Networking

NSURLConnection

NSURLConnection is the builtin class for making HTTP requests.

- The old fashioned way was to use the NSURLConnectionDelegate
- Since blocks and Grand Central Dispatch (GCD) were introduced, it's more common to use NSOperationQueue to make requests.

NSURLConnection

Make asynchronous requests like this...

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];
NSURLRequest *request = [NSURLRequest requestWithURL:url];
NSOperationQueue *queue = [[NSOperationQueue alloc] init];

[NSURLConnection sendAsynchronousRequest:request
   queue:queue
   completionHandler:^(NSURLResponse *response, NSData *data,
NSError *error) {
   // Do stuff with the response
}];
```

NSURLRequest

Form the HTTP request using NSURLRequest to do things like:

- Set the HTTP method
- Add HTTP headers
- Set HTTP body for POST and PUT methods
- Set caching policy

Caching Policies

NSURLRequest has several available caching policies:

- UseProtocolCachePolicy
- ReloadIgnoringLocalCacheData
- ReturnCacheDataElseLoad
- ReturnCacheDataDontLoad

Configuring the cache

All NSURLRequests use the NSURLCache, a memory/disk cache which must be initialized.

```
- (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
  NSURLCache *URLCache = [[NSURLCache alloc]
    initWithMemoryCapacity:4 * 1024 * 1024
    diskCapacity:20 * 1024 * 1024
    diskPath:nil];
  [NSURLCache setSharedURLCache:URLCache];
}
```

Reachability

Reachability was written by an Apple developer, but not packaged as part of Foundation. It allows you to:

- Determine the current state of wifi, cellular network, and internet accessibility.
- Register for changes in state of wifi, cellular network, and internet accessibility.

AFNetworking

AFNetworking is an open source networking library created by Matt Thompson, now lead mobile developer of Heroku.

Why AFNetworking?

- Inherits from NSOperation, allowing requests to be cancelled, suspended/resumed, and managed by an NSOperationQueue.
- Allows for easy streaming uploads and downloads, handling authentication challenges, and control caching behavior and request.
- Distinguishes between success and failure based on HTTP status codes.
- Pluggable serialization for formats such as JSON, XML, images, and property lists.
- Many more reasons...

AFNetworking

Creating an AFNetworking request is similar to using NSURLConnection.

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];
NSURLRequest *request = [NSURLRequest requestWithURL:url];
AFJSONRequestOperation *operation = [AFJSONRequestOperation
JSONRequestOperationWithRequest:request success:^(NSURLRequest
*request, NSHTTPURLResponse *response, id JSON) {
    // Do stuff with the response
} failure:nil];
[operation start];
```

AFHTTPClient

AFHTTPClient is a class that streamlines working with an API. Features include:

- Default headers
- Authentication
- Network reachability monitoring
- Query string serialization
- Batched operations
- Multipart form requests

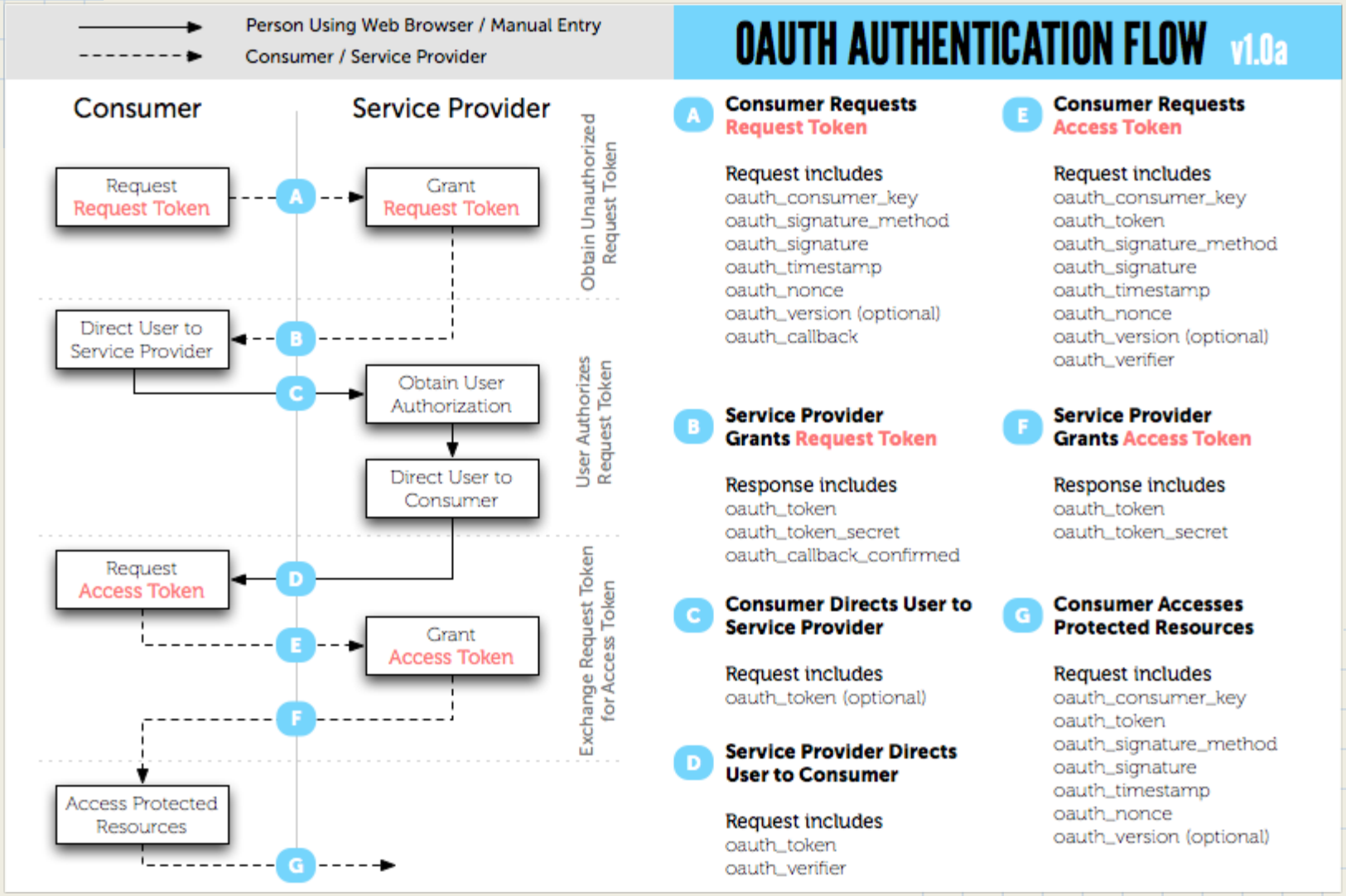
Authentication

Authentication

Interacting with APIs typically involves some type of authentication. Common techniques include:

- OAuth 1.0a (Yelp, Twitter, Tumblr)
- OAuth 2.0 (Facebook, Instagram, Google)
- Builtin iOS authentication for Facebook and Twitter

OAuth 1.0a


Person Using Web Browser / Manual Entry
Consumer / Service Provider
Consumer
Service Provider
Request Request Token
A
Grant Request Token
Obtain Unauthorized Request Token
Direct User to Service Provider
B
C
Obtain User Authorization
Direct User to Consumer
User Authorizes Request Token
Request Access Token
D
E
Grant Access Token
F
Exchange Request Token for Access Token
Access Protected Resources
G
OAUTH AUTHENTICATION FLOW v1.0a
A Consumer Requests Request Token
Request includes
oauth_consumer_key
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)
oauth_callback
B Service Provider Grants Request Token
Response includes
oauth_token
oauth_token_secret
oauth_callback_confirmed
C Consumer Directs User to Service Provider
Request includes
oauth_token (optional)
D Service Provider Directs User to Consumer
Request includes
oauth_token
oauth_verifier
E Consumer Requests Access Token
Request includes
oauth_consumer_key
oauth_token
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)
oauth_verifier
F Service Provider Grants Access Token
Response includes
oauth_token
oauth_token_secret
G Consumer Accesses Protected Resources
Request includes
oauth_consumer_key
oauth_token
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)

OAuth 1.0a

OAuth 1.0a is kind of a PITA. Common issues include:

- Out of sync clock will cause signature mismatch
- Inconsistent implementation across providers
- Multipart uploads (like photo uploading)

OAuth 1.0a

OAuth 1.0a is so complex because it provides secure authentication over an unsecure transport.

OAuth 2.0

Most APIs are moving to OAuth 2.0 which sidesteps OAuth 1.0a complexity by requiring https

- Requires https
- Initial authentication grants access token
- Each subsequent request includes access token

OAuth 2.0

If you have the user's password, why bother with the access token?

Social.framework

iOS 6 includes the Social Framework which integrates with social networking services, e.g., Facebook and Twitter

- Handles SSO authentication
- Gets activity feed
- Make a new post
- Wraps additional APIs

Persistence

Persistence

There are a variety of persistence options on iOS:

- NSUserDefaults
- File system
- SQLite
- Core Data

Object serialization

Similar to Java serializable, Foundation has the NSCoding protocol which requires the methods:

- - (id)initWithCoder:(NSCoder *)decoder;
- - (void)encodeWithCoder:(NSCoder *)encoder;

Object serialization

Alternatively, JSON is a convenient protocol for object serialization and deserialization.

NSUserDefaults

NSUserDefaults is defined for user preferences, but is commonly used for the simple persistence of a small number of objects, for example:

- Simple application state
- Currently logged in user

NSUserDefaults

NSUserDefaults can store objects of type:

- NSData
- NSString
- NSNumber
- NSDate
- NSArray
- NSDictionary

NSUserDefaults

For example,

```
NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
[defaults setObject:firstName forKey:@"firstName"];
[defaults synchronize];
```

File I/O

Each application has access to its own area of the file system, with default folders for Documents and Cache.

File I/O

Access the Documents directory, as below:

```
NSString *rootPath =
   [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
    NSUserDomainMask, YES) objectAtIndex:0];

NSString *filePath = [rootPath
   stringByAppendingPathComponent:@"myfile.plist"];
```

File I/O

Many classes have builtin file helpers, such as NSArray

```
NSArray *items = @[@”Hello”, @”How”, @”Are”, @”You”];
[items writeToFile:filePath atomically:YES];
```

SQLite

iOS includes the popular SQLite library, which is the industry de facto standard for lightweight embedded SQLite programming.

SQLite

Working with SQLite typically involves the following tasks:

- Creating a database
- Adding/Modifing/Deleting tables
- Queries
- Inserts/Updates/Deletes

Why SQLite?

- SQLite is simple to use and is a great solution for situations that requires a high performance database.
- If you’re using SQLite, consider using the popular SQLite wrapper, FMDB.

Why not SQLite?

If you're using SQLite for object management, you should consider using a library with a higher level of abstraction, such as Core Data or other ORM/DAOs.

Models

Models on mobile usually reflect the server state for **data retrieved via APIs** including:

- Model definitions (properties, relationships)
- JSON/XML serialization/deserialization
- Persistence Storage (database)
- Data Formatters

Core Data

Core Data is an incredibly powerful framework for object management and persistence.

- Entity modeling, with relationships
- Advanced querying
- Multiple serialization options
- Automatic migrations
- Automatic undo manager

But...

Core Data is a gigantic pain to use and most of its features are unused in a REST client.

If you insist on Core Data...

Consider using one of the many libraries related to Core Data

- Wrappers - Magical Record, Objective-Record, SSDataKit, ios-queryable, ReactiveCoreData
- Adapters - RestKit, AFIncrementalStore, MMRecord, SLRESTfulCoreData, Overcoat, Mantle
- Utilities - mogenerator