macOS Networking

macOS 10.15 • Xcode 11

STUDENT GUIDE

ABOUTOBJECTS

Contact

About Objects
11911 Freedom Drive
Suite 700
Reston, VA 20190

main: 571–346–7544
email: info@aboutobjects.com
web: www.aboutobjects.com

Course Information

Author: Jonathan Lehr
Revision: 1.0.0
Last Update: 6/1/2020

Classroom materials for a course that provides a rapid introduction to macOS network programming. Geared to developers interested in learning to use Network Extensions on the Macintosh platform.

Copyright Notice

© Copyright 2020 About Objects, Inc.

Under the copyright laws, this documentation may not by copied, photographed, reproduced, translated, or reduced to any electronic medium or machine–readable form, in whole or in part, without express written consent.

All other copyrights, trademarks, registered trademarks and service marks throughout this document are the property of their respective owners.

All rights reserved worldwide. Printed in the USA.

macOS Networking

STUDENT GUIDE

About the Author

Jonathan Lehr

Co-Founder and VP, Training
About Objects

jonathan@aboutobjects.com

Section 1: Security

Copyright © 2020, About Objects

Kernel Extensions

- Kernel Extensions (kexts) deprecated in macOS Catalina.
- As of release10.15.4
  - Use of a kext notifies user that the software includes deprecated API.
  - Suggests that user contact developer for an alternative.
- Kexts run in kernel space so errors/malicious code can:
  - Compromise security
  - Crash the operating system

Copyright © 2020, About Objects

Rootless Filesystem

- An earlier release of macOS added System Integrity Protection (SIP) and disabled write access to `/System`, `/bin`, `/usr/` and `/sbin`.
- In Catalina, entire OS is in a separate, read-only partition named **Macintosh HD**.
  - Only code signed by Apple can update.
  - Root directory (`/`) isn't writeable.

Copyright © 2020, About Objects

Catalina Security Features

🔒 Access Control

- Catalina apps must get user permission to access user folders and volumes (e.g., **Documents**, **Desktop**, and **Downloads)**.
- Users enable access via **System Preferences | Security & Privacy | Privacy** tab.

 Gatekeeper

- Enforces app code signing and notarization.
- Apps quarantined by default until verified by Gatekeeper or explicitly enabled by user.

WWDC 2019 Video: Advances in macOS Security

https://developer.apple.com/videos/play/wwdc2019/701

Copyright © 2020, About Objects

Code Signing

- Certifies that an app was created by a given developer.
- Requires a developer ID provided by an account on developer.apple.com.
- Done automatically during Xcode builds.
- Can optionally be performed via `codesign` command-line tool.

Example

To view entitlements for Coolness.app:

```
codesign –d ––entitlements :– Coolness.app
```

Copyright © 2020, About Objects

App Sandbox

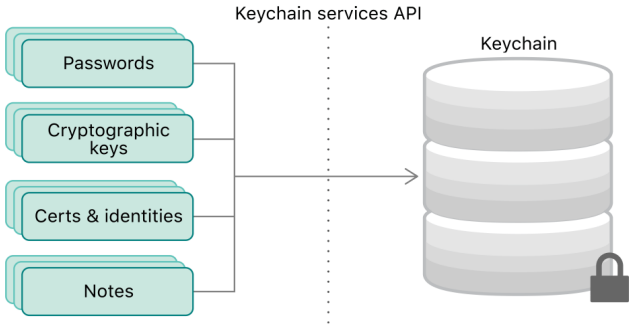
- An environment in which apps are executed that limits:
  - The visibility of the filesystem
  - An app's capabilities (i.e., access to system resources)
- Enables privilege separation, wherein an app delegates certain behaviors to one or more sandboxed processes that have additional privileges not available to the app itself.
  - XPC was added in 10.7 to make this easier to implement.
- Each sandboxed app is restricted to its own, unique container directory, in which it has full read-write access.
  - User can give an app permission to access specific files outside its sandbox on a case-by-case basis.
  - An app also has some access to specific world-readable locations, e.g. `tmp` directories, `bin` directories, etc.
  - If an app requests an app group entitlement, it can also access sandbox directories of other apps within its group.

Copyright © 2020, About Objects

Keychain

- Managed by user through Keychain Access app.
- Managed by apps via Keychain Services C API.

Securing User Secrets



- On macOS, each item is bound to a collection of apps and operations (ACL) governing access.
    - System automatically checks ACLs whenever an app requests access to a protected item.
    - If not, the system automatically prompts the user for confirmation.
    - If user confirms, the system automatically adds the app to the list of trusted apps for that item.

Copyright © 2020, About Objects

App Transport Security

- Enforces consistent use of TLS on network connections made by apps and app extensions that use the URL Loading System.
  - Automatically blocks connections that don't meet security requirements.
  - Exceptions can be requested in Info.plist.
  - Exceptions may trigger additional App Store review.
- Doesn't apply to calls made by apps using lower-level networking interfaces.

Copyright © 2020, About Objects

Section 2: Concurrency

Copyright © 2020, About Objects

Run Loops

- **NSRunLoop** objects manage input sources and timers.
- Example input sources:
  - Touch, motion, and keyboard events
  - **performSelector:onThread:...** messages
- A run loop must be present and running in order for the following to work:
  - **performSelector…** methods
  - **NSTimer** instances
  - Keeping a thread alive to perform work periodically
  - Using Mach ports or custom input sources to communicate with other threads

Copyright © 2020, About Objects

Run Loop Modes

- Run loops can run in several different *modes*.
  - Modes allow events from a given set of input sources to be funneled to a specific observer.
- Predefined modes are as follows:

  Default: **NSDefaultRunLoopMode**

  Modal: **NSModalPanelRunLoopMode**

  Event Tracking: **NSEventTrackingRunLoopMode**

  Common Modes: **NSRunLoopCommonModes**
- You can also define custom run loop modes.

Copyright © 2020, About Objects

Multithreading

- In iOS, every thread must have its own:
  - autorelease pool
  - run loop
- If you create your own threads, you're responsible for creating and managing these yourself.
- Pitfalls of hand-written threading code:
  - Hard to write and debug
  - Resource intensive
- Instead use **Grand Central Dispatch** or higher-level APIs that are layered on top of GCD (e.g. **NSOperation**).

Copyright © 2020, About Objects

Grand Central Dispatch

- Block-based, C API
- Creates and manages:
  - highly-optimized thread pool
  - global concurrent queue
  - main (serial) queue
- GCD tasks can be
  - grouped
  - dispatched based on timer intervals
  - dispatched in a loop

Copyright © 2020, About Objects

DispatchSource

- Coordinates the processing of file-system events, timers, and UNIX signals.
- Provides API for
    - Receiving messages from Mach ports
    - Monitoring Unix signals
    - Being notified of changes to memory pressure
    - Monitoring reads and writes using a file descriptor
    - Monitoring timer events

Copyright © 2020, About Objects

DispatchIO / DispatchData

DispatchIO

- Manages operations on a file descriptor.
- May be either stream-based or random access.

DispatchData

- Manages a memory buffer.
- Capable of consolidating multiple, discontiguous blocks.

Copyright © 2020, About Objects

NSOperation

- **NSOperation** is an abstract class used to encapsulate state and behavior for a given task.
  - Concrete subclasses are **NSInvocationOperation** and **NSBlockOperation**.
  - You can easily create your own custom subclasses if desired.
- Operations can be executed in either of two ways:
  - Directly (by sending them a **run** message)
  - By adding them to an **NSOperationQueue**.
- An operation can wrap one or more dependent operations.
  - Will not execute until all its dependent operations have completed.

Copyright © 2020, About Objects

NSOperationQueue

- Operations start executing as soon as they're added to a queue.
  - Automatically removed from queue when finished.
  - Queue holds strong references to its operations.
- To add operations to a queue:

```
open func addOperation(_ op: Operation)

@available(OSX 10.6, *)
open func addOperations(_ ops: [Operation], waitUntilFinished wait: Bool)

@available(OSX 10.6, *)
open func addOperation(_ block: @escaping () -> Void)
```

- Supporting concurrent operations:

```
open var maxConcurrentOperationCount: Int
```

Copyright © 2020, About Objects

Copyright © 2020, About Objects

Section 3: XPC Services

Copyright © 2020, About Objects

Overview

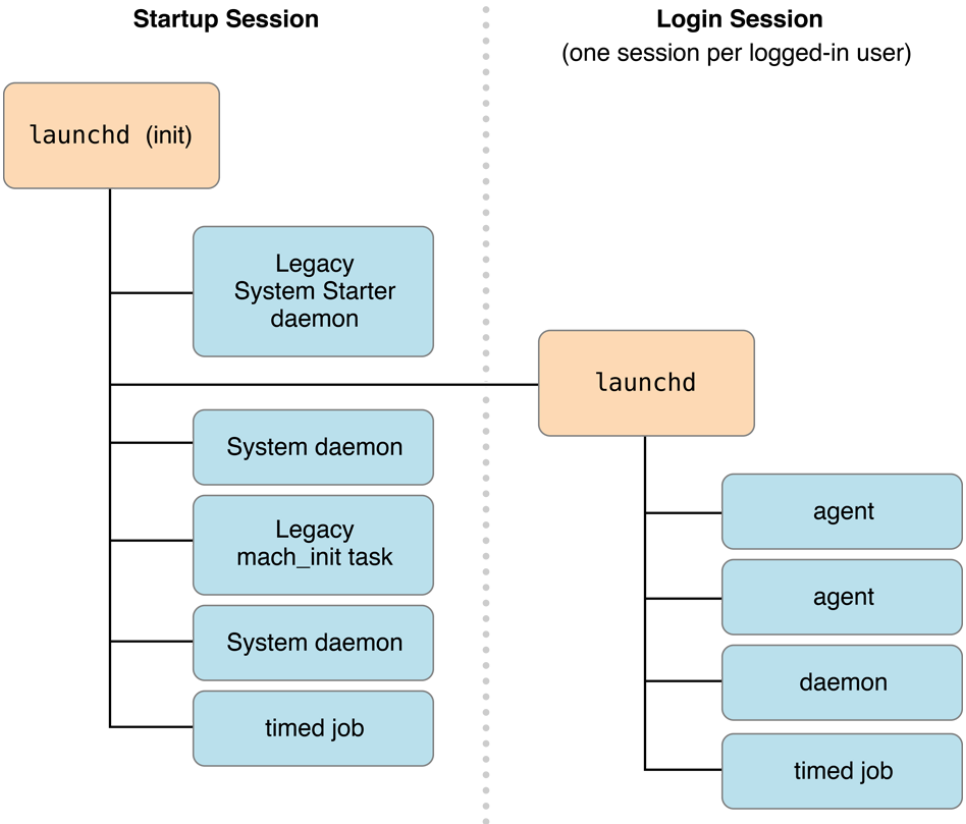
- XPC Services is an API defined in `libSystem`.
- Lightweight mechanism for interprocess communication that runs in user space.
- Benefits
    - Increased stability (can make apps more crash-resistant)
    - Enhanced security via privilege separation
    - Distributed as part of app bundle, so no installer required.

Copyright © 2020, About Objects

Working with launchd

- launchd can provide XPC services with the following:
  - Launch on demand
  - Restart on crash
  - Terminate when idle

launchd Session Contexts

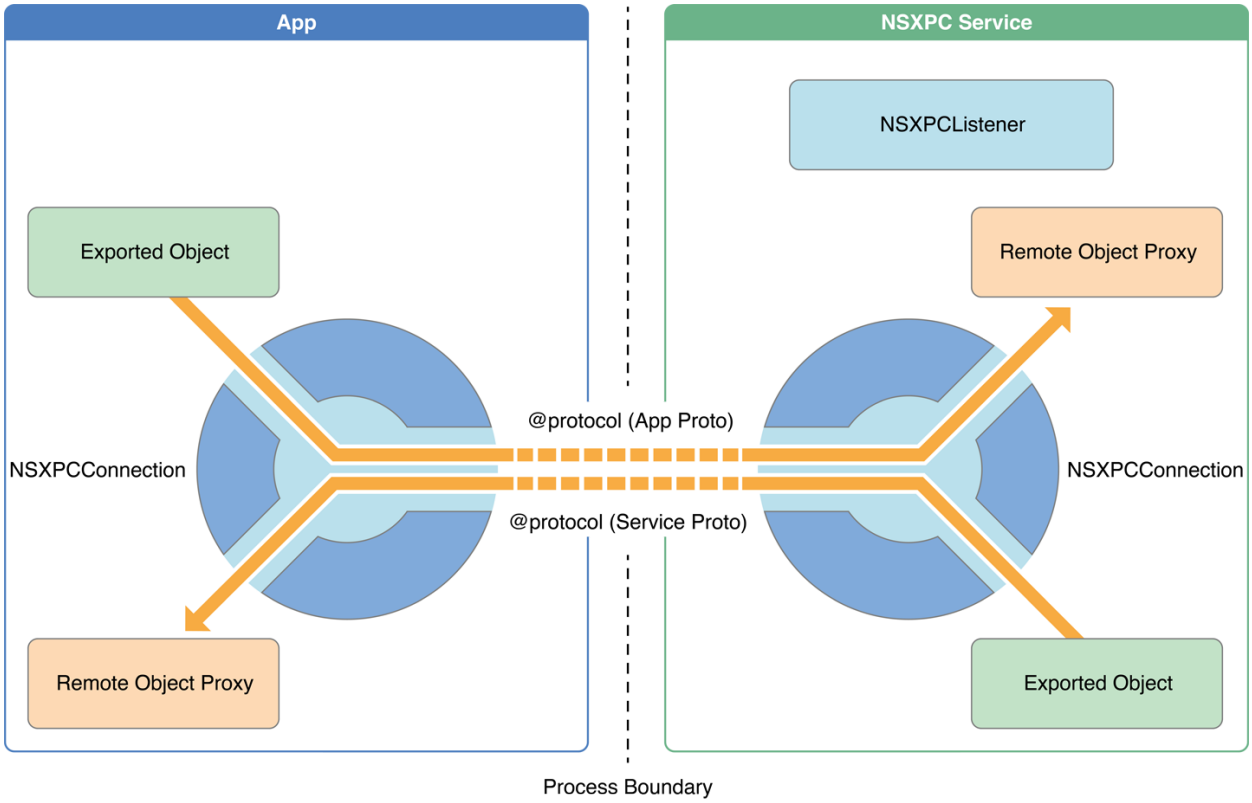


Copyright © 2020, About Objects

NSXPCConnection API

- An abstraction of a remote procedure call (RPC) interface.
- Allows objects in separate processes to call methods on one another.
    - Automatically serializes/deserializes objects and data structures as needed for data transmission.
    - Objects transported via an NSXPC connection must conform to `NSSecureCoding`.

NSXPC Architecture



Copyright © 2020, About Objects

The NSSecureCoding Protocol

- Classes that adopt NSSecureCoding and override `initWithCoder(_:)` must override `supportsSecureCoding` and return `true`.

```
public protocol NSSecureCoding : NSCoding {
      static var supportsSecureCoding: Bool { get }
}
```

- The decoding methods in the NSSecureCoding API require passing the expected type as an additional parameter, for example:

```
public func decodeObject<DecodedObjectType>(of cls: DecodedObjectType.Type,
                                            forKey key: String) -> DecodedObjectType?
                                            where DecodedObjectType:
                                            NSObject, DecodedObjectType : NSCoding
```

- This prevents attacks that substitute a different class than the expected one

Copyright © 2020, About Objects

Communicating Across XPC

- NSXPCConnection is primary means of creating and configuring the communication mechanism between two processes.
  - Your app creates an instance to connect to a service.
  - App then communicates through a remote object proxy.
- An XPC service implements the XPCListenerDelegate protocol to listen for a connection.
  - Publishes an API (protocol) for remote requests.
  - Configures an object that provides implementations of the defined API.

For More Information:
XPC · objc.io

Copyright © 2020, About Objects

Section 4: System Extensions

Copyright © 2020, About Objects

System Extensions

- Allow code running in user space to access system functionality that previously required writing kernel extensions.
- Distributed along with app as nested bundle.
- Deployed by the OS to a hidden system directory.
- Automatically started by the OS at boot time.

WWDC 2019 Video: System Extensions and DriverKit

https://developer.apple.com/videos/play/wwdc2019/702

Copyright © 2020, About Objects

Frameworks

Set of frameworks includes:

- SystemExtensions
  - Installing and updating extensions
- DriverKit, USBDriverKit, NetworkingDriverKit, etc.
  - Fundamental device driver behaviors
- EndpointSecurity
  - C API for monitoring system events for malicious activity
- NetworkExtension
  - Customization of core networking features including DNS, WiFi, VPN, and content filtering

Copyright © 2020, About Objects

Configuration

- Extension must match bundle identifier plus appropriate file extension
- Bundle ID typically reverse domain path.
  - Example: `com.aboutobjects.coolstuff`
- File extensions include `.systemextension`, `.dext`, etc.
  - Example extension ID:

```
com.aboutobjects.coolstuff.systemextension
```

- Use the same Team ID when signing the extension that was used to sign your app (unless the extension has the `com.apple.developer.system-extension.redistributable` entitlement).

Copyright © 2020, About Objects

Section 5: Network Extensions

Copyright © 2020, About Objects

Platform Modernization

- Apple announced deprecation of kernel extensions at WWDC 2019.
- Future OS releases won't load kernel extensions that use deprecated APIs without rework.
- Deprecated network filtering APIs include the following:

```
ipf_addv4
ipf_addv6
ipf_inject_input
ipf_inject_output
ipf_remove
sflt_attach
sflt_detach
sflt_register
sflt_unregister
sock_accept
sock_bind
sock_close
sock_connect
sock_getpeername
sock_getsockname
sock_getsockopt
sock_gettype
sock_inject_data_in
sock_inject_data_out
sock_ioctl
sock_isconnected
sock_isnonblocking
sock_listen
sock_receive
sock_receivembuf
sock_send
sock_sendmbuf
sock_setpriv
sock_setsockopt
sock_shutdown
sock_socket
sockopt_copyin
sockopt_copyout
sockopt_direction
sockopt_level
sockopt_name
sockopt_valsize
```

Copyright © 2020, About Objects

NetworkExtension Framework

- Some features are iOS-only or require App Store distribution.
- Supports the following capabilities:
  - Changing the system's Wi-Fi configuration (iOS only)
  - Integration with hotspot network subsystem (iOS only)
  - Creating and managing VPN configurations using built-in or custom VPN protocols (Mac App Store only)
  - Implementing an app proxy (Mac App Store only)
  - Content filtering
  - Packet tunneling
  - Implementing a DNS proxy
- Require the System Extension entitlement.
- Key: `com.apple.developer.system-extension.install`

WWDC 2019 Video: Network Extensions for the Modern Mac

https://developer.apple.com/videos/play/wwdc2019/714/

Copyright © 2020, About Objects

Content Filtering

Consists of two providers that collaborate with one another as follows:

- A filter data provider examines network content and decides whether it should be blocked or allowed.
    - Runs in highly restrictive sandbox.
    - Sandbox prevents user network content from escaping.
- A filter control provider provides needed configuration information to the filter data provider.
    - Has access to network, but not user network content.
- Requires Network Extensions entitlement.
- Key: `com.apple.developer.networking.networkextension`

Copyright © 2020, About Objects

Section 6: The Network Framework

Copyright © 2020, About Objects

Overview

- Provides direct access to protocols like TLS, TCP, and UDP.
- Underpins the URL Loading System (URLSession, etc.)

WWDC 2019 Videos: Advances in Networking, Parts 1 and 2

https://developer.apple.com/videos/play/wwdc2019/712

https://developer.apple.com/videos/play/wwdc2019/713

Copyright © 2020, About Objects

Section 7: Web Services

Copyright © 2020, About Objects

The URL Loading System

- Provides on-disk and in-memory cache on a per-application basis.
    - Individual URL requests can specify their desired cache policy.
    - Can also control caching policy by implementing a delegate callback.
- Supports authentication and credentials, including configurable credential persistence in memory while the app is running, or for greater durations via the user's keychain.
- Provides object-oriented access to cookies.
- Built-in support for **http**, **https**, **file**, and **ftp** protocols, as well as custom protocols.

Apple's *URL Loading System Programming Guide* is an excellent resource.

Copyright © 2020, About Objects

URL Connections

- **NSURLConnection** has three related delegate protocols:
  - **NSURLConnectionDelegate** – asynchronous callbacks sent to the delegate during authentication challenges, as well as to notify the delegate of connection failure.
  - **NSURLConnectionDataDelegate** – asynchronous callbacks sent to the delegate as the connection is downloading data.
  - **NSURLConnectionDownloadDelegate** – asynchronous sent to notify the delegate about download progress.
- Asynchronous loading:
  - Implement delegate callbacks; or
  - Use block-based API

```
+ (void)sendAsynchronousRequest:(NSURLRequest *) request
                          queue:(NSOperationQueue *) queue
              completionHandler:(void (^)(NSURLResponse *response,
                                          NSData *data,
                                          NSError *connectionError)) handler;
```

- Synchronous loading methods also available.
  - *Avoid using on the main thread*.

Copyright © 2020, About Objects

URL Sessions

- **NSURLSession** creates and manages instances of **NSURLSessionTask**.
  - Session task manages an **NSURLConnection**.
- Session tasks are created in suspended state.
  - Send a **resume** message to execute.
- Subclasses of **NSURLSessionTask**:

  NSURLSessionDataTask — Loads URL resource in memory as instance of **NSData**.

  NSURLSessionUploadTask — Initialized with a file, data object, or stream to upload.

  NSURLSessionDownloadTask — Downloads response data directly to a file. Notifies its delegate when download complete.

Copyright © 2020, About Objects

Working with URL Sessions

- `NSURLSession` provides a global session via its `+sharedSession` method.
- You can also create your own sessions.
- Custom sessions can be configured as background sessions.
  - Managed by macOS.
  - Can continue download when app is not running.
  - macOS will relaunch app when download completes.

Copyright © 2020, About Objects

URL Protocols (Optional)

Copyright © 2020, About Objects

Reachability

- Add input sources to main run loop to receive notifications about changes to reachability of network endpoints.
- File SCNetworkReachability.h in SystemConfiguration framework declares C functions for:
  - Configuring callbacks
  - Scheduling and unscheduling in run loop
- Callback structure includes flags with detailed network status.

Copyright © 2020, About Objects

Testing Tips

- Apple provides **Network Link Conditioner**
  - System Preferences plugin
  - Simulates various network conditions, including performance degradation.
- **Wireshark** and similar tools provide rich network monitoring.
- Consider providing a way to use dummy data in development when web services are unavailable.

Copyright © 2020, About Objects