#WWDC19

Advances in App Background Execution

Roberto Alvarez, Software Battery Life
Thomas Zhao, Software Battery Life


box
AR
R
Disney
CNN
Hilton
BBC NEWS
HBO GO
FedEx
Hello FRESH

Different Use Cases

Calls	Finish Foreground Work	Music	Maintenance
Downloading and Uploading Files		Siri Integration	MFi Accessory
Bluetooth	Periodic Content Updates		Connecting to Hotspots
Step Counting	Watch App	New Server Data	Navigation

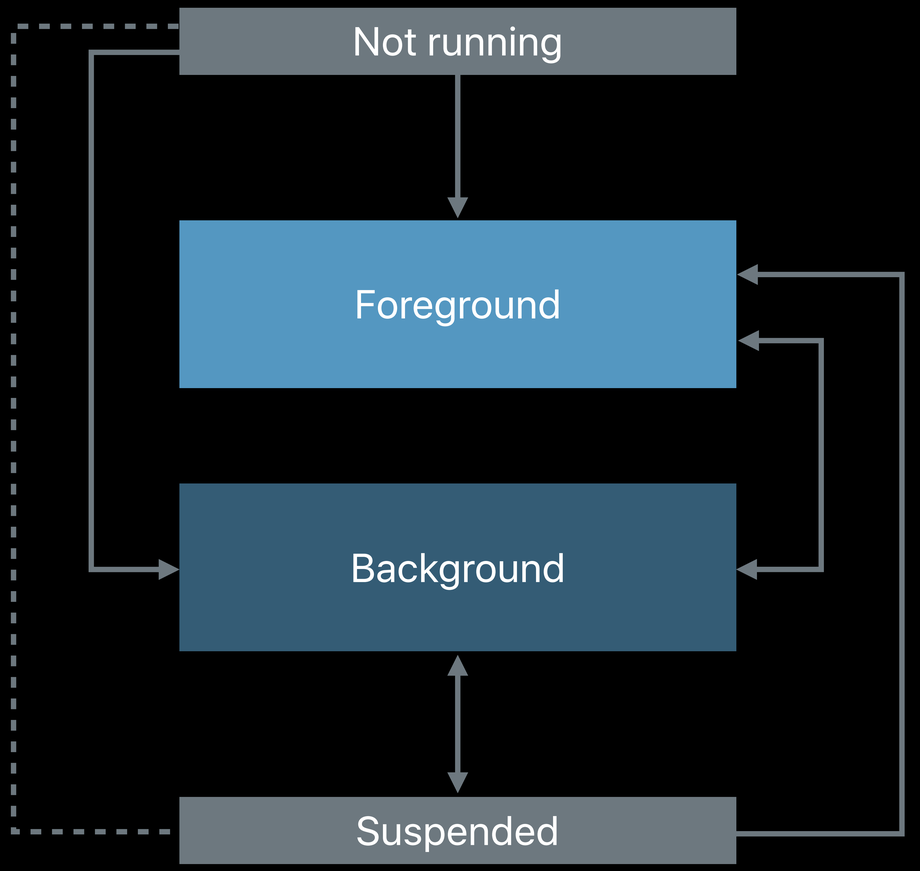
Overview of Background Execution

Best practices

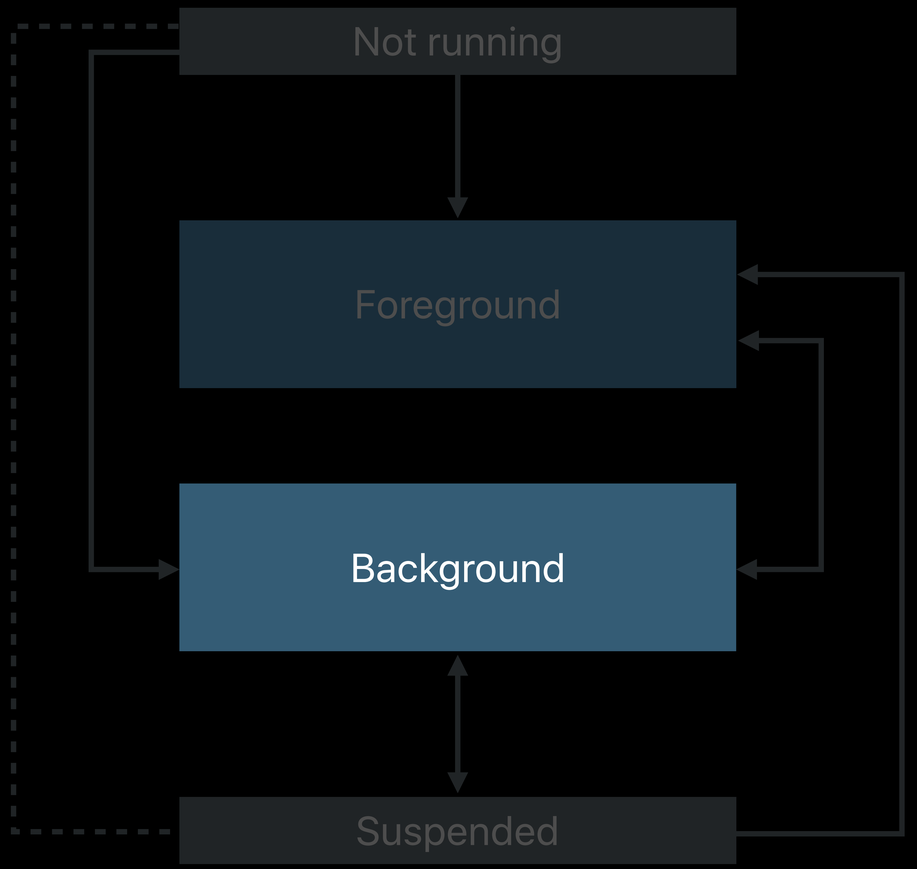
New BackgroundTasks framework

Overview of Background Execution

What is Background Execution?



What is Background Execution?



Why Do We Enter This State?

Why Do We Enter This State?

App request

Why Do We Enter This State?

App request

Event trigger

Important Considerations for Background Execution

Important Considerations for Background Execution

Power

Important Considerations for Background Execution

Power

Performance

Important Considerations for Background Execution

Power

Performance

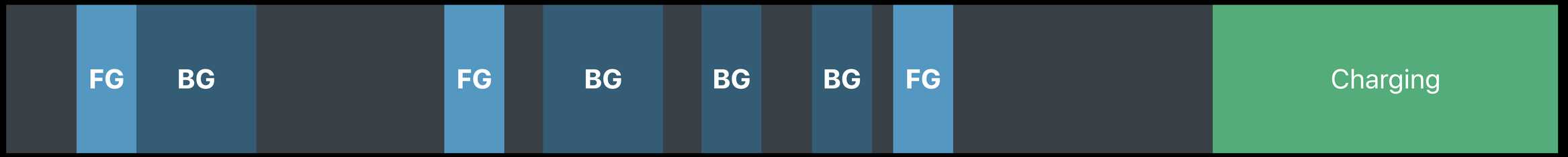
Privacy

Power

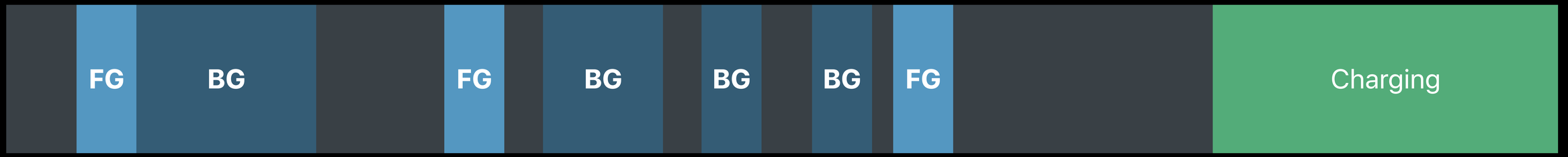
Power

Power

Power



Power

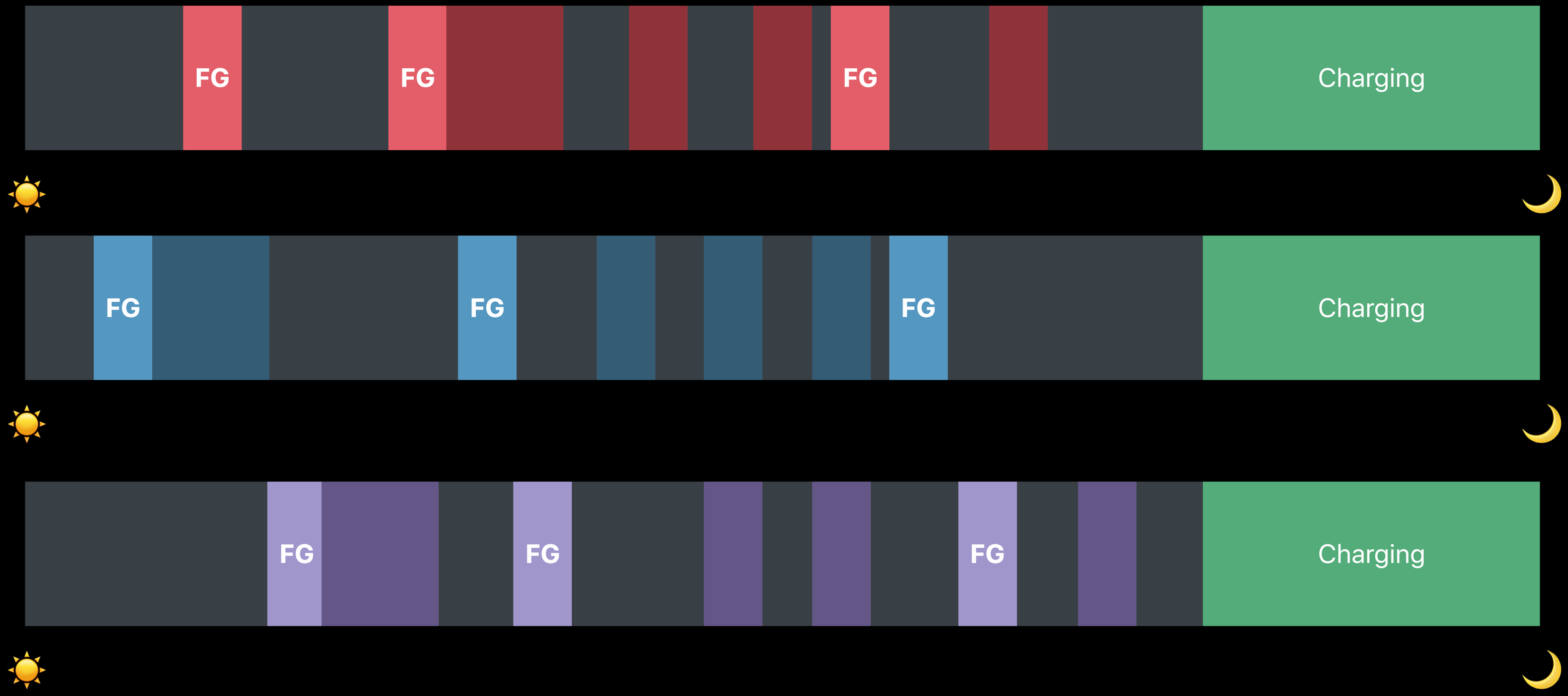


Power

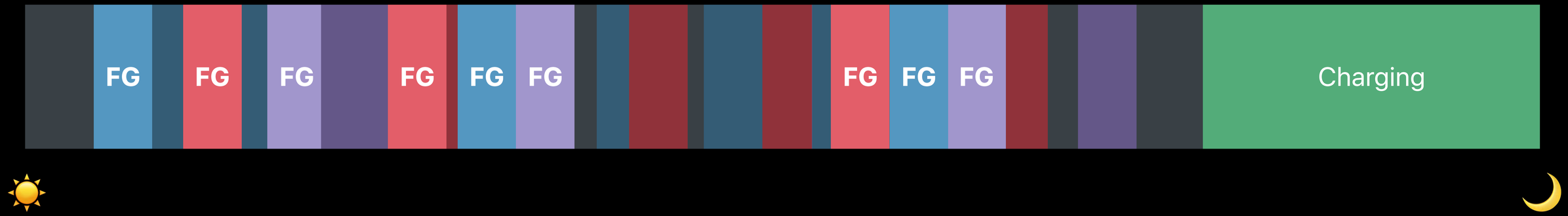
Performance

Performance

Performance



Performance

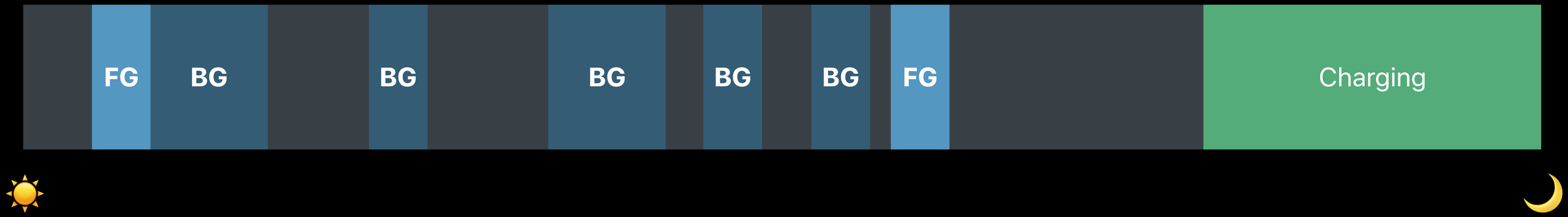


Privacy

Privacy

Privacy

Privacy



Important Considerations for Background Execution

Power

Performance

Privacy

Different Use Cases

Calls	Finish Foreground Work	Music	Maintenance
Downloading and Uploading Files		Siri Integration	MFi Accessory
Bluetooth	Periodic Content Updates		Connecting to Hotspots
Step Counting	Watch App	New Server Data	Navigation

Different Modes

VoIP

Background Task Completion

Audio

BGProcessingTask

Background URLSessionTask

SiriKit Intent

Accessory

Bluetooth

Background App Refresh

Network Authentication

HealthKit

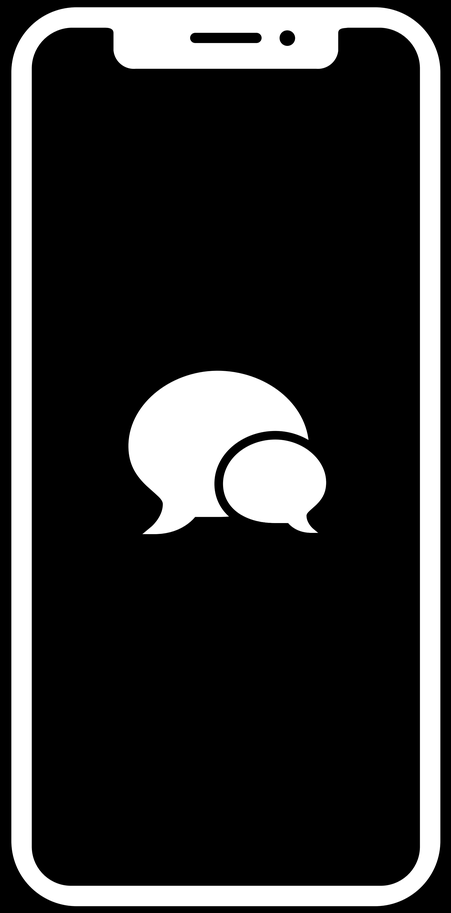
WatchConnectivity

Background Push

Location

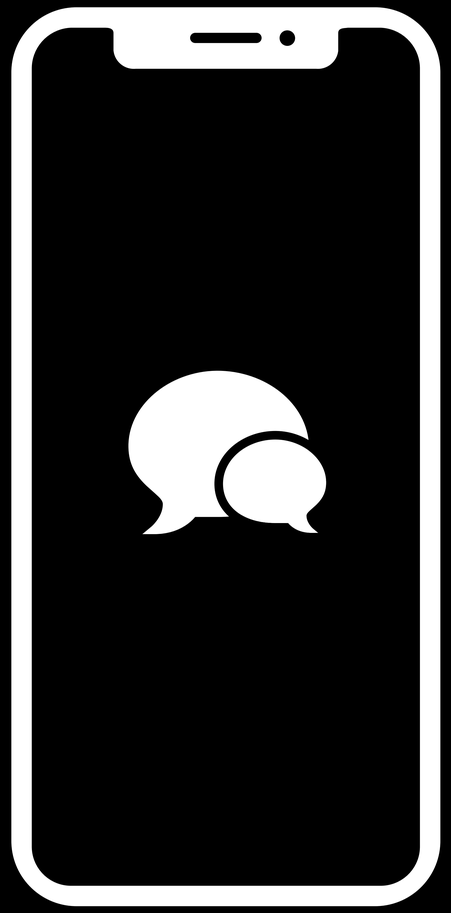
Background Execution Best Practices

Messaging App



Messaging App

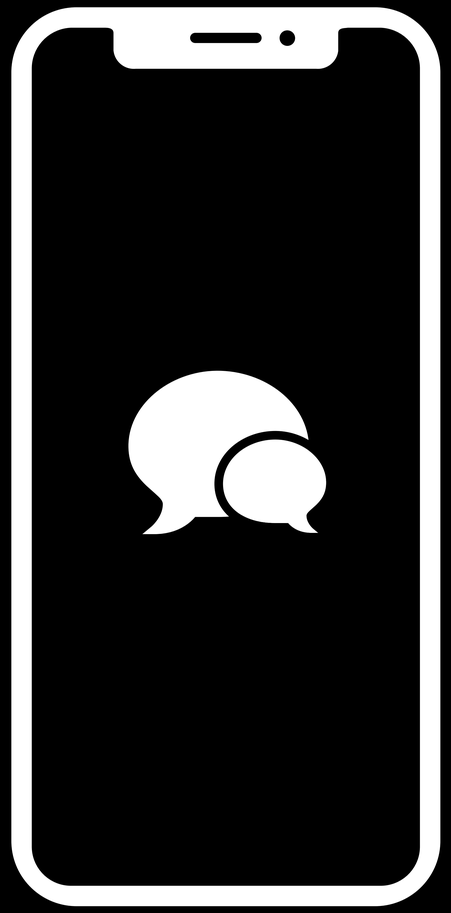
Send Messages



Messaging App

Send Messages

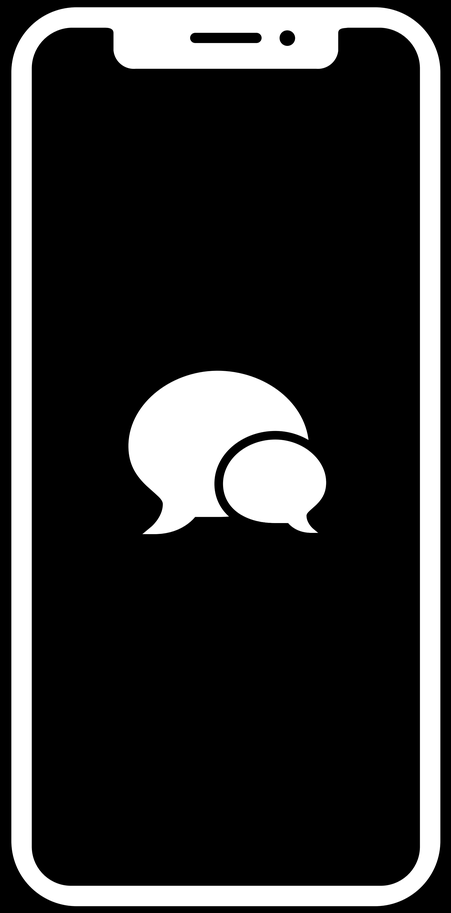
Calls

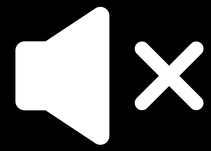


Messaging App

Send Messages

Calls



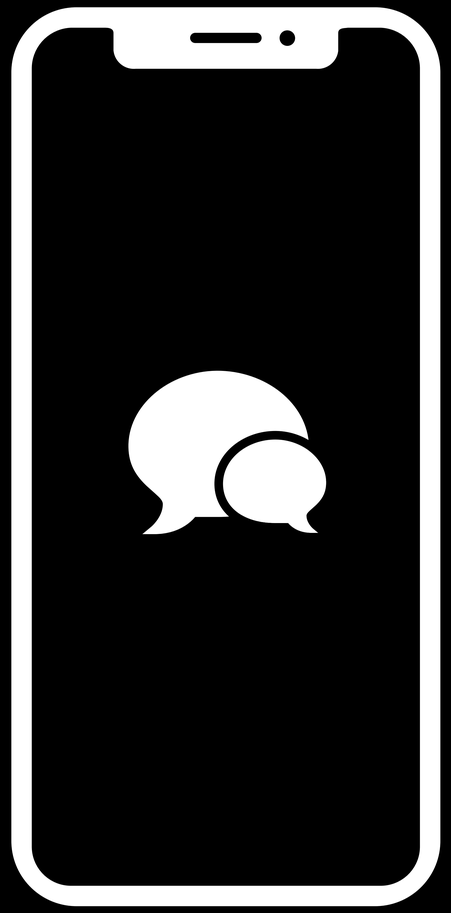


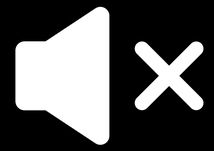
Muted Threads

Messaging App

Send Messages

Calls





Muted Threads

Download Past Attachments

Send Messages

Send Messages

User expects immediate completion

Send Messages

User expects immediate completion

Protect completion

Background Task Completion

Background Task Completion

Gives app additional time to run in the background before being suspended

- `UIApplication.beginBackgroundTask(expirationHandler:)`
- `ProcessInfo.performExpiringActivity(withReason:using:)`

Background Task Completion

Gives app additional time to run in the background before being suspended

- `UIApplication.beginBackgroundTask(expirationHandler:)`
- `ProcessInfo.performExpiringActivity(withReason:using:)`

Complete work started in the foreground

- Saving files to disk, completing user-initiated requests

```swift
// Guarding Important Tasks While App is Still in the Foreground

func send(_ message: Message) {
    let sendOperation = SendOperation(message: message)
    var identifier: UIBackgroundTaskIdentifier!
    identifier = UIApplication.shared.beginBackgroundTask(expirationHandler: {
        sendOperation.cancel()
        postUserNotification("Message not sent, please resend")
        // Background task will be ended in the operation's completion block below
    })
    sendOperation.completionBlock = {
        UIApplication.shared.endBackgroundTask(identifier)
    }
    operationQueue.addOperation(sendOperation)
}
```

```
// Guarding Important Tasks While App is Still in the Foreground

func send(_ message: Message) {
    let sendOperation = SendOperation(message: message)
    var identifier: UIBackgroundTaskIdentifier!
    identifier = UIApplication.shared.beginBackgroundTask(expirationHandler: {
        sendOperation.cancel()
        postUserNotification("Message not sent, please resend")
        // Background task will be ended in the operation's completion block below
    })
    sendOperation.completionBlock = {
        UIApplication.shared.endBackgroundTask(identifier)
    }
    operationQueue.addOperation(sendOperation)
}
```

```swift
// Guarding Important Tasks While App is Still in the Foreground

func send(_ message: Message) {
    let sendOperation = SendOperation(message: message)
    var identifier: UIBackgroundTaskIdentifier!
    identifier = UIApplication.shared.beginBackgroundTask(expirationHandler: {
        sendOperation.cancel()
        postUserNotification("Message not sent, please resend")
        // Background task will be ended in the operation's completion block below
    })
    sendOperation.completionBlock = {
        UIApplication.shared.endBackgroundTask(identifier)
    }
    operationQueue.addOperation(sendOperation)
}
```

```swift
// Guarding Important Tasks While App is Still in the Foreground

func send(_ message: Message) {
    let sendOperation = SendOperation(message: message)
    var identifier: UIBackgroundTaskIdentifier!
    identifier = UIApplication.shared.beginBackgroundTask(expirationHandler: {
        sendOperation.cancel()
        postUserNotification("Message not sent, please resend")
        // Background task will be ended in the operation's completion block below
    })
    sendOperation.completionBlock = {
        UIApplication.shared.endBackgroundTask(identifier)
    }
    operationQueue.addOperation(sendOperation)
}
```

```swift
// Guarding Important Tasks While App is Still in the Foreground

func send(_ message: Message) {
    let sendOperation = SendOperation(message: message)
    var identifier: UIBackgroundTaskIdentifier!
    identifier = UIApplication.shared.beginBackgroundTask(expirationHandler: {
        sendOperation.cancel()
        postUserNotification("Message not sent, please resend")
        // Background task will be ended in the operation's completion block below
    })
    sendOperation.completionBlock = {
        UIApplication.shared.endBackgroundTask(identifier)
    }
    operationQueue.addOperation(sendOperation)
}
```

Send Messages

Send Messages

Protect by using Background Task Completion

Send Messages

Protect by using Background Task Completion

Start based on user action

Phone Calls

Phone Calls

VoIP push notifications

Phone Calls

VoIP push notifications

Special type of push that launches your app

Phone Calls

VoIP push notifications

Special type of push that launches your app

```
func registerForVoIPPushes() {
    self.voipRegistry = PKPushRegistry(queue: nil)
    self.voipRegistry.delegate = self
    self.voipRegistry.desiredPushTypes = [.voIP]
}
```

Phone Calls

VoIP push notifications

Special type of push that launches your app

```
func registerForVoIPPushes() {
    self.voipRegistry = PKPushRegistry(queue: nil)
    self.voipRegistry.delegate = self
    self.voipRegistry.desiredPushTypes = [.voIP]
}
```

VoIP Pushes

NEW

VoIP Pushes

NEW

Must report incoming call with CallKit in `didReceiveIncomingPush` callback

VoIP Pushes

NEW

Must report incoming call with CallKit in `didReceiveIncomingPush` callback

If not, system may stop launching your app for VoIP pushes

```swift
let provider = CXProvider(configuration: providerConfiguration)

func pushRegistry(_ registry: PKPushRegistry, didReceiveIncomingPushWith payload:
PKPushPayload, for type: PKPushType, completion: @escaping () -> Void) {
    if type == .voIP {
        if let handle = payload.dictionaryPayload["handle"] as? String {
            let callUpdate = CXCallUpdate()
            callUpdate.remoteHandle = CXHandle(type: .phoneNumber, value: handle)
            let callUUID = UUID()
            provider.reportNewIncomingCall(with: callUUID, update: callUpdate) { _ in
                completion()
            }
            establishConnection(for: callUUID)
        }
    }
}
```

```swift
let provider = CXProvider(configuration: providerConfiguration)

func pushRegistry(_ registry: PKPushRegistry, didReceiveIncomingPushWith payload:
PKPushPayload, for type: PKPushType, completion: @escaping () -> Void) {
    if type == .voIP {
        if let handle = payload.dictionaryPayload["handle"] as? String {
            let callUpdate = CXCallUpdate()
            callUpdate.remoteHandle = CXHandle(type: .phoneNumber, value: handle)
            let callUUID = UUID()
            provider.reportNewIncomingCall(with: callUUID, update: callUpdate) { _ in
                completion()
            }
            establishConnection(for: callUUID)
        }
    }
}
```

```swift
let provider = CXProvider(configuration: providerConfiguration)

func pushRegistry(_ registry: PKPushRegistry, didReceiveIncomingPushWith payload:
PKPushPayload, for type: PKPushType, completion: @escaping () -> Void) {
    if type == .voIP {
        if let handle = payload.dictionaryPayload["handle"] as? String {
            let callUpdate = CXCallUpdate()
            callUpdate.remoteHandle = CXHandle(type: .phoneNumber, value: handle)
            let callUUID = UUID()
            provider.reportNewIncomingCall(with: callUUID, update: callUpdate) { _ in
                completion()
            }
            establishConnection(for: callUUID)
        }
    }
}
```

```swift
let provider = CXProvider(configuration: providerConfiguration)

func pushRegistry(_ registry: PKPushRegistry, didReceiveIncomingPushWith payload:
PKPushPayload, for type: PKPushType, completion: @escaping () -> Void) {
    if type == .voIP {
        if let handle = payload.dictionaryPayload["handle"] as? String {
            let callUpdate = CXCallUpdate()
            callUpdate.remoteHandle = CXHandle(type: .phoneNumber, value: handle)
            let callUUID = UUID()
            provider.reportNewIncomingCall(with: callUUID, update: callUpdate) { _ in
                completion()
            }
            establishConnection(for: callUUID)
        }
    }
}
```

VoIP Pushes

Other tips

VoIP Pushes

Other tips

Caller info in push payload

VoIP Pushes

Other tips

Caller info in push payload

Set `apns-expiration` on push to `0` or something small

VoIP Pushes

Other tips

Caller info in push payload

Set `apns-expiration` on push to `0` or something small

Prefer a banner? Use standard pushes

- Notification Service Extension to modify content

Muted Threads

Muted Threads

Many different threads

Muted Threads

Many different threads

User may not want alerts for specific thread

Muted Threads

Many different threads

User may not want alerts for specific thread

Message content may be relevant

Muted Threads

Many different threads

User may not want alerts for specific thread

Message content may be relevant

Alert device, not user

Background Pushes

Background Pushes

Mechanism to tell device new data is available without alerting user

Background Pushes

Mechanism to tell device new data is available without alerting user

Send push with `"content-available: 1"` without `"alert"` `"sound"` or `"badge"`

Background Pushes

Mechanism to tell device new data is available without alerting user

Send push with `"content-available: 1"` without `"alert"` `"sound"` or `"badge"`

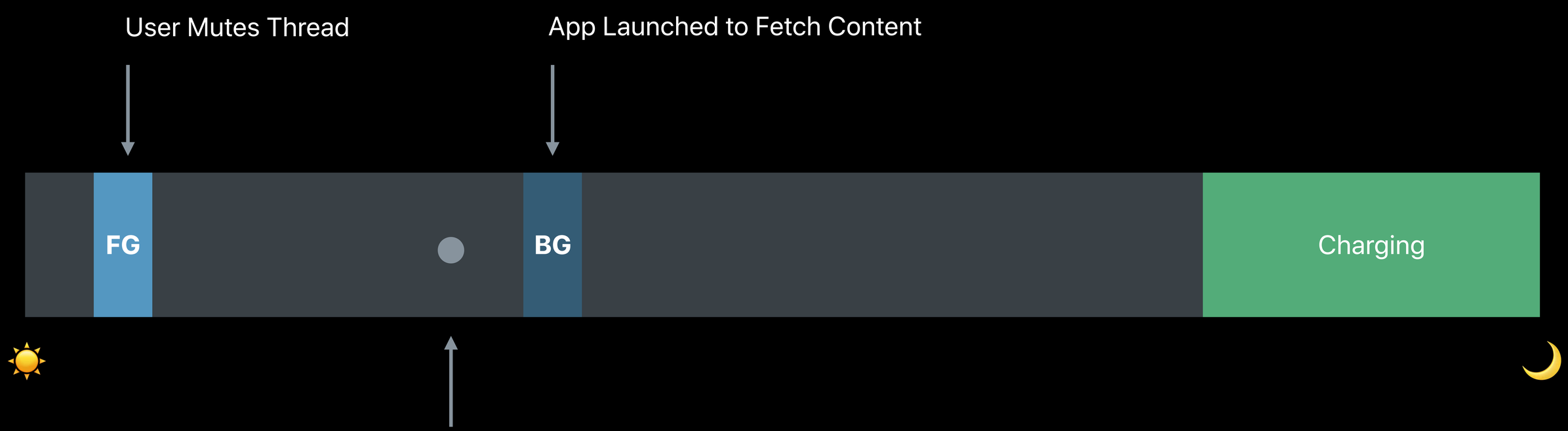
System intelligently decides when to launch the app to download content

Muted Thread

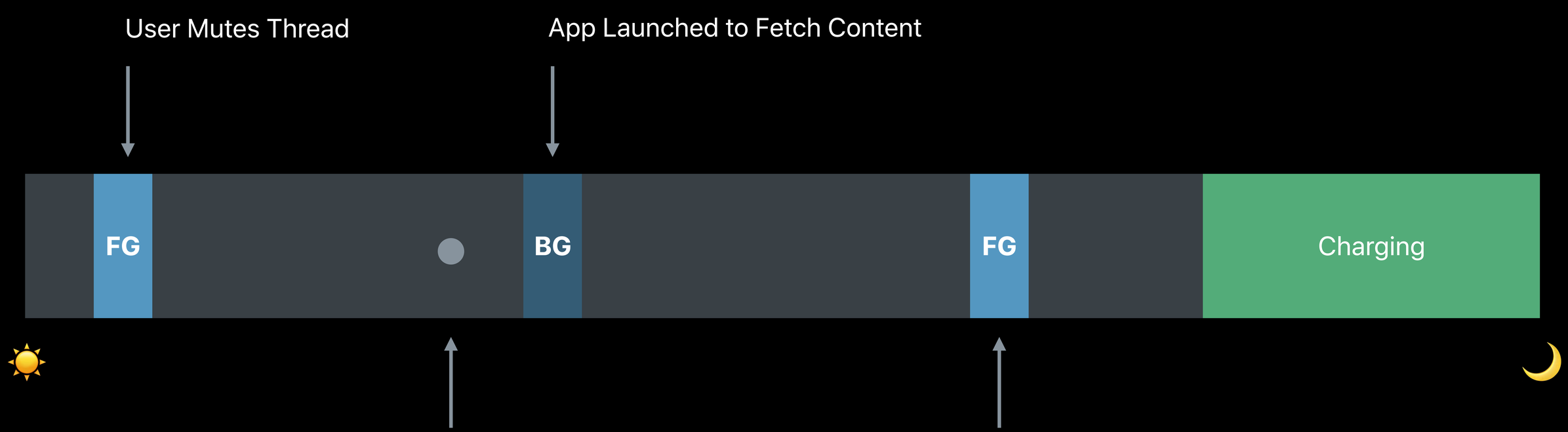
Muted Thread

Muted Thread

Muted Thread



Muted Thread



Background Pushes

NEW

Background Pushes

NEW

Must set `apns-priority = 5` or app will not launch

Background Pushes

NEW

Must set `apns-priority = 5` or app will not launch

Should set `apns-push-type = background`

- Required for watchOS
- Highly recommended for all platforms

Background Pushes

NEW

Must set `apns-priority = 5` or app will not launch

Should set `apns-push-type = background`

- Required for watchOS
- Highly recommended for all platforms

Creating Independent Watch Apps

Tuesday, 3:00

Muted Threads

Muted Threads

Use background push as best effort way to download content

Muted Threads

Use background push as best effort way to download content

Can always download when app re-enters foreground

Download Past Attachments

Download Past Attachments

User signs into account

Download Past Attachments

User signs into account

Download conversation list, recent messages immediately

Download Past Attachments

User signs into account

Download conversation list, recent messages immediately

Defer download of older content

Download Past Attachments

User signs into account

Download conversation list, recent messages immediately

Defer download of older content

Download Past Attachments

User signs into account

Download conversation list, recent messages immediately

Defer download of older content

Download Past Attachments

User signs into account

Download conversation list, recent messages immediately

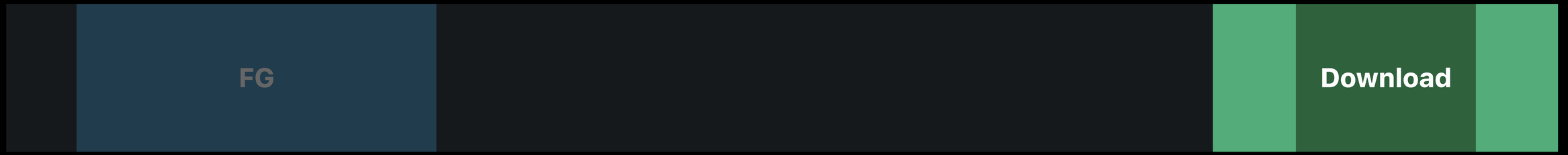
Defer download of older content

Download Past Attachments

User signs into account

Download conversation list, recent messages immediately

Defer download of older content



Discretionary Background URL Session

Discretionary Background URL Session

Allows the system to defer the download until a better time

Discretionary Background URL Session

Allows the system to defer the download until a better time

Provide information to system for smarter scheduling

Discretionary Background URL Session

Allows the system to defer the download until a better time

Provide information to system for smarter scheduling

```
// Set up background URL session
let config = URLSessionConfiguration.background(withIdentifier: "com.app.attachments")
let session = URLSession(configuration: config, delegate: ..., delegateQueue: ...)

// Set discretionary
config.discretionary = true
```

Discretionary Background URL Session

Allows the system to defer the download until a better time

Provide information to system for smarter scheduling

```
// Set up background URL session
let config = URLSessionConfiguration.background(withIdentifier: "com.app.attachments")
let session = URLSession(configuration: config, delegate: ..., delegateQueue: ...)

// Set discretionary
config.discretionary = true
```

```swift
// Set timeout intervals
config.timeoutIntervalForResource = 24 * 60 * 60
config.timeoutIntervalForRequest = 60

// Create request and task
var request = URLRequest(url: url)
request.addValue("...", forHTTPHeaderField: "...")
let task = session.downloadTask(with: request)

// Set time window
task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

// Set workload size
task.countOfBytesClientExpectsToSend = 160
task.countOfBytesClientExpectsToReceive = 4096
task.resume()
```

```
// Set timeout intervals
config.timeoutIntervalForResource = 24 * 60 * 60
config.timeoutIntervalForRequest = 60

// Create request and task
var request = URLRequest(url: url)
request.addValue("...", forHTTPHeaderField: "...")
let task = session.downloadTask(with: request)

// Set time window
task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

// Set workload size
task.countOfBytesClientExpectsToSend = 160
task.countOfBytesClientExpectsToReceive = 4096
task.resume()
```

```
// Set timeout intervals
config.timeoutIntervalForResource = 24 * 60 * 60
config.timeoutIntervalForRequest = 60

// Create request and task
var request = URLRequest(url: url)
request.addValue("...", forHTTPHeaderField: "...")
let task = session.downloadTask(with: request)

// Set time window
task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

// Set workload size
task.countOfBytesClientExpectsToSend = 160
task.countOfBytesClientExpectsToReceive = 4096
task.resume()
```

```swift
// Set timeout intervals
config.timeoutIntervalForResource = 24 * 60 * 60
config.timeoutIntervalForRequest = 60

// Create request and task
var request = URLRequest(url: url)
request.addValue("...", forHTTPHeaderField: "...")
let task = session.downloadTask(with: request)

// Set time window
task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

// Set workload size
task.countOfBytesClientExpectsToSend = 160
task.countOfBytesClientExpectsToReceive = 4096
task.resume()
```

```swift
// Set timeout intervals
config.timeoutIntervalForResource = 24 * 60 * 60
config.timeoutIntervalForRequest = 60

// Create request and task
var request = URLRequest(url: url)
request.addValue("...", forHTTPHeaderField: "...")
let task = session.downloadTask(with: request)

// Set time window
task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

// Set workload size
task.countOfBytesClientExpectsToSend = 160
task.countOfBytesClientExpectsToReceive = 4096
task.resume()
```

Download Past Attachments

Download Past Attachments

Defer work if possible to minimize user-visible impact

Download Past Attachments

Defer work if possible to minimize user-visible impact

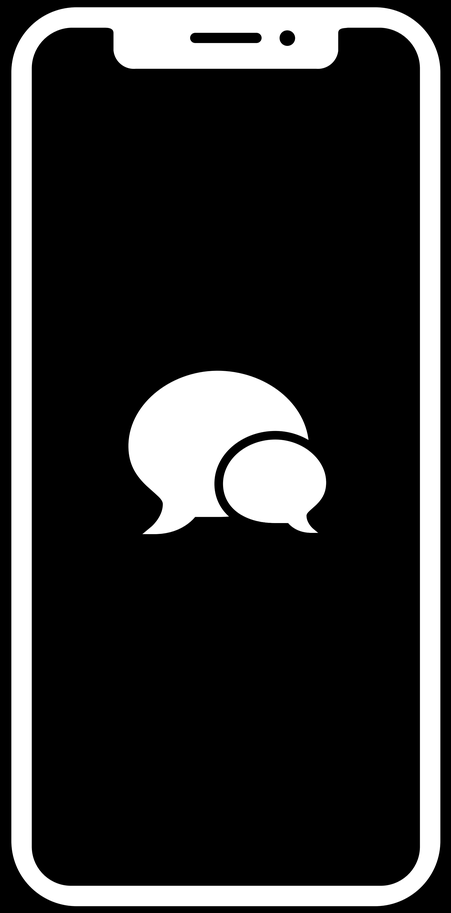
Same principles to any deferrable download or upload

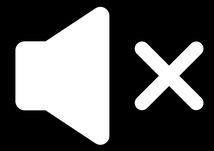
- Batching analytics uploads
- Photos

Recap

Send Messages

Calls





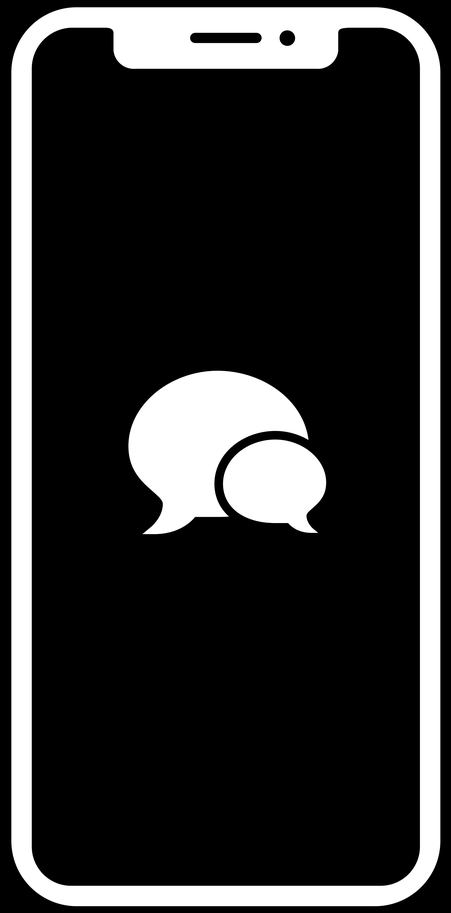
Muted Threads

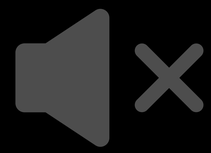
Download Past Attachments

Recap

Background Task Completion

VoIP Pushes

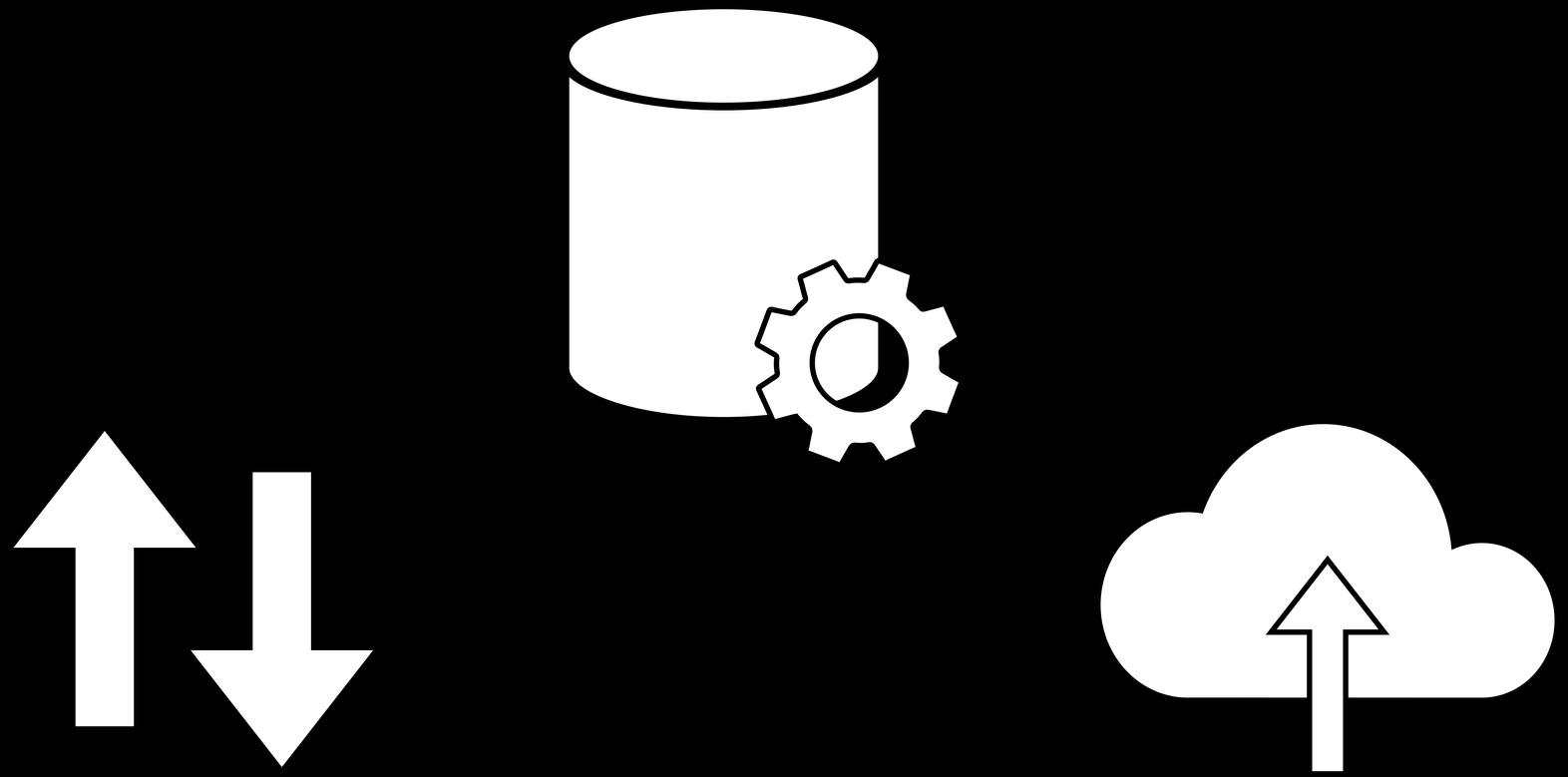




Background Pushes

Discretionary URL Session

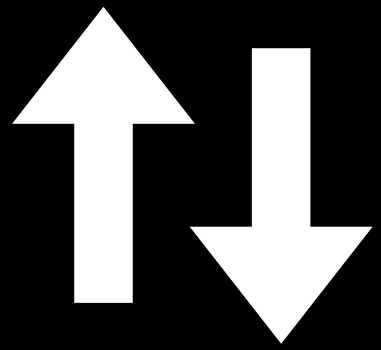
There’s More



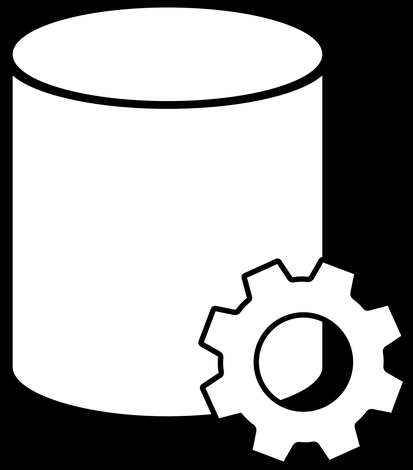
New BackgroundTasks Framework

Thomas Zhao, Software Battery Life

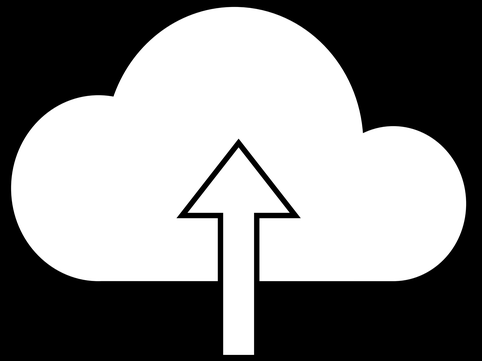
Deferrable Maintenance Work



Syncing



Database Cleanup



Backups

Deferrable Maintenance Work

Deferrable Maintenance Work

Deferrable Maintenance Work

BackgroundTasks

NEW

A new framework for scheduling background work

BackgroundTasks

NEW

A new framework for scheduling background work

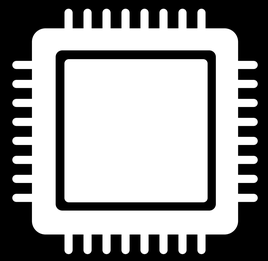
Available on iOS, iPadOS, tvOS, and iPad Apps on Mac

BackgroundTasks

NEW

A new framework for scheduling background work

Available on iOS, iPadOS, tvOS, and iPad Apps on Mac



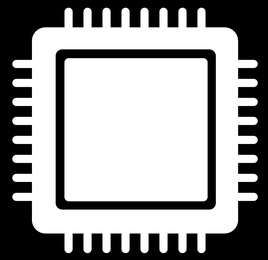
Background
Processing Tasks

BackgroundTasks

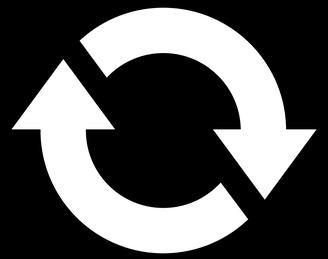
NEW

A new framework for scheduling background work

Available on iOS, iPadOS, tvOS, and iPad Apps on Mac



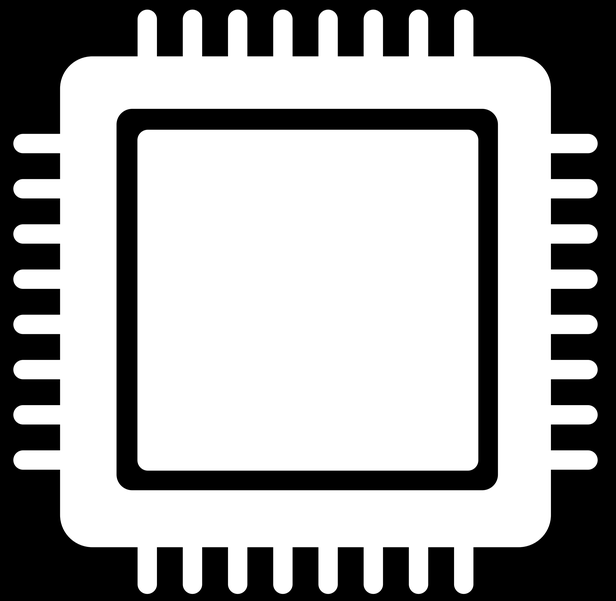
Background
Processing Tasks



Background App
Refresh Tasks

Background Processing Tasks

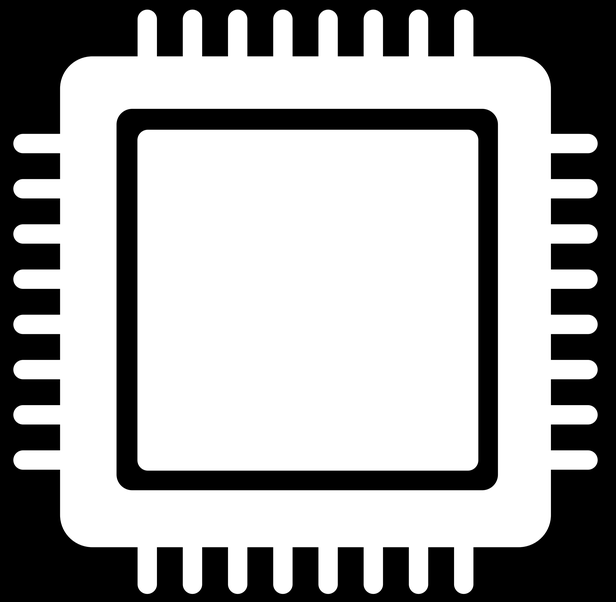
NEW



Background Processing Tasks

NEW

Several minutes of runtime at
system-friendly times

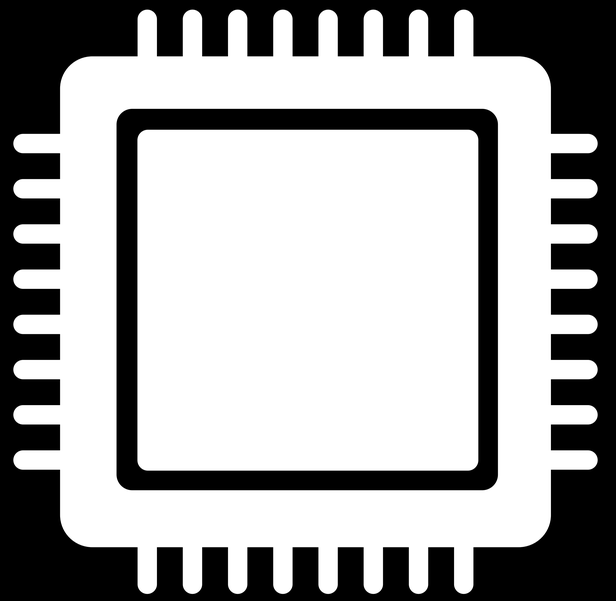


Background Processing Tasks

NEW

Several minutes of runtime at system-friendly times

- Deferrable maintenance work

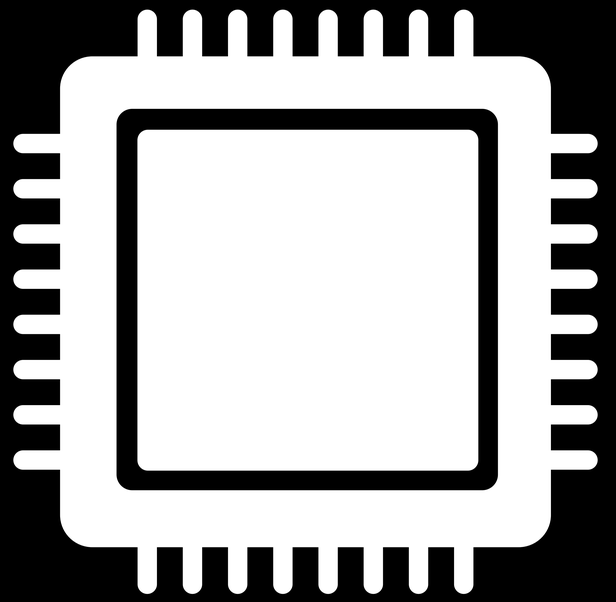


Background Processing Tasks

NEW

Several minutes of runtime at system-friendly times

- Deferrable maintenance work
- Core ML training/inference



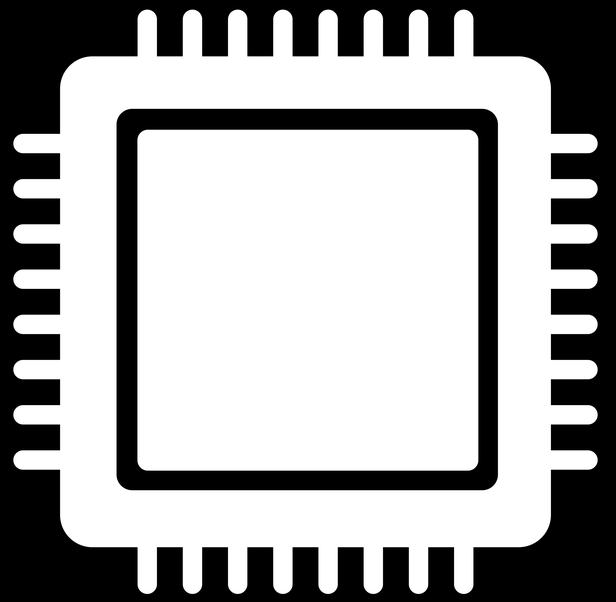
Background Processing Tasks

NEW

Several minutes of runtime at system-friendly times

- Deferrable maintenance work
- Core ML training/inference

Can turn off CPU Monitor for intensive work



Background Processing Tasks

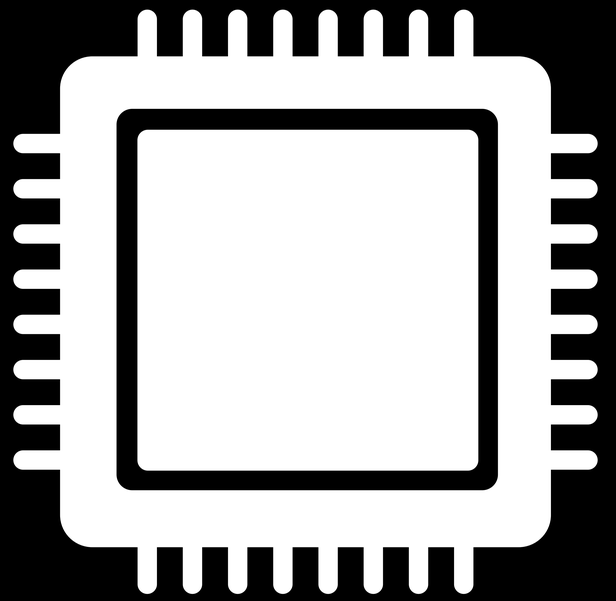
NEW

Several minutes of runtime at system-friendly times

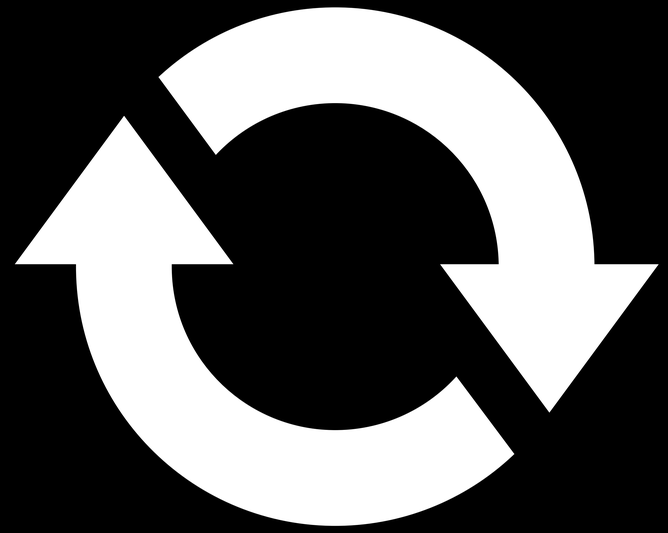
- Deferrable maintenance work
- Core ML training/inference

Can turn off CPU Monitor for intensive work

Eligible if requested in foreground or if app has been recently used



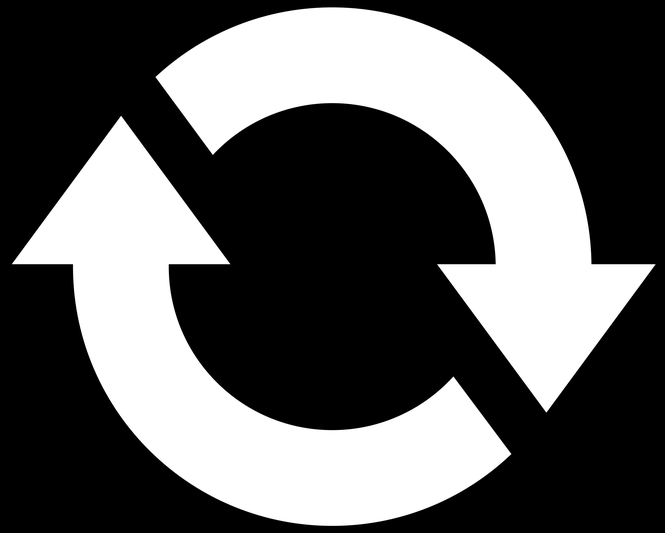
Background App Refresh Task



Background App Refresh Task

New API, same policies

- 30 seconds of runtime
- Keep app up-to-date throughout the day

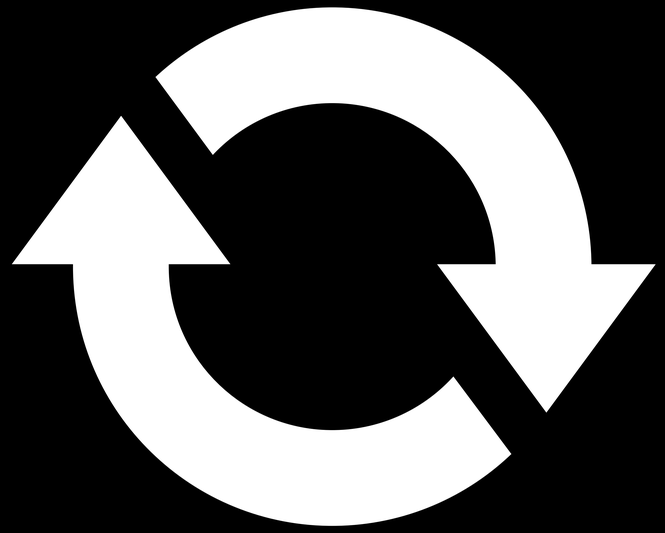


Background App Refresh Task

New API, same policies

- 30 seconds of runtime
- Keep app up-to-date throughout the day

Eligibility based on usage patterns



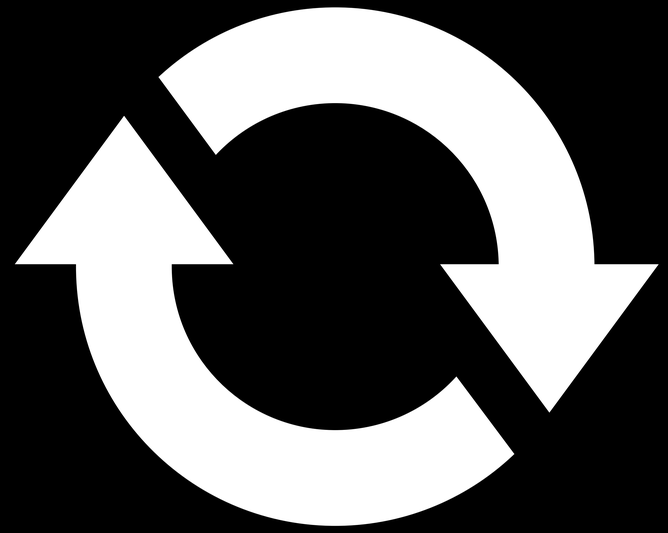
Background App Refresh Task

Background App Refresh Task

Background App Refresh Task

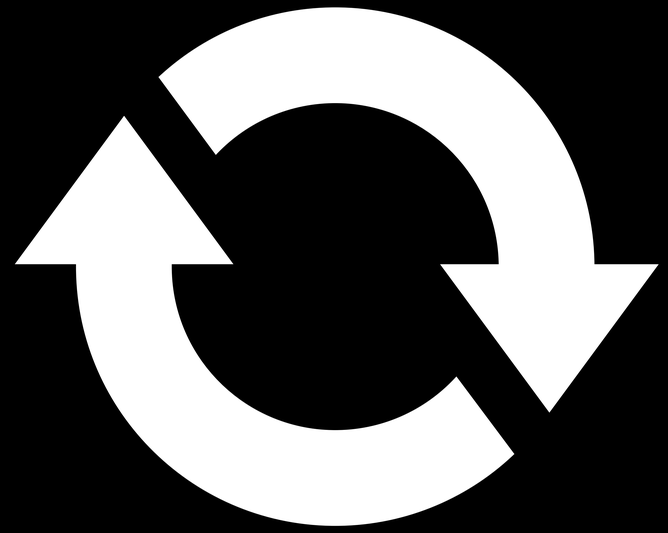
Background App Refresh Task

Old UIApplication fetch API is deprecated,
and not supported on Mac



Background App Refresh Task

Old UIApplication fetch API is deprecated,
and not supported on Mac



```
UIApplication.setMinimumBackgroundFetchInterval(_:)
UIApplicationDelegate.application(_:performFetchWithCompletionHandler:)
```

Using BackgroundTasks

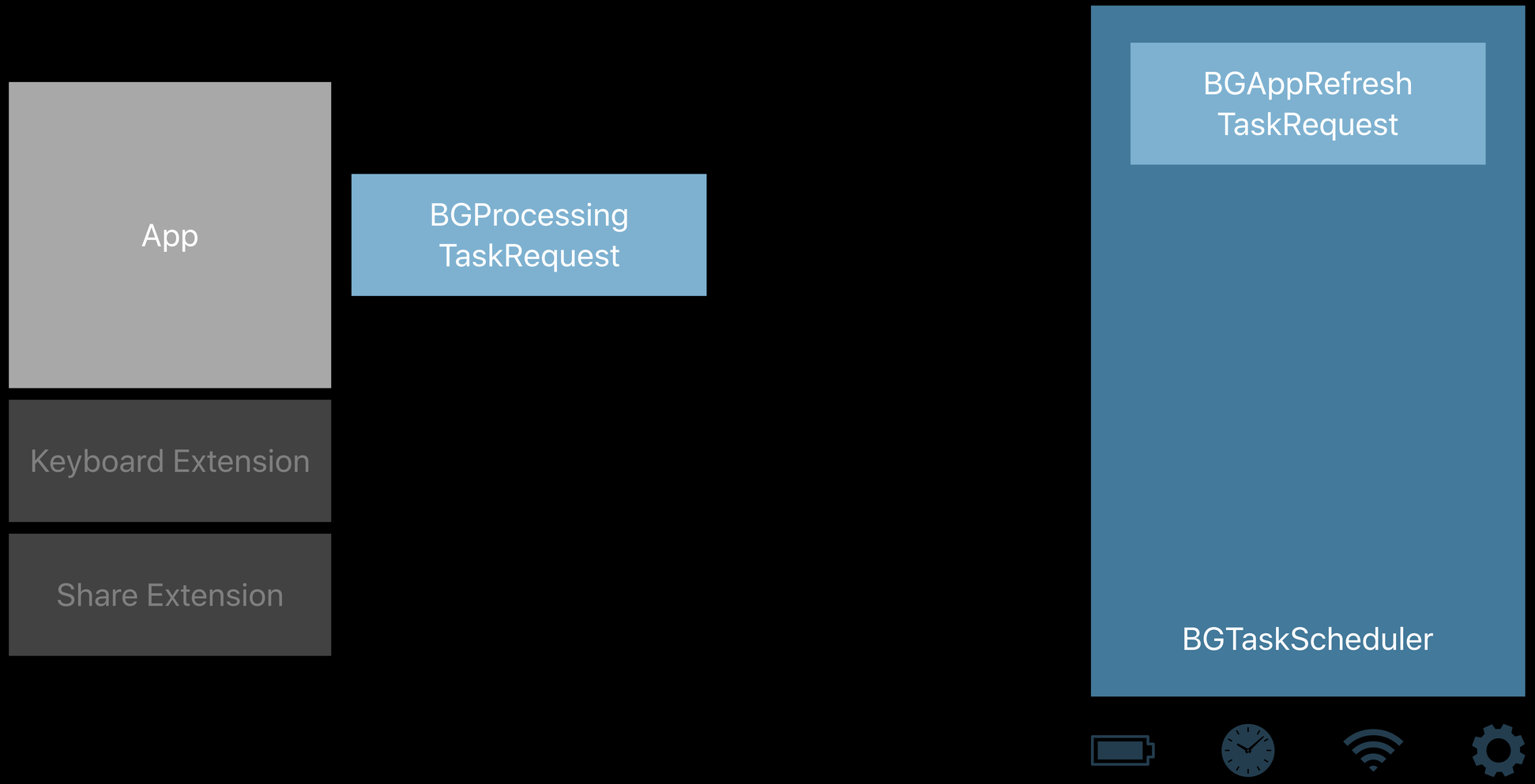
Using BackgroundTasks

Using BackgroundTasks

Using BackgroundTasks

Using BackgroundTasks

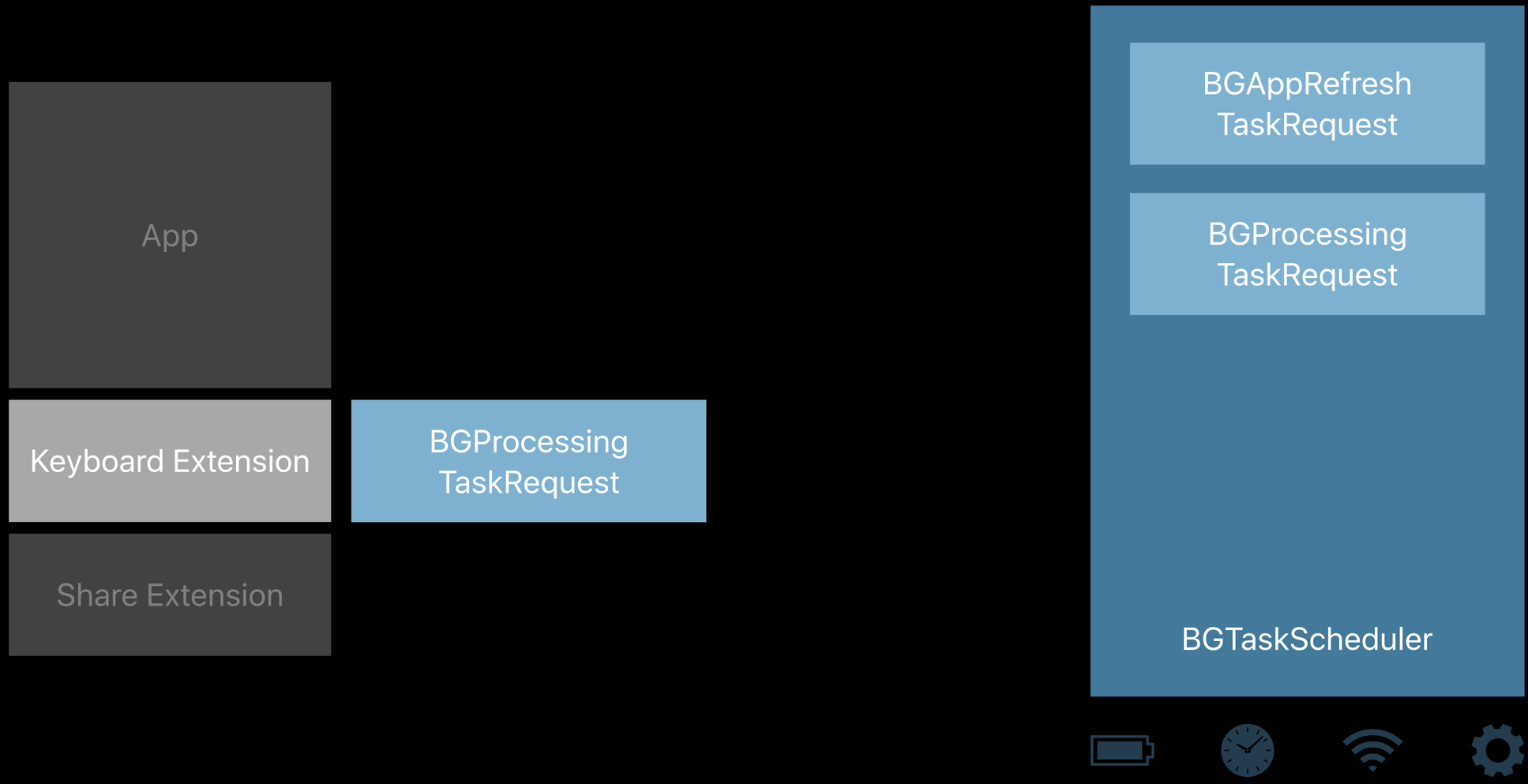
Using BackgroundTasks



Using BackgroundTasks

Using BackgroundTasks

Using BackgroundTasks



Using BackgroundTasks

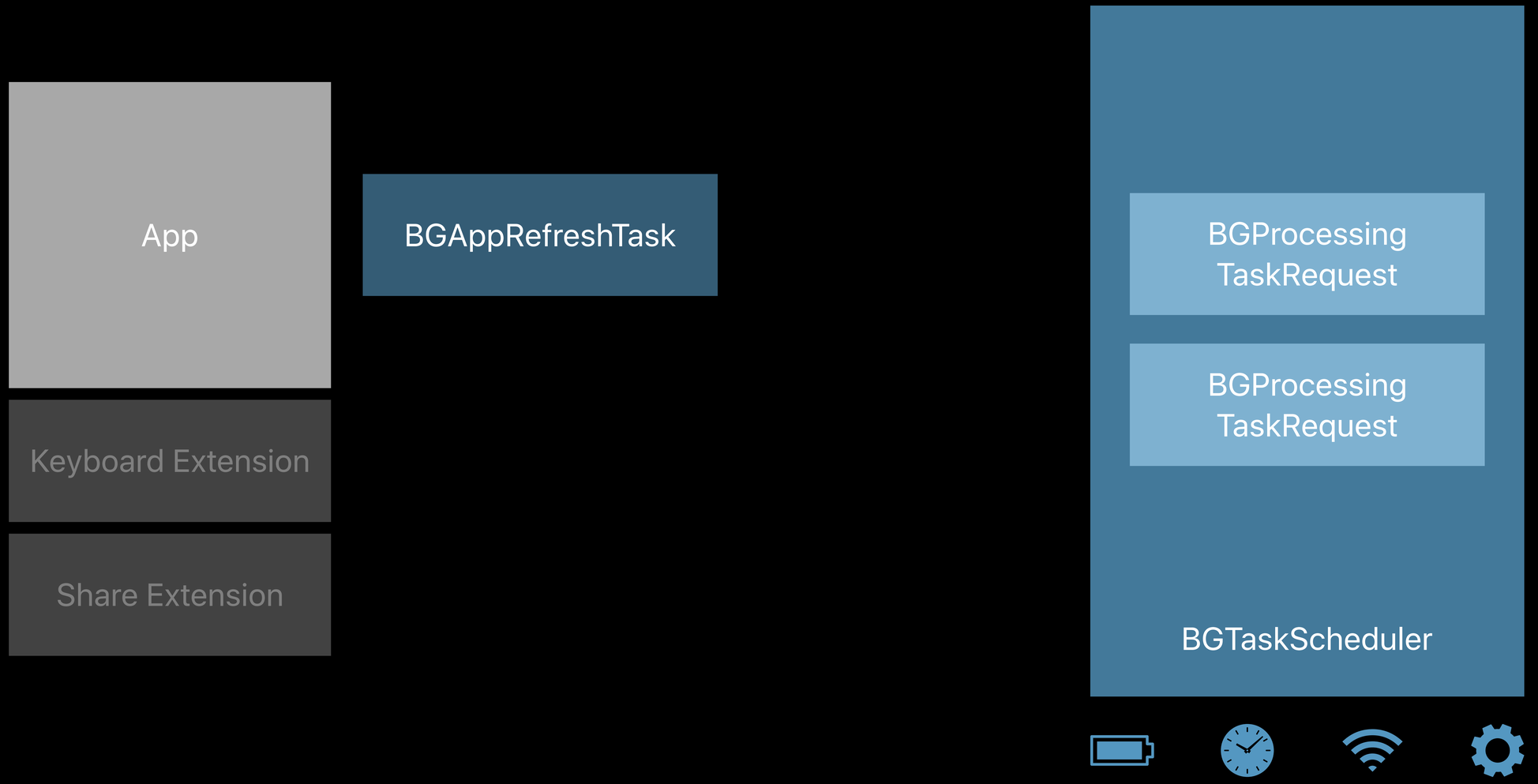
Using BackgroundTasks

Using BackgroundTasks

Using BackgroundTasks

Using BackgroundTasks

Using BackgroundTasks



Using BackgroundTasks

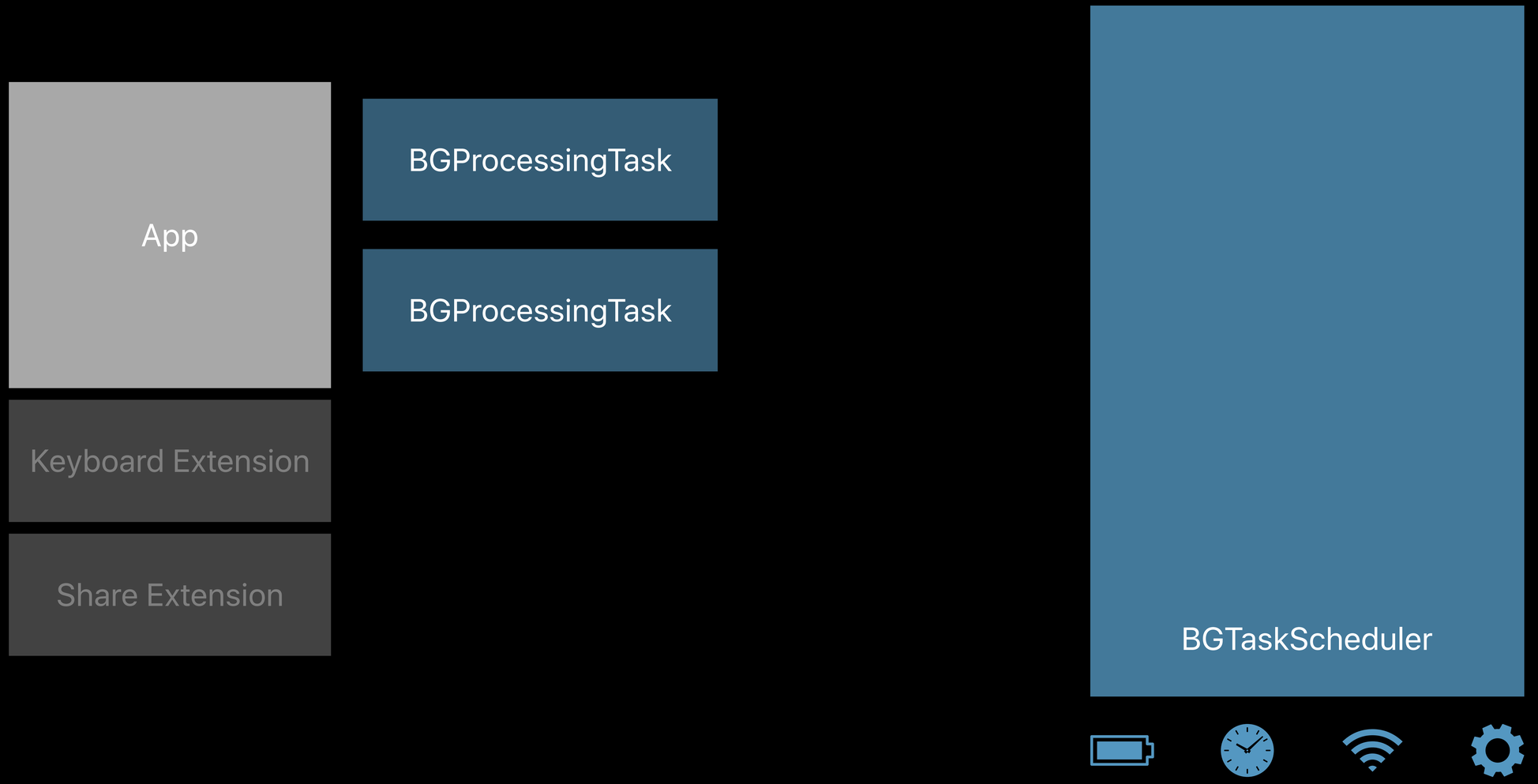
Using BackgroundTasks

Using BackgroundTasks

Using BackgroundTasks

Using BackgroundTasks

Using BackgroundTasks



Using BackgroundTasks

Using BackgroundTasks

Demo

Additional Considerations

Additional Considerations

Don't set `earliestBeginDate` too far into the future

Additional Considerations

Don't set `earliestBeginDate` too far into the future

Ensure files are accessible while device is locked

- `FileProtectionType.completeUntilFirstUserAuthentication`

Additional Considerations

Don't set `earliestBeginDate` too far into the future

Ensure files are accessible while device is locked

- `FileProtectionType.completeUntilFirstUserAuthentication`

UIScene apps should call `UIApplication.requestSceneSessionRefresh(_:)`

Additional Considerations

Don't set `earliestBeginDate` too far into the future

Ensure files are accessible while device is locked

- `FileProtectionType.completeUntilFirstUserAuthentication`

UIScene apps should call `UIApplication.requestSceneSessionRefresh(_:)`

Consider calling `BGTaskScheduler.submit(_:)` on a background queue
if submitting at launch

Summary

Summary

Be conscientious when running in the background

Summary

Be conscientious when running in the background

Use the right background mode for the job

Summary

Be conscientious when running in the background

Use the right background mode for the job

Schedule deferrable work with BackgroundTasks

More Information

developer.apple.com/wwdc19/707

Performance, Power, Crashes, and Debugging Lab	Wednesday, 3:00
CallKit, VoIP Pushes, and Identity Lookup Lab	Friday, 3:00


WWDC19