When Things Go Wrong

Error Handling and Logging Best Practices for Developer Sanity

Justin Williams

www.justinw.me

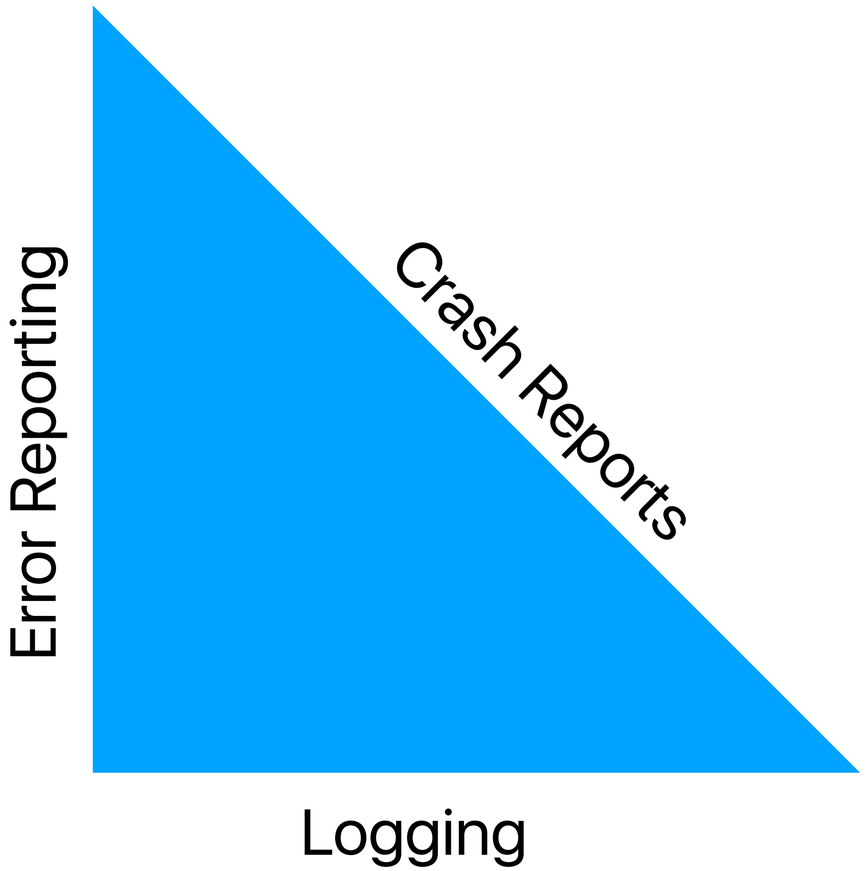

Solar System Mac File Edit Format View Window Help
Mon 10:09 AM
SceneViewController.swift
Main.storyboard
Solar System
README.md
Shared
Networking
Primitives
PlanetUpdateService.swift
CameraModel Classes
Fly-By Classes
Model Classes
Model Protocols
PhysicsBody.swift
OrbitingBody.swift
View Classes
View Model Classes
Solar System iOS
Solar System Mac
Unit Tests
Scene Assets
Assets.xcassets
AstronomicalObjectViewItem.swift
AstronomicalObjectViewItem.xib
AppDelegate.swift
PlanetDataFetcher.swift
ParticleSystemsAnimator.swift
SceneViewController.swift
SceneWindowController.swift
SolarSystemPlanetsDataSource.swift
SceneViewAppearanceManager.swift
NetworkRequestProgressView.swift
SingleAstronomicalObjectScene.scn
Main.storyboard
MainWindow.xib
PlanetDetailsWindow.xib
Info.plist
Solar_System_Mac.entitlements
Resources
PlanetPhysics
Solar System iOS UI Tests
Instrumentation
Products
var isUpdatingNewsFeed: Bool {
let timerIsActive = newsRequestDispatchTimer != nil
return timerIsActive
}
// MARK: - User Interaction
/// Performs network request to Solar System REST service which asks about news and planets.
func refreshPlanetsAndNews() {
// MARK: - N
internal fun
// TODO:
internal fun
// TODO:
Window Controller
Scene View Controller
Mercury
Venus
Earth
Mars
Jupiter
Saturn
Uranus
Neptune
Earth
Milky Way
37° 20' 4" N, 122° 0' 32" W
Current Location
05:48
Sunrise
20:25
Sunset
Radius: 6371 km
Mass: 5.97×10²⁴ kg
Gravity: 9.81 m/s²
View as: Dark Appearance
Appearance

Via https://unsplash.com/photos/EiPJjVgP5-U

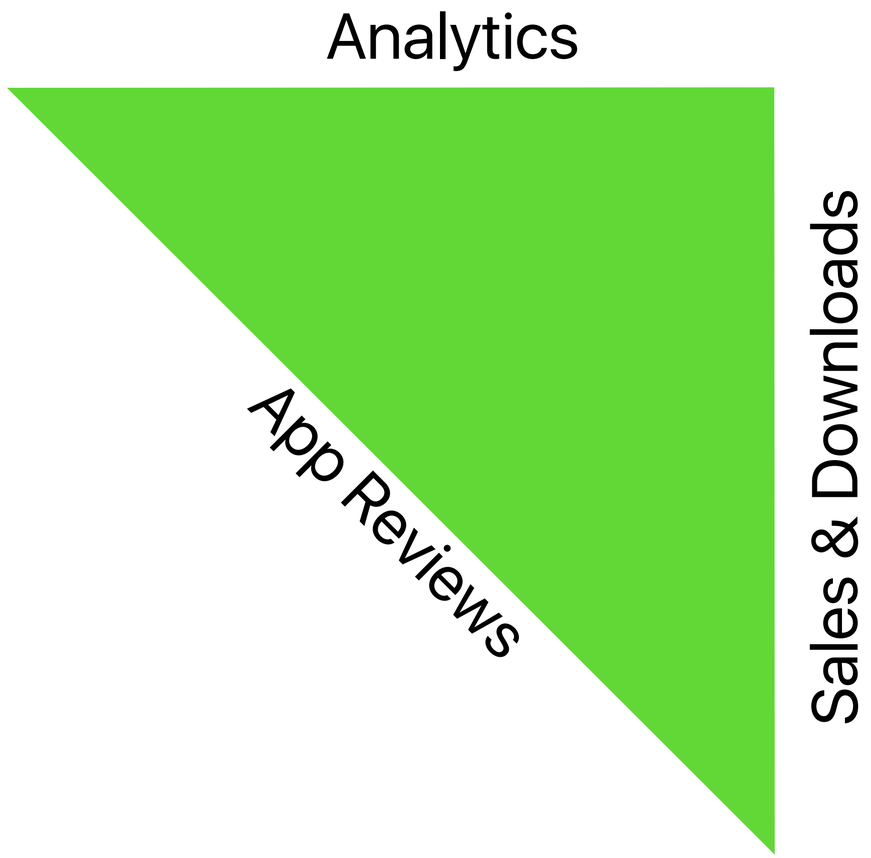
Observability

That's Our Agenda

Developer

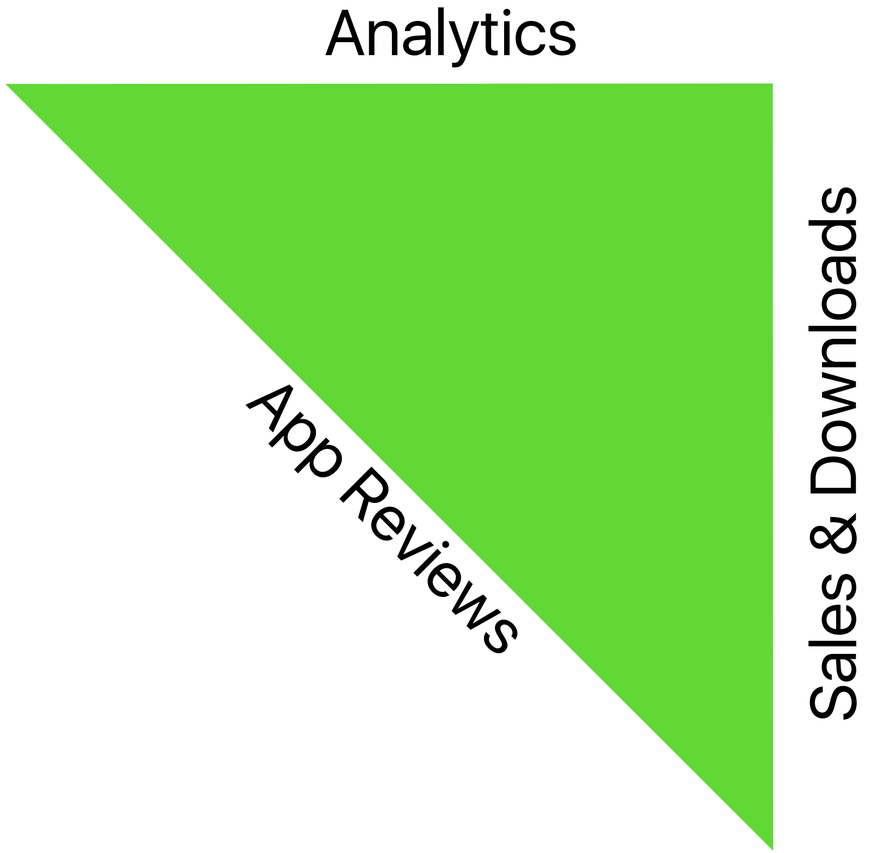


Product

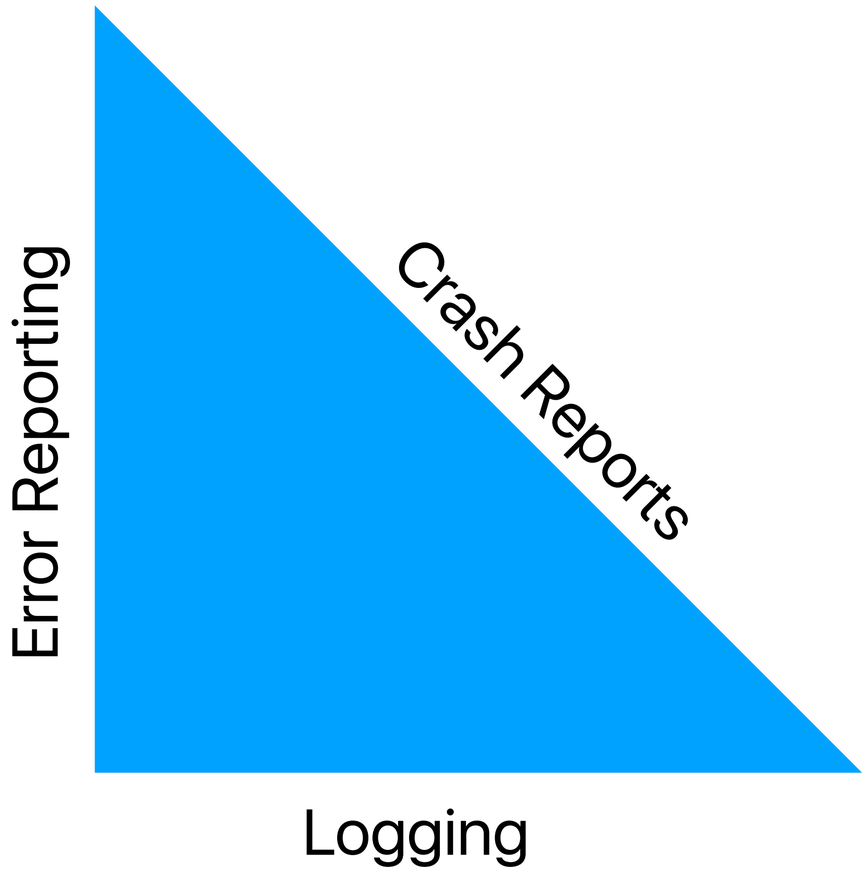


Product Metrics

- Google Analytics
- Some sort of push notification analytics
- Probably another analytics library
- User reviews on the app store
- Sales & Downloads reports



Developer Metrics

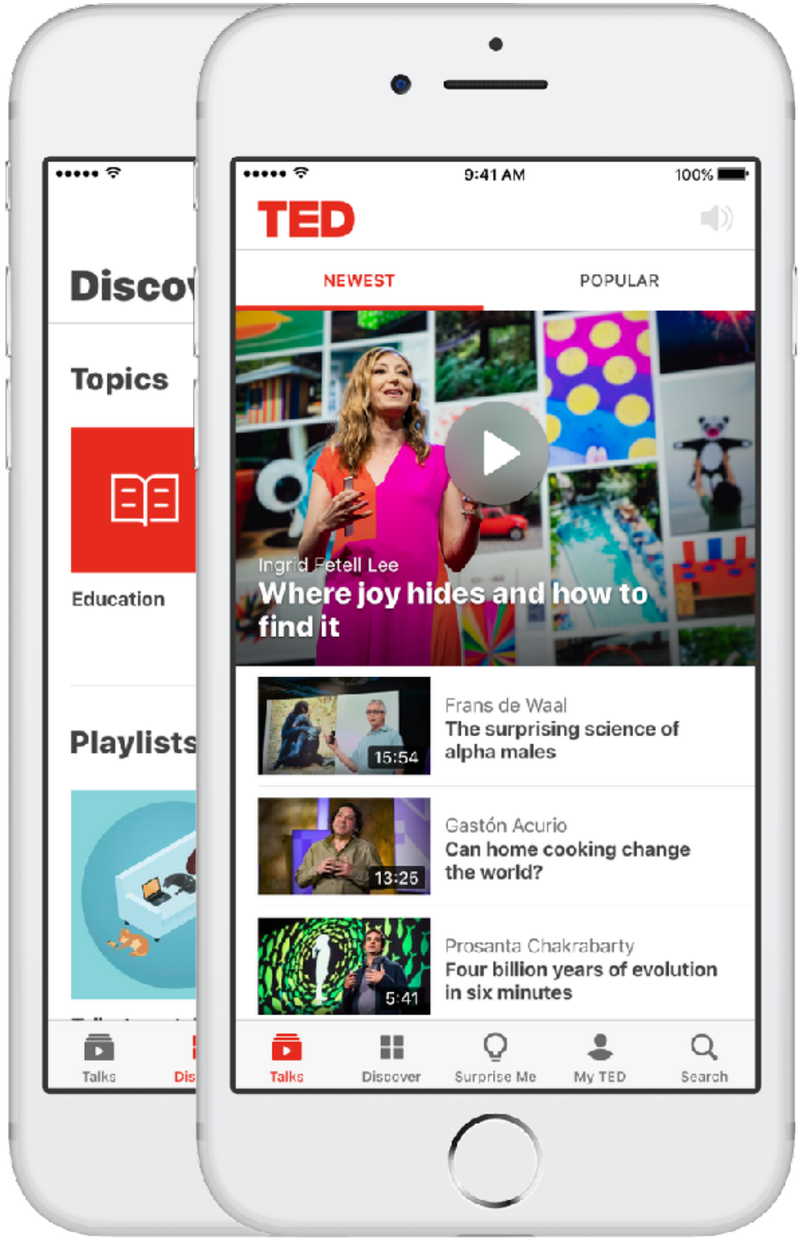


- Device Logging
- Crash Reports
- Error Reports

Developer Metrics

- **Crash Reporting**: The 🔥 you can see.
- **Error reporting**: The 🔥 you can’t see.
- **Logging**: How the 🔥 are starting, hopefully.

Logging



TED for iOS

A history of logging

- **2012:** NSLog + Objective-C macros.
- **2013**: CocoaLumberjack introduced to the project.
- **2015:** CocoaLumberjack replaced by in-house “Aspen” framework with Swift support.
- **2016**: Aspen is replaced by a wrapper around `os.log`.
- **2018**: The `os.log` wrapper is deprecated for direct `os_log` access.

OSLog

- Introduced in iOS 10
- Works across watchOS, iOS, tvOS, and macOS
- Unification of syslog and Apple System Logger APIs

Simple

```
os_log(message: StaticString, log: OSLog, type: OSLogType, args: CVarArg...)
```

Thread Safe

Fast

```
let launchDate = Date()
let formatter = DateFormatter()
let dateString = formatter.string(from: launchDate)

logVerbose("Launched at \(dateString)")
```

```
let launchDate = Date()
os_log("Launched at %{time_t}d”,
       time_t(launchDate.timeIntervalSince1970))
```

Powerful

Console Demo

Please be kind, Demo Gods

Using OSLog

1. Import `os.log`

2. Start using `os_log` calls in your code.

```
let log = OSLog(subsystem: "JWWSubsystem", category: "Cats")
os_log("Cats, cats, cat!", log: log)
```

Levels

Debug	For development use only
Info	For additional info that may be useful but not as verbose.
Default	The default level that is always visible to all users.
Fault	Issues that involve multiple processes.
Error	Errors in the current process or framework.

TED Levels

Debug	For development use only
Default	The default level that is always visible to all users.
Error	Errors in the current process or framework.

Disk vs Memory Persistence

Debug	Memory
Info	Memory
Default	Disk
Fault	Disk
Error	Disk

Private Data

```
struct Account {
    let email: String
    let password: String
}

let account = Account(email: "justinw@me.com", password: “1ns3cure”)
os_log("User signed in. Email: %@. Password: %@“,
       account.email, account.password)
```

User signed in. Email: **<private>**. Password: **<private>**

```
let account = Account(email: "justinw@me.com", password: “1ns3cure”)
os_log("User signed in. Email: %{public}@. Password: %@“,
       account.email, account.password)
```

User signed in. Email: **justinw@me.com**. Password: **<private>**

```
let accountID = Int.random(in: 1..<5000)
os_log("Account ID = %{private}i", accountID)
```

Data Formatting

```
let launchDate = Date()
os_log("Launched at %{time_t}d”,
       time_t(launchDate.timeIntervalSince1970))
```

```
let isASuperGenius = Bool.random()
os_log("Are you a super genius? %@“,
      isASuperGenius ? "YES" : "NO")
os_log("Are you a super genius? %{BOOL}d", isASuperGenius)
os_log("Are you a super genius? %{bool}d”, isASuperGenius)
```

Are you a super genius? **YES**

Are you a super genius? **YES**

Are you a super genius? **true**

The unified logging system considers dynamic strings and complex dynamic objects to be private, and does not collect them automatically. To ensure the privacy of users, it is recommended that log messages consist strictly of static strings and numbers, which are collected automatically by the system. In situations where it is necessary to capture a dynamic string, and it would not compromise user privacy, you may explicitly declare the string public by using the public keyword in the log format string. For example, %{public}s. Log arguments can also be specified as private by using the private keyword in the log format string. For example, %{private}d.

To format a log message, use a printf(3) format string. You may also use the "%@" format specifier for use with Obj-C/CF/Swift objects, and %.*P which can be used to decode arbitrary binary data. The logging system also supports custom decoding of values by denoting value types inline in the format %{value_type}d. The built-in value type decoders are:

Value type	Custom specifier	Example output
BOOL	%{BOOL}d	YES
bool	%{bool}d	true
darwin.errno	%{darwin.errno}d	[32: Broken pipe]
darwin.mode	%{darwin.mode}d	drwxr-xr-x
darwin.signal	%{darwin.signal}d	[sigsegv: Segmentation Fault]
time_t	%{time_t}d	2016-01-12 19:41:37
timeval	%{timeval}.*P	2016-01-12 19:41:37.774236
timespec	%{timespec}.*P	2016-01-12 19:41:37.2382382823
bitrate	%{bitrate}d	123 kbps
iec-bitrate	%{iec-bitrate}d	118 Kibps
uuid_t	%{uuid_t}.16P	10742E39-0657-41F8-AB99-878C5EC2DCAA
sockaddr	%{network:sockaddr}.*P	fe80::f:86ff:fee9:5c16
in_addr	%{network:in_addr}d	127.0.0.1
in6_addr	%{network:in6_addr}.16P	fe80::f:86ff:fee9:5c16

:

Error Handling

I'm afraid I've got some bad news. . .

Error Formatters

```
%{errno}

%{signal}

%{mode}
```

Happy Path

```
os_log("Error %{errno}d”,
       POSIXError.Code.EADDRNOTAVAIL.rawValue)
```

Error [49: Can't assign requested address]

Sad Path

```
let error = NSError(domain: NSURLErrorDomain,
      code: NSURLErrorTimedOut,
      userInfo: nil)

os_log("Error %{errno}d", error.code)
```

Error [-1001: Unknown error: -1001]

```
public protocol ErrorReporting {
    var loggingDescription: String { get }
}
```

```
extension NSError: ErrorReporting {
    public var loggingDescription: String {
        let desc = userInfo[NSDebugDescriptionErrorKey] as?
String ?? localizedDescription

        return "{ \(desc). domain: \(domain). code: \(code) }"
    }
}
```

Almost Happy Path

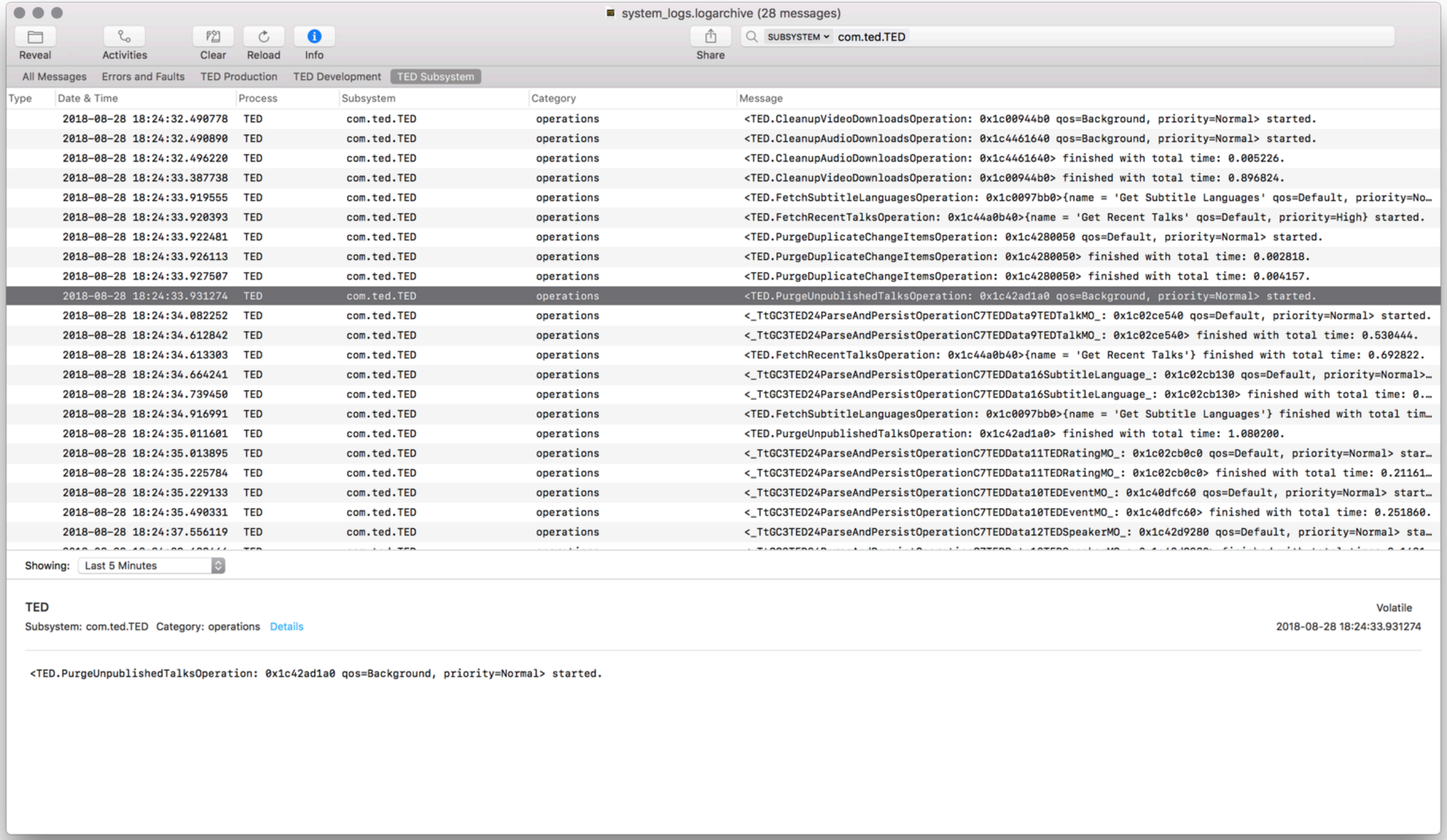
```
os_log("Error %{public}@“, error.loggingDescription)
```

Error {
The operation couldn’t be completed.
domain: NSURLErrorDomain.
code: -1001
}

Prove Me Wrong

I'm asking to be "Well, actually. . . "-ed

sysdiagnose


system_logs.logarchive (28 messages)
Reveal
Activities
Clear
Reload
Info
Share
SUBSYSTEM com.ted.TED
All Messages
Errors and Faults
TED Production
TED Development
TED Subsystem
Type
Date & Time
Process
Subsystem
Category
Message
2018-08-28 18:24:32.490778 TED com.ted.TED operations <TED.CleanupVideoDownloadsOperation: 0x1c00944b0 qos=Background, priority=Normal> started.
2018-08-28 18:24:32.490890 TED com.ted.TED operations <TED.CleanupAudioDownloadsOperation: 0x1c4461640 qos=Background, priority=Normal> started.
2018-08-28 18:24:32.496220 TED com.ted.TED operations <TED.CleanupAudioDownloadsOperation: 0x1c4461640> finished with total time: 0.005226.
2018-08-28 18:24:33.387738 TED com.ted.TED operations <TED.CleanupVideoDownloadsOperation: 0x1c00944b0> finished with total time: 0.896824.
2018-08-28 18:24:33.919555 TED com.ted.TED operations <TED.FetchSubtitleLanguagesOperation: 0x1c0097bb0>{name = 'Get Subtitle Languages' qos=Default, priority=No…
2018-08-28 18:24:33.920393 TED com.ted.TED operations <TED.FetchRecentTalksOperation: 0x1c44a0b40>{name = 'Get Recent Talks' qos=Default, priority=High} started.
2018-08-28 18:24:33.922481 TED com.ted.TED operations <TED.PurgeDuplicateChangeItemsOperation: 0x1c4280050 qos=Default, priority=Normal> started.
2018-08-28 18:24:33.926113 TED com.ted.TED operations <TED.PurgeDuplicateChangeItemsOperation: 0x1c4280050> finished with total time: 0.002818.
2018-08-28 18:24:33.927507 TED com.ted.TED operations <TED.PurgeDuplicateChangeItemsOperation: 0x1c4280050> finished with total time: 0.004157.
2018-08-28 18:24:33.931274 TED com.ted.TED operations <TED.PurgeUnpublishedTalksOperation: 0x1c42ad1a0 qos=Background, priority=Normal> started.
2018-08-28 18:24:34.082252 TED com.ted.TED operations <_TtGC3TED24ParseAndPersistOperationC7TEDData9TEDTalkMO_: 0x1c02ce540 qos=Default, priority=Normal> started.
2018-08-28 18:24:34.612842 TED com.ted.TED operations <_TtGC3TED24ParseAndPersistOperationC7TEDData9TEDTalkMO_: 0x1c02ce540> finished with total time: 0.530444.
2018-08-28 18:24:34.613303 TED com.ted.TED operations <TED.FetchRecentTalksOperation: 0x1c44a0b40>{name = 'Get Recent Talks'} finished with total time: 0.692822.
2018-08-28 18:24:34.664241 TED com.ted.TED operations <_TtGC3TED24ParseAndPersistOperationC7TEDData16SubtitleLanguage_: 0x1c02cb130 qos=Default, priority=Normal>…
2018-08-28 18:24:34.739450 TED com.ted.TED operations <_TtGC3TED24ParseAndPersistOperationC7TEDData16SubtitleLanguage_: 0x1c02cb130> finished with total time: 0.…
2018-08-28 18:24:34.916991 TED com.ted.TED operations <TED.FetchSubtitleLanguagesOperation: 0x1c0097bb0>{name = 'Get Subtitle Languages'} finished with total tim…
2018-08-28 18:24:35.011601 TED com.ted.TED operations <TED.PurgeUnpublishedTalksOperation: 0x1c42ad1a0> finished with total time: 1.080200.
2018-08-28 18:24:35.013895 TED com.ted.TED operations <_TtGC3TED24ParseAndPersistOperationC7TEDData11TEDRatingMO_: 0x1c02cb0c0 qos=Default, priority=Normal> star…
2018-08-28 18:24:35.225784 TED com.ted.TED operations <_TtGC3TED24ParseAndPersistOperationC7TEDData11TEDRatingMO_: 0x1c02cb0c0> finished with total time: 0.21161…
2018-08-28 18:24:35.229133 TED com.ted.TED operations <_TtGC3TED24ParseAndPersistOperationC7TEDData10TEDEventMO_: 0x1c40dfc60 qos=Default, priority=Normal> start…
2018-08-28 18:24:35.490331 TED com.ted.TED operations <_TtGC3TED24ParseAndPersistOperationC7TEDData10TEDEventMO_: 0x1c40dfc60> finished with total time: 0.251860.
2018-08-28 18:24:37.556119 TED com.ted.TED operations <_TtGC3TED24ParseAndPersistOperationC7TEDData12TEDSpeakerMO_: 0x1c42d9280 qos=Default, priority=Normal> sta…
Showing: Last 5 Minutes
TED
Volatile
Subsystem: com.ted.TED Category: operations Details
2018-08-28 18:24:33.931274
<TED.PurgeUnpublishedTalksOperation: 0x1c42ad1a0 qos=Background, priority=Normal> started.

OSLog Limitations

- No file-based logging.
- Objective-C designed. Swift friendly.
- No way to adjust logging settings on iOS devices (that I could get to work).
- Binary log formats are not backwards compatible.
- Improvements come as Apple needs them.

Best Practices

- Don’t wrap the API.
- No really! Don’t wrap the API!
- Use a single subsystem for your application.
- Use a small subset of categories for personal sanity.

Sample Code

https://github.com/justin/360iDev2018

```swift
import Foundation
import os.log

public struct Logging {
    public typealias Subsystem = String

   /// The logging categories supported by the different app subsystems.
   public enum Category: String {
        case `default`
        case database
        case networking
        case operations
        case playback
        case reporting
        case ui
    }

   public static func makeLog(subsystem: Subsystem = .default, category: Category) -> OSLog {
        return OSLog(subsystem: subsystem, category: category)
    }
}

extension Logging.Subsystem {
    /// Convenience accessor for the setting the subystem to the main app's bundle identifier.
    public static var `default` = Bundle.main.bundleIdentifier ?? ""
}


public extension OSLog {
    /// Initialize a new OSLog instance using the parent application bundle identifier and a custom category.
    ///
    /// - Parameter subsystem: The subsystem for your logging instance.
    /// - Parameter category: The category for your logging instance.
    public convenience init(subsystem: Logging.Subsystem = .default, category: Logging.Category = .default) {
        self.init(subsystem: subsystem, category: category.rawValue)
    }
}
```

```
import Foundation
import os.log

struct Log {
    static let `default` = Logging.makeLog(category: .default)
    static let networking = Logging.makeLog(category: .networking)
}

os_log("Debug Level", log: Log.default, type: .debug)
os_log("Info Level", log: Log.networking, type: .info)
os_log("Default Level", log: Log.default, type: .default)
os_log("Fault Level", log: Log.networking, type: .fault)
os_log("Error Level", log: Log.default, type: .error)
```

Sample Code

https://github.com/justin/360iDev2018

Thank You.