```
Session State

`session.paired == false`
Session State

session.paired == false
Session State

News App

\[ \text{session.paired} == \text{false} \]
Session State

News App

session.paired == false

func sessionWatchStateDidChange(_)

 nhuun:func

func sessionWatchStateDidChange(_):func

session.paired == false
Session State

```
session.paired == true
func sessionWatchStateDidChange(_)
```
Session State

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

News App
Session State

```javascript
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

News App
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
```

News App
Session State

News App

```
session.complicationEnabled == false
```
Session State

**News App**

`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

News App
A B C D

WatchKit App
C D
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

Social Networking App
Background Transfers

Application context

Social Networking App

applicationContext

gerceivedApplicationContext
Background Transfers
Application context

updateApplicationContext(_:)
applicationContext

Social Networking App
receivedApplicationContext
Background Transfers

Application context

updateApplicationContext(_:)

applicationContext

receivedApplicationContext

Social Networking App
Background Transfers

Application context

`updateApplicationContext(_:)`

`applicationContext`
Background Transfers

Application context

updateApplicationContext(_:)

Social Networking App

applicationContext

receivedApplicationContext
Background Transfers
Application context

updateApplicationContext(_:)

Social Networking App

applicationContext

receivedApplicationContext
Background Transfers
Application context

updateApplicationContext(_:)

applicationContext

receivedApplicationContext

Social Networking App
Background Transfers

Application context

updateApplicationContext(_:)  
applicationContext  

Social Networking App  

receivedApplicationContext  
WatchKit Extension
Background Transfers
Application context: Code
do {
    let context = // Create context dictionary with latest state
do {
    let context = // Create context dictionary with latest state
    try WCSession.defaultSession().updateApplicationContext(context)
}
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

Game
Background Transfers
User info transfer

Outstanding User Info Transfers

Game
Background Transfers
User info transfer

Outstanding User Info Transfers

L1
L2

Game
Background Transfers
User info transfer

Outstanding User Info Transfers

L1
L2

transferUserInfo(_:)

Game
Background Transfers

User info transfer

transferUserInfo(_:_)

Outstanding User Info Transfers

L1

L2

L3
Background Transfers

User info transfer

Outstanding User Info Transfers

transferUserInfo(_:)

Game

L1  L2  L3
Background Transfers
User info transfer

```swift
transferUserInfo(_:)
```

Outstanding User Info Transfers

---

L1  L2  L3
Background Transfers

User info transfer

iOS Game

transferUserInfo(_:)

Game

Outstanding User Info Transfers
Background Transfers

User info transfer: Code
let userInfo = // Create dictionary of userInfo
let userInfo = // Create dictionary of userInfo
let userInfoTransfer = WCSession.defaultSession().transferUserInfo(userInfo)
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

Image Editing App
Background Transfers

File transfer

Outstanding File Transfers

~/Documents/Inbox/

Image Editing App
Background Transfers

File transfer

Outstanding File Transfers

~/Documents/Inbox/

Image Editing App

P1

P2
Background Transfers

File transfer

transferFile(_: metadata:)

~/Documents/Inbox/

Outstanding File Transfers

~/Documents/Inbox/

P2  P1
Background Transfers

File transfer

`transferFile(_: metadata:)`

Outstanding File Transfers

~/Documents/Inbox/
Background Transfers

File transfer

transferFile(_, metadata:)

Outstanding File Transfers

P3  P2  P1

~/Documents/Inbox/
Background Transfers

File transfer

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

Outstanding File Transfers

`~/Documents/Inbox/`

Image Editing App

P1  P2  P3
Background Transfers

File transfer

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

Outstanding File Transfers

~/Documents/Inbox/

Image Editing App

WatchKit Extension
Background Transfers

File transfer: Code
Background Transfers

File transfer: Code

let url = // Retrieve URL of file
let url = // Retrieve URL of file
let metadata = // Create dictionary of metadata
let url = // Retrieve URL of file
let metadata = // Create dictionary of metadata
let fileTransfer = WCSession.defaultSession().transferFile(url, metadata:metadata)
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
// Receiver Callback
func session(session: WCSession, didReceiveFile file: WCSessionFile) {
    // Handle file URL and metadata in WCSessionFile object
}
// Receiver Callback
func session(session: WCSession, didReceiveFile file: WCSessionFile) {
    // Handle file URL and metadata in WCSessionFile object
}

Background Transfers
File transfer: Delegate callback

WCSessionFile
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
}

// 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
}

// 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

```java
WCSession.defaultSession().reachable
```
Interactive Messaging
Reachability: iPhone

iOS App
Interactive Messaging
Reachability: iPhone

Devices connected
Interactive Messaging
Reachability: iPhone

Devices connected
Watch app foreground
Interactive Messaging
Reachability: iPhone

Devices connected
Watch app foreground

sessionreachable == 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

Run Tracker App

WatchKit Extension
Interactive Messaging
Background launching: iOS app only
Interactive Messaging
Background launching: iOS app only

Run Tracker App
WatchKit Extension
iOS App Background
Interactive Messaging

Background launching: iOS app only

- Run Tracker App
- WatchKit Extension
- iOS App Background
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

```swift
func sendMessage(message:, replyHandler:, errorHandler:)
```
Interactive Messaging

Types

Dictionary
- Property list types

Data

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

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

Types

Dictionary
- Property list types

Data
- Custom data

---

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

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

Types

Dictionary
• Property list types

Data
• Custom data
• Own serialization

---

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

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

Replying
Interactive Messaging

Replying

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

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

Replying

Optional handler

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

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

Replying

Optional handler

Recommended

```swift
func sendMessage(message: String, replyHandler: () -> Void, errorHandler: () -> Void)
func sendMessageData(data: Data, replyHandler: () -> Void, errorHandler: () -> Void)
```
Interactive Messaging

Replying

Optional handler
Recommended
• Confirmation by receiver

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

Interactive Messaging
Delegate callbacks
Interactive Messaging
Delegate callbacks

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

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

Code

```java
if (WCSession.defaultSession().reachable) {
```
Interactive Messaging

Code

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

Code

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

When to use
NSURLConnection

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
NSURLConnection and WatchConnectivity
NSURLConnection and WatchConnectivity

News App

Application Context

WatchKit Extension

Watch
NSURLConnection and WatchConnectivity

News App

Application Context

WatchKit Extension
NSURLSession and WatchConnectivity

HTTP Request

Application Context

News App

WatchKit Extension

WatchKit Extension
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

Weather App
Complications
Updating clock face

WatchKit Extension

ClockKit
Complications

Updating clock face
Complications

Updating clock face

WatchKit Extension Background

ClockKit
Complications
Updating clock face

let server = CLKComplicationServer.sharedInstance()
Complications
Updating clock face

let server = CLKComplicationServer.sharedInstance()
server.extendTimelineForComplication(aComplication)
Complications
Updating clock face

WatchKit
Extension
Background

ClockKit
Complications

Updating clock face

gGetCurrentTimelineEntryForComplication(_:withHandler:)
Complications
Updating clock face

WatchKit Extension
Background

ClockKit

getCurrentTimelineEntryForComplication(_:withHandler:)
getTimelineEntriesForComplication(_:beforeDate:limit:withHandler:)
Complications
Updating clock face

getCurrentTimelineEntryForComplication(_:withHandler:)
getTimelineEntriesForComplication(_:beforeDate:limit:withHandler:)
getTimelineEntriesForComplication(_:afterDate:limit:withHandler:)
Complications
Updating clock face

WatchKit Extension
Background

ClockKit

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
Complications

Updating clock face

ClockKit

Content for past, present, and future

Background launched for updating

Specify when content provided will be stale
Initial Activation

Complications
Complications

Initial activation
Complications
Initial activation
Complications

Initial activation
Complications

Initial activation
Complications

Initial activation
Complications
Initial activation

NSURLSession -> WatchKit Extension Background
Complications
Initial activation
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

sendMessage(_:replyHandler:errorHandler:)

session.reachable == true
Complications

Initial activation

sendMessage(_:replyHandler:errorHandler:)

session.reachable == true

Weather App Background

WatchKit Extension Background
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

Cloud

iPhone
Apple Watch
Complications

Pushed

Cloud

iPhone

Apple Watch

Sports App
Complications

Requested interval fetch

Cloud

iPhone

Apple Watch
Complications

Requested interval fetch

Cloud

iPhone

Apple Watch

Surfing App
Complications
Requested interval fetch

NSURLSession
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

WatchKit Extension
Background

Timeline
Updates
Extend
Timeline

Requested interval fetch

Complications
Requested interval fetch

WatchKit Extension
Background

Timeline
Updates
Extend
Timeline

Requested interval fetch

Complications
Requested interval fetch

WatchKit Extension
Background

Timeline
Updates
Extend
Timeline

Requested interval fetch

Complications
Requested interval fetch

WatchKit Extension
Background

Timeline
Updates
Extend
Timeline

Requested interval fetch

Complications
Requested interval fetch

WatchKit Extension
Background

Timeline
Updates
Extend
Timeline

Requested interval fetch

Complications
Requested interval fetch

WatchKit Extension
Background

Timeline
Updates
Extend
Timeline

Requested interval fetch

Complications
Requested interval fetch

WatchKit Extension
Background

Timeline
Updates
Extend
Timeline

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
NSURLConnection requests might not complete until next time the extension runs
Complications

Requested interval fetch: Summary

Use a NSURLSession background session if possible
NSURLConnection 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 URLSession 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

Cloud

iPhone

Apple Watch

Sports App
Complications
Pushed
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

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

Pushed: Part 1

Updated PushKit framework

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

Pushed: Part 1

Updated PushKit framework

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

Pushed: Part 1

Updated PushKit framework

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

```swift
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

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

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

```swift
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

WatchConnectivity
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

Sports App

→

Sports App

→

Watch
Complications

Pushed: Summary

Sports App

Registration

Sports App

Registration
Complications

Pushed: Summary

- Push Token
- Sports App
- Registration
Complications
Pushed: Summary

Sports App

Registrations
Push Token
Complications
Pushed: Summary
Complications

Pushed: Summary

- Sports App
- Push Token
- Push Token
- Push Token

Diagram showing a cloud connected to a smartphone and then to a smartwatch, indicating data flow.
Complications
Pushed: Summary
Complications
Pushed: Summary
Complications
Pushed: Summary

[Diagram: Cloud connected to a phone, then to a watch, with a 'Push' label in the cloud]
Complications
Pushed: Summary
Complications
Pushed: Summary

Sports App
Background

[Diagram showing a connection between a cloud, a smartphone, and a smartwatch]
Complications
Pushed: Summary

transferUserInfo(_:)

Outstanding User Info Transfers

U3

U2 U1

Sports App
Background
Complications

Pushed: Summary

Outstanding User Info Transfers

Sports App
Background

U3  U2  U1
Complications
Pushed: Summary

transferCurrent
ComplicationUserInfo(_:)

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

<table>
<thead>
<tr>
<th>U3</th>
<th>U2</th>
<th>U1</th>
</tr>
</thead>
</table>

| U4 |
Complications

Pushed: Summary

Outstanding User Info Transfers

WatchKit Extension Background
Complications

Pushed: Summary

Outstanding User Info Transfers

WatchKit Extension Background
Complications
Pushed: Summary
Complications

Pushed: Summary

WatchKit Extension Background

ClockKit

Update
Complications
Pushed: Summary
Complications

Pushed: Summary
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
More Information

Documentation
watchOS 2 Transition Guide
WatchKit Programming Guide

Sample Code
Lister
WatchKit Catalog

Technical Support
Apple Developer Forums
Developer Technical Support

General Inquiries
Jake Behrens,
watchOS Frameworks Evangelist
behrens@apple.com

http://developer.apple.com/watchOS
<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introducing WatchKit for watchOS 2</td>
<td>Presidio</td>
<td>Tuesday 10:00AM</td>
</tr>
<tr>
<td>Privacy and Your App</td>
<td>Pacific Heights</td>
<td>Tuesday 2:30PM</td>
</tr>
<tr>
<td>Building Watch Apps</td>
<td>Pacific Heights</td>
<td>Tuesday 4:30PM</td>
</tr>
<tr>
<td>Security and Your Apps</td>
<td>Mission</td>
<td>Tuesday 4:30PM</td>
</tr>
<tr>
<td>Creating Complications with ClockKit</td>
<td>Pacific Heights</td>
<td>Wednesday 11:00AM</td>
</tr>
<tr>
<td>Networking with NSURLSession</td>
<td>Pacific Heights</td>
<td>Thursday 9:00AM</td>
</tr>
<tr>
<td>WatchKit Tips and Tricks</td>
<td>Presidio</td>
<td>Friday 10:00AM</td>
</tr>
<tr>
<td>Labs</td>
<td>Frameworks Lab</td>
<td>Time</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Watch Connectivity Lab</td>
<td>B</td>
<td>Thursday 1:30PM</td>
</tr>
<tr>
<td>WatchKit and ClockKit Complications Lab</td>
<td>A</td>
<td>Friday 1:30PM</td>
</tr>
</tbody>
</table>