What's New in Foundation Networking

A session on NSURLSession

Session 705

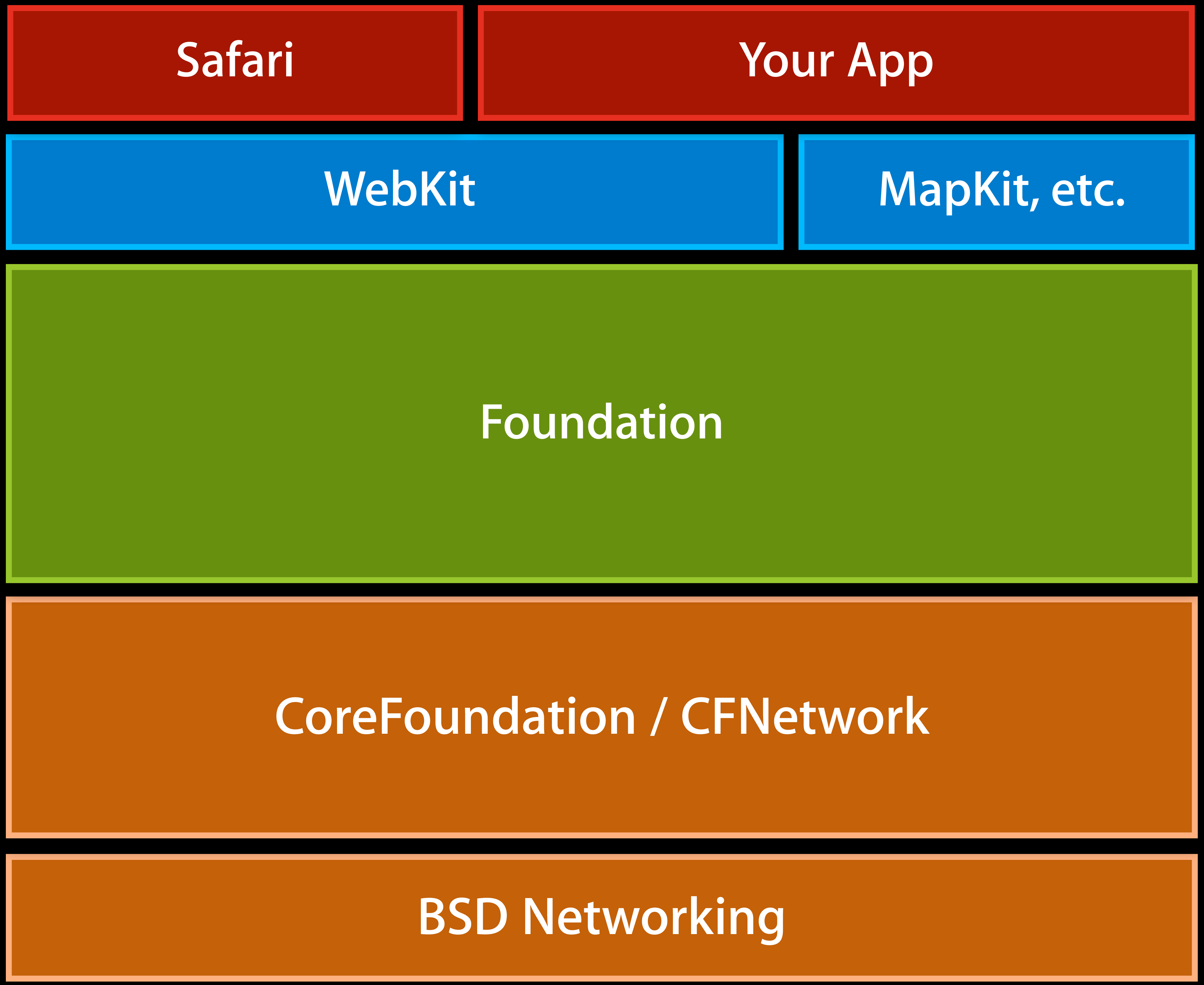
Steve Algernon
Senior Wrangler

What’s New in Foundation Networking

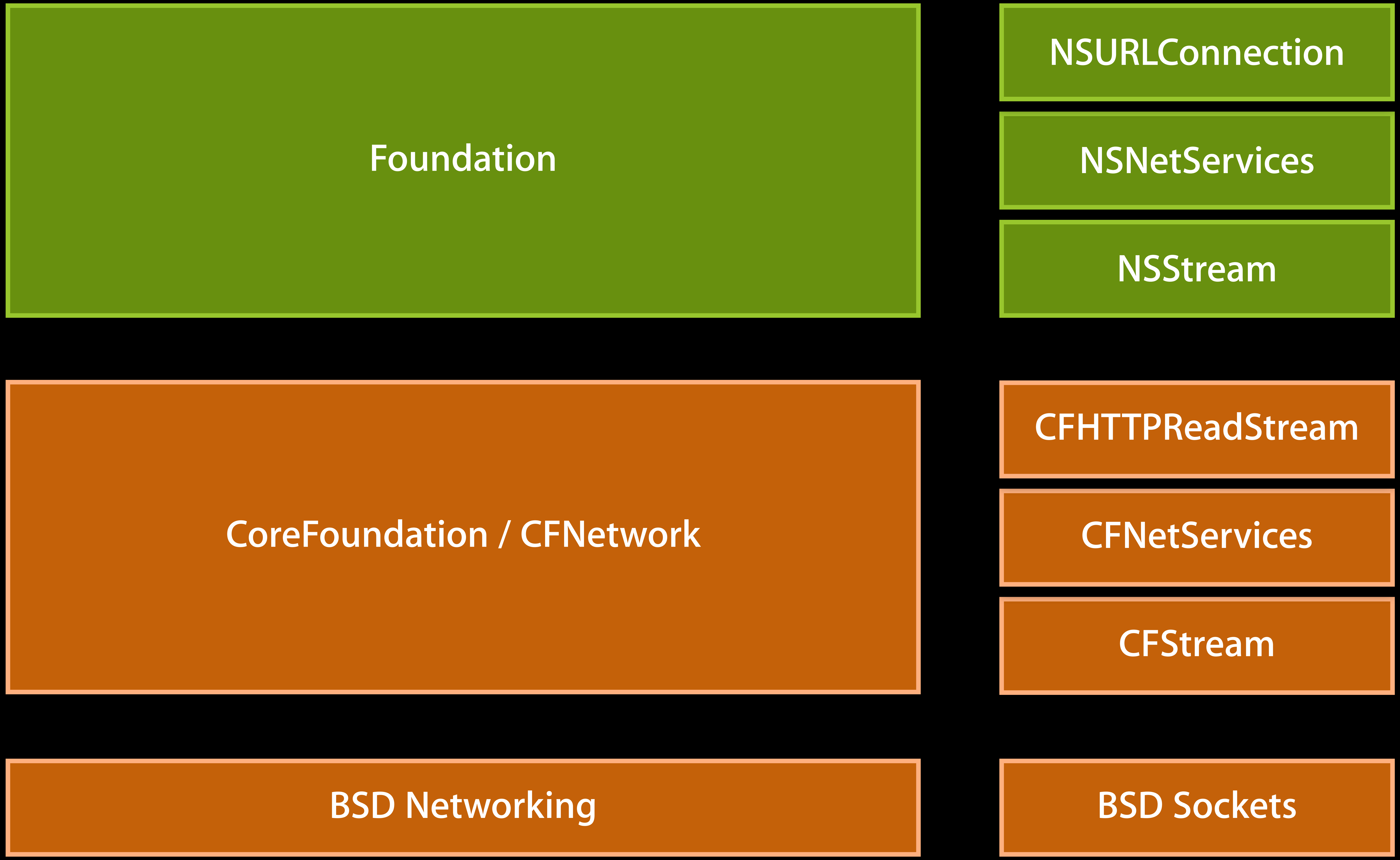
NEW

- New NSURLSession API
  - iOS 7, OS X 10.9
  - Out-of-process background transfers
- Framework Enhancements
  - NSNetServices
  - Single sign-on
  - iCloud credential syncing

Foundation Networking



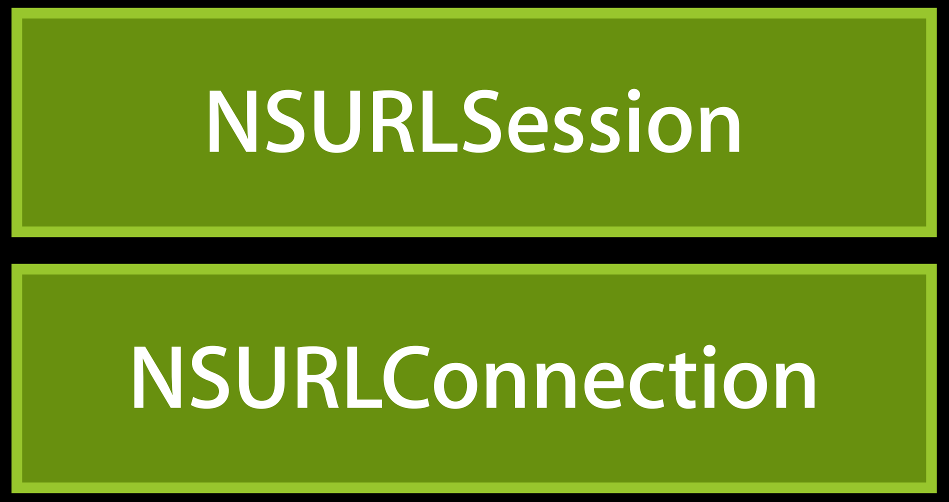
Foundation Networking



Foundation Networking

NSURLConnection
NSNetServices

Foundation Networking



NSURLConnection

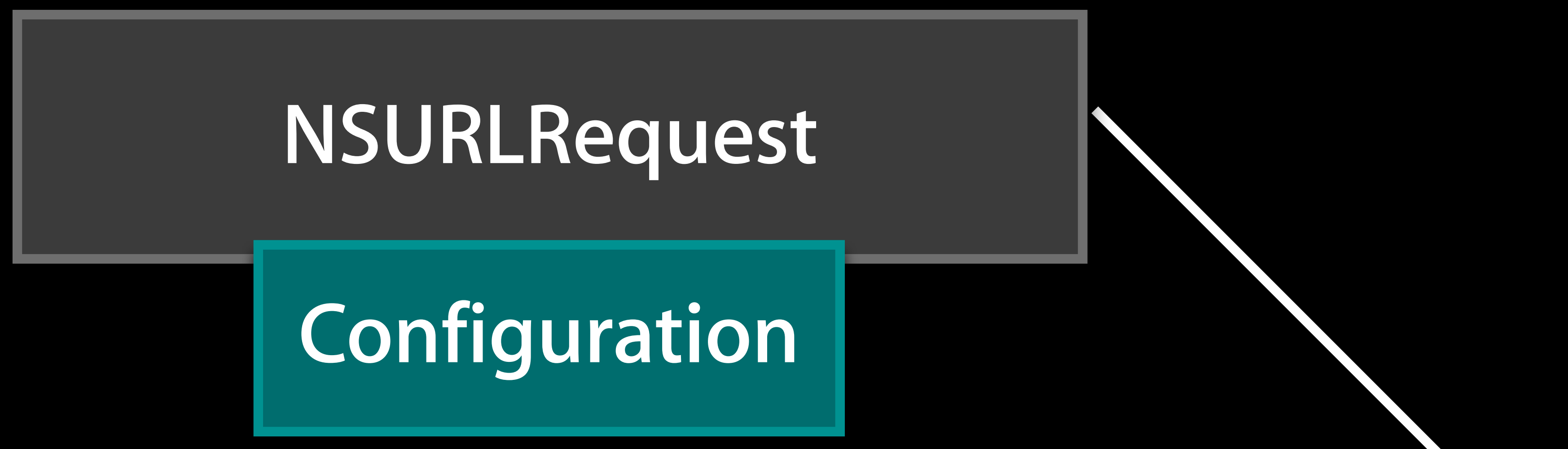
- Both a technology and a specific class
- Originally written for Safari, available via Foundation
    - WWDC 2003 - Session 418
- URL resolution and loading
    - file:// http:// https:// data://
    - Extensible via NSURLProtocol
- URL Loading machinery, policies
    - Configured via NSURLRequest properties
    - Shared Persistent Storage: Cache, Credentials, Cookies
- Authentication and Proxies

NSURLConnection

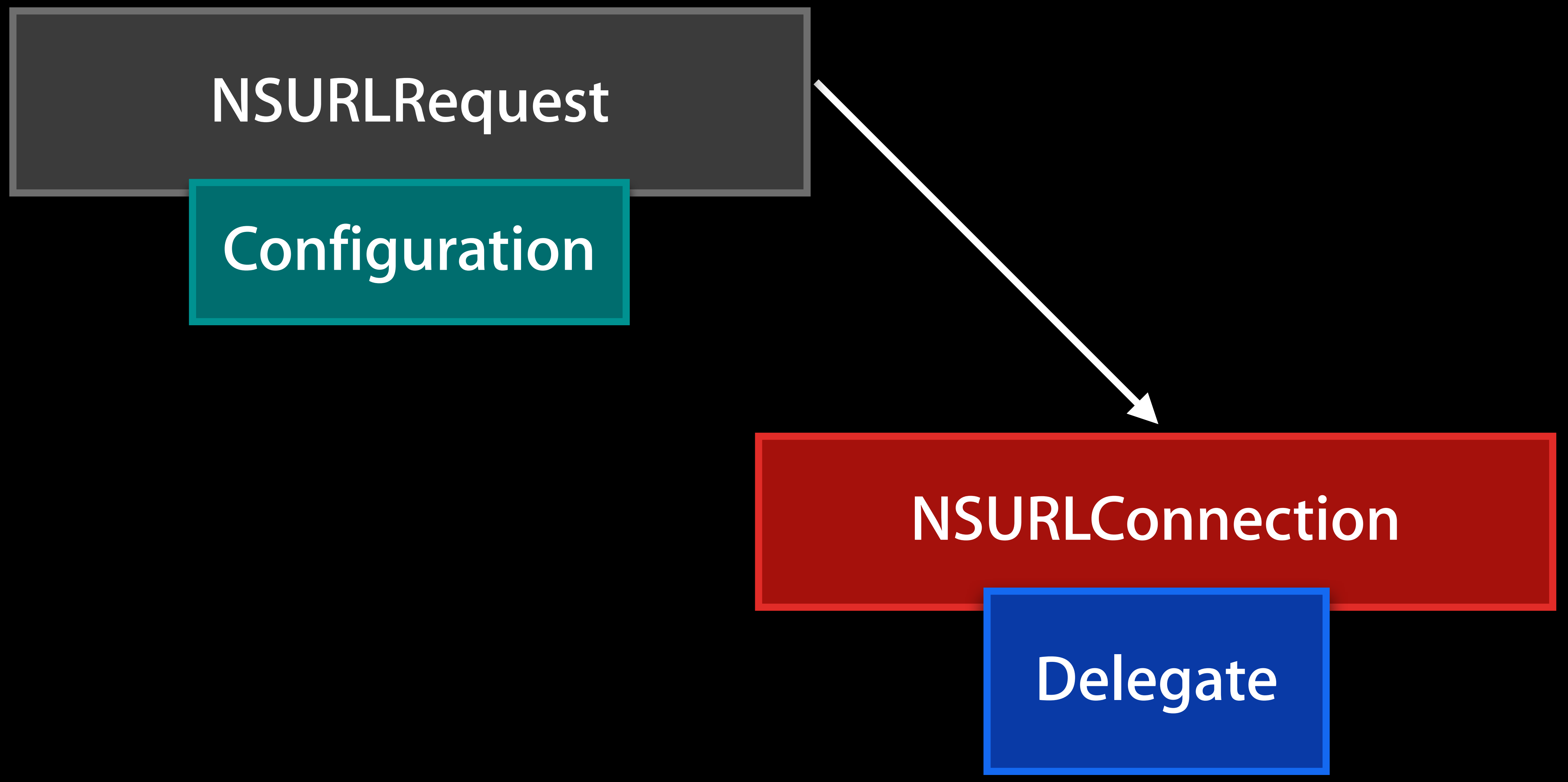
NSURLConnection

NSURLRequest

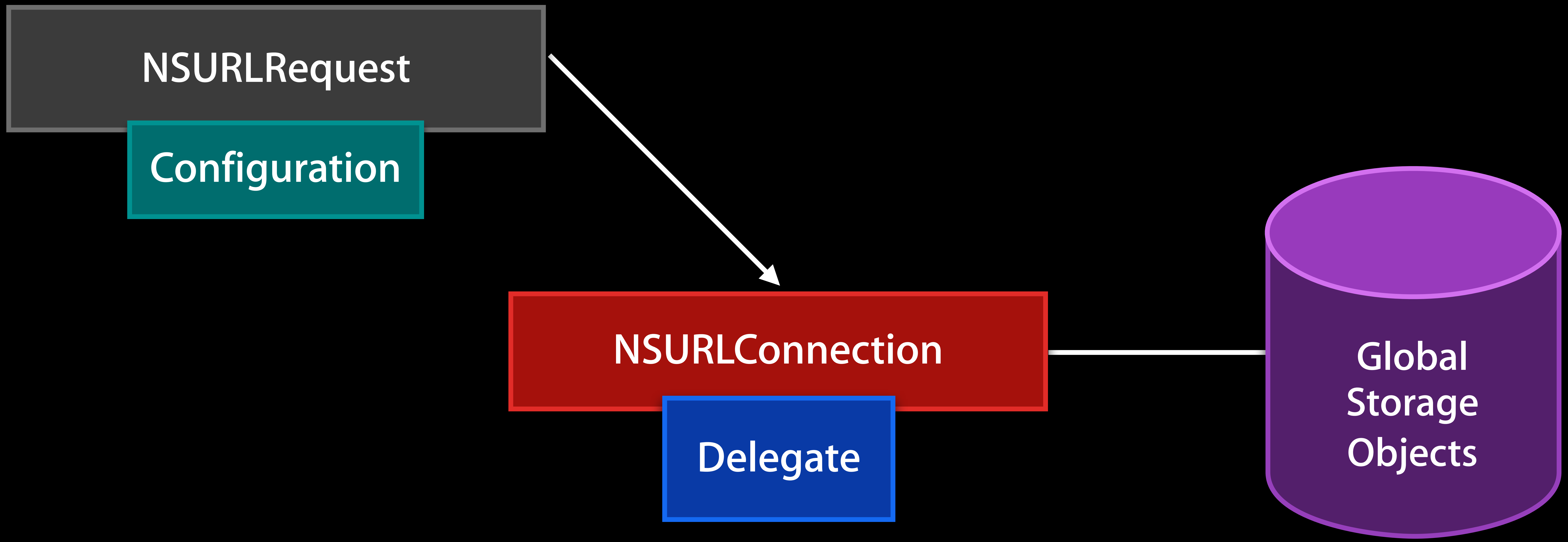
NSURLConnection



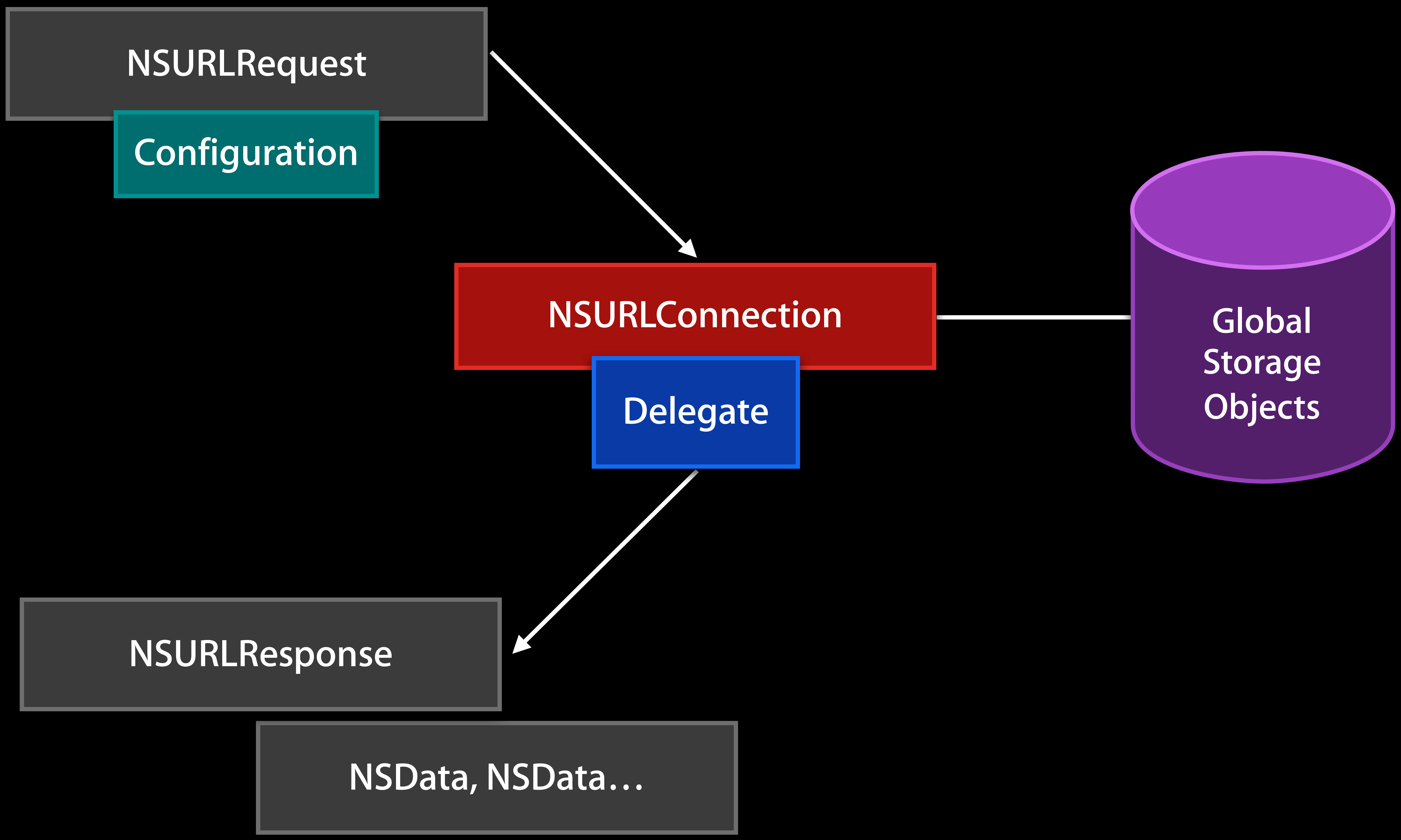
NSURLConnection



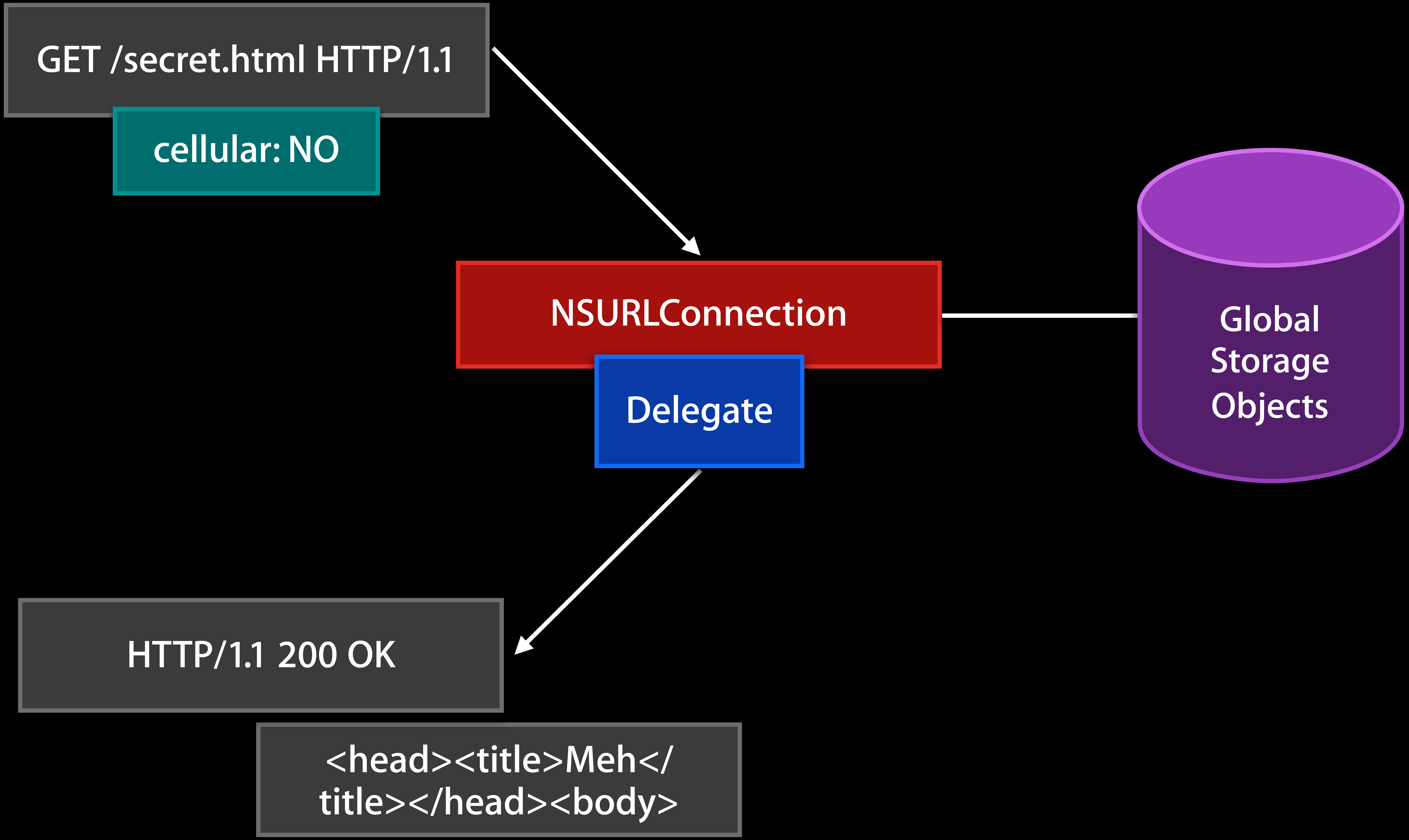
NSURLConnection



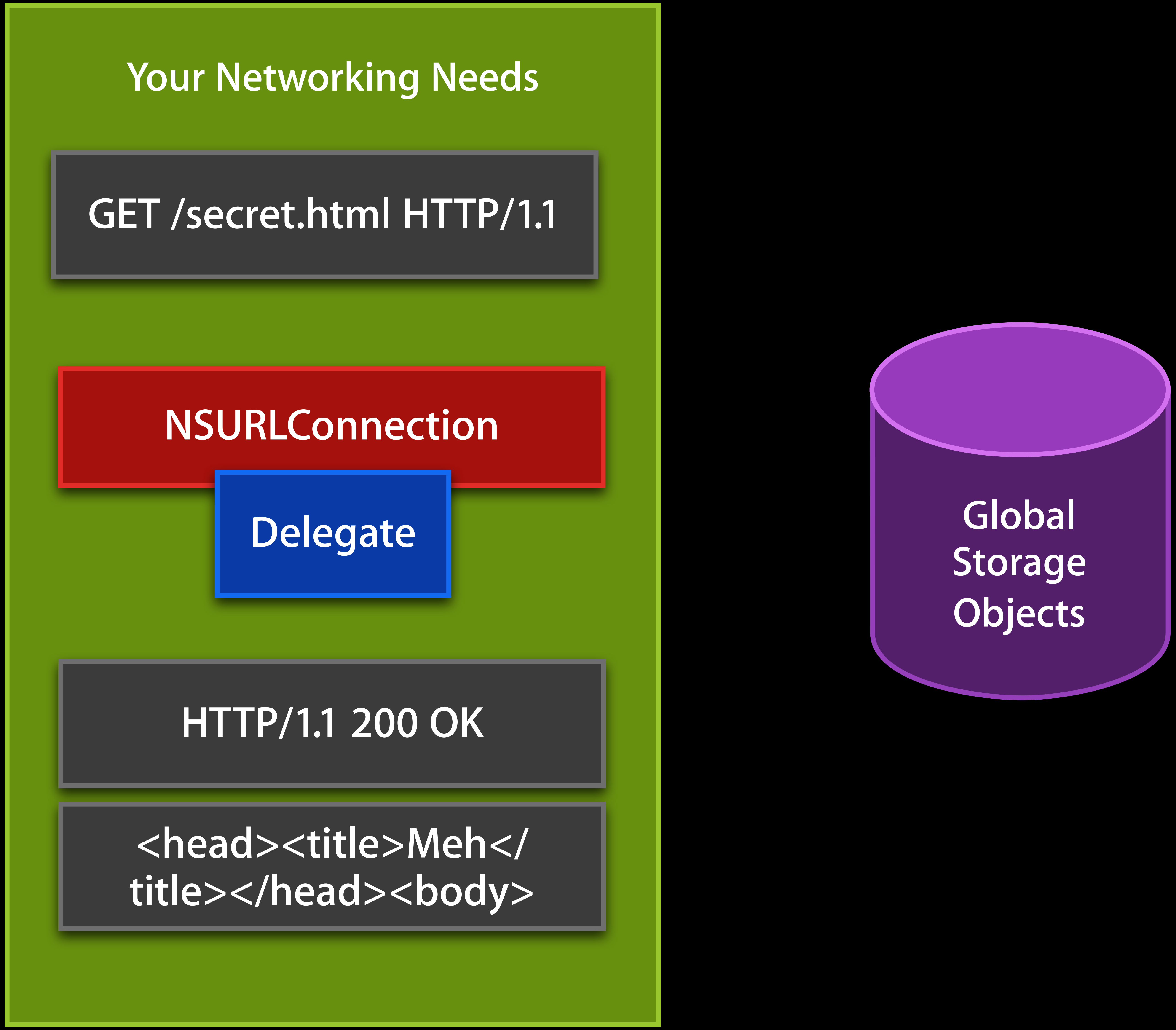
NSURLConnection



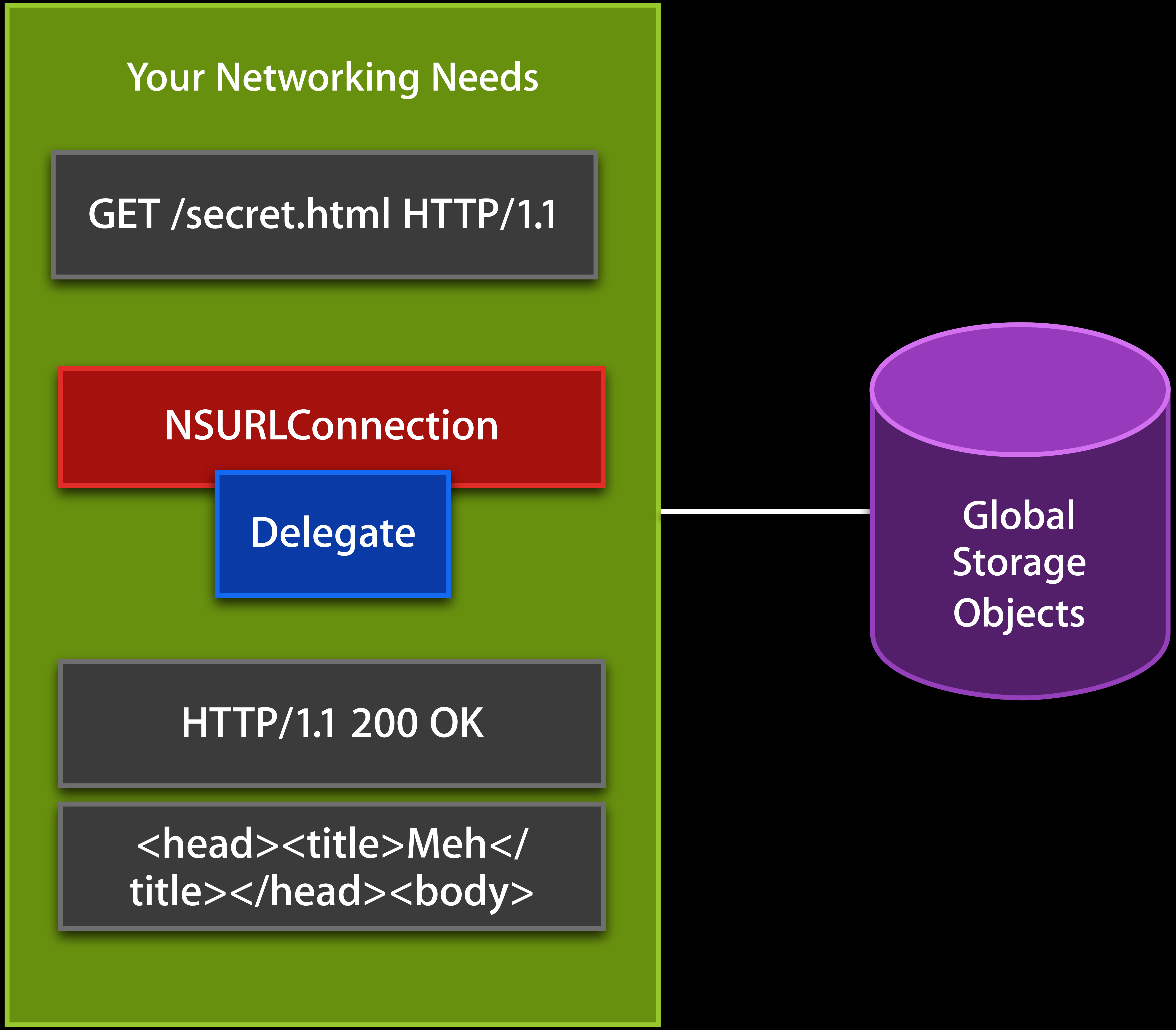
NSURLConnection



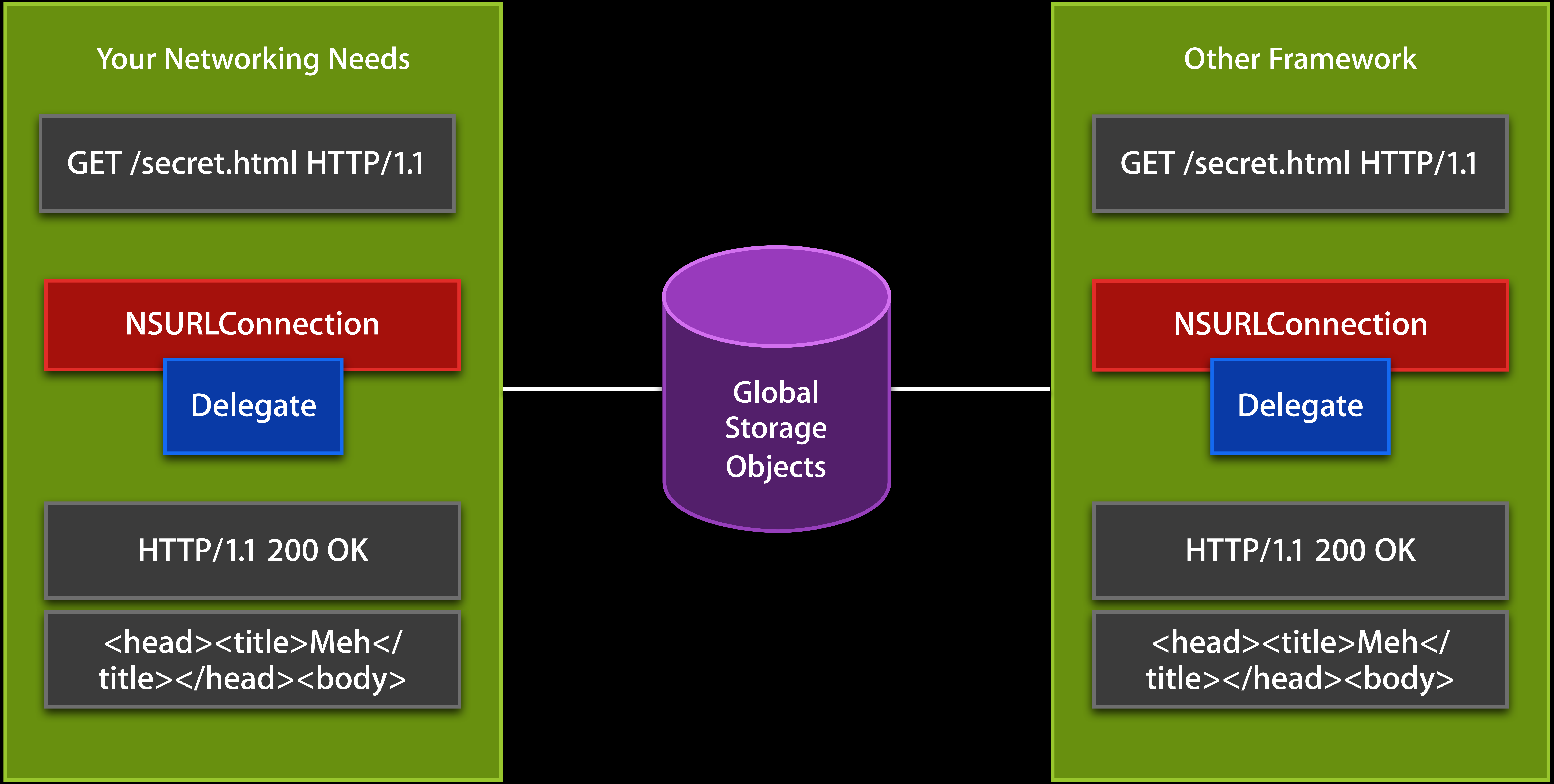
NSURLConnection



NSURLConnection



NSURLConnection



NSURLSession

- Both a technology and a specific class
- Replaces NSURLConnection
    - Preserves existing concepts and objects
    - NSURLRequest, NSURLResponse, etc.
- Configurable Container
    - HTTP options
    - Subclassable and private storage
- Improved authentication handling
    - Connection vs. Request authentication
- Rich delegate model

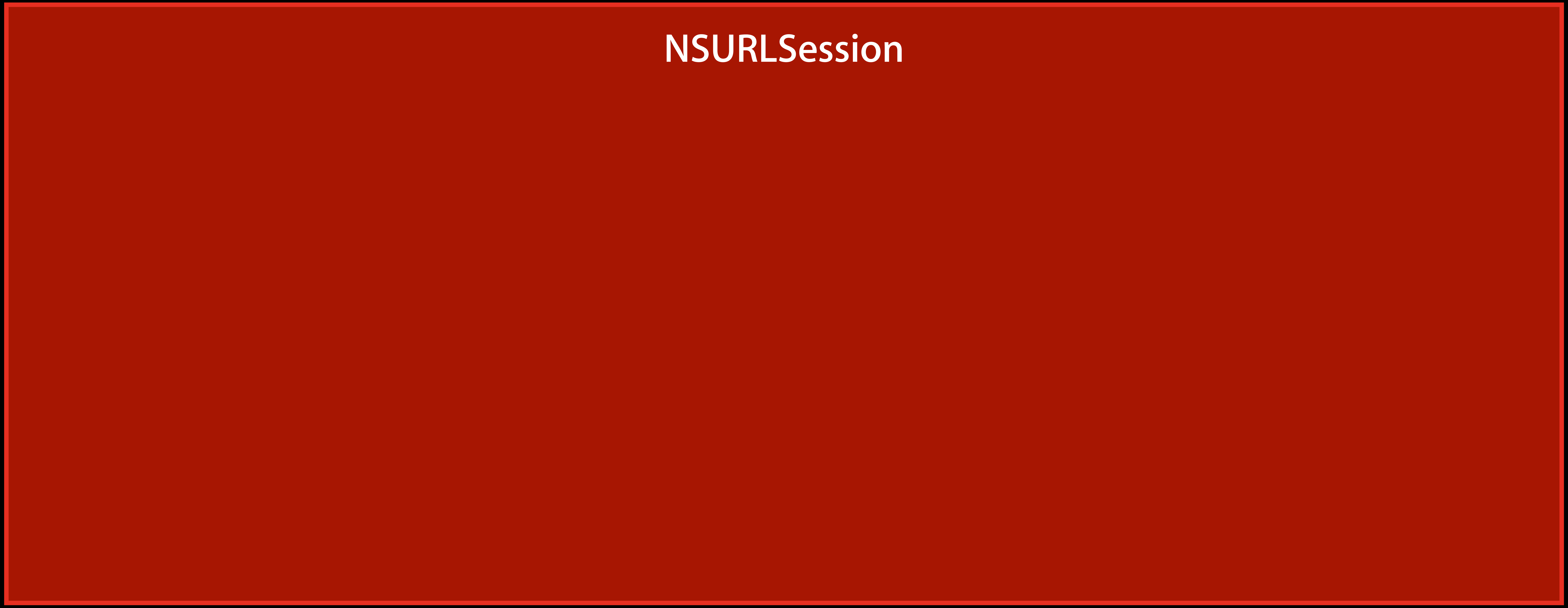
NSURLSession

NEW

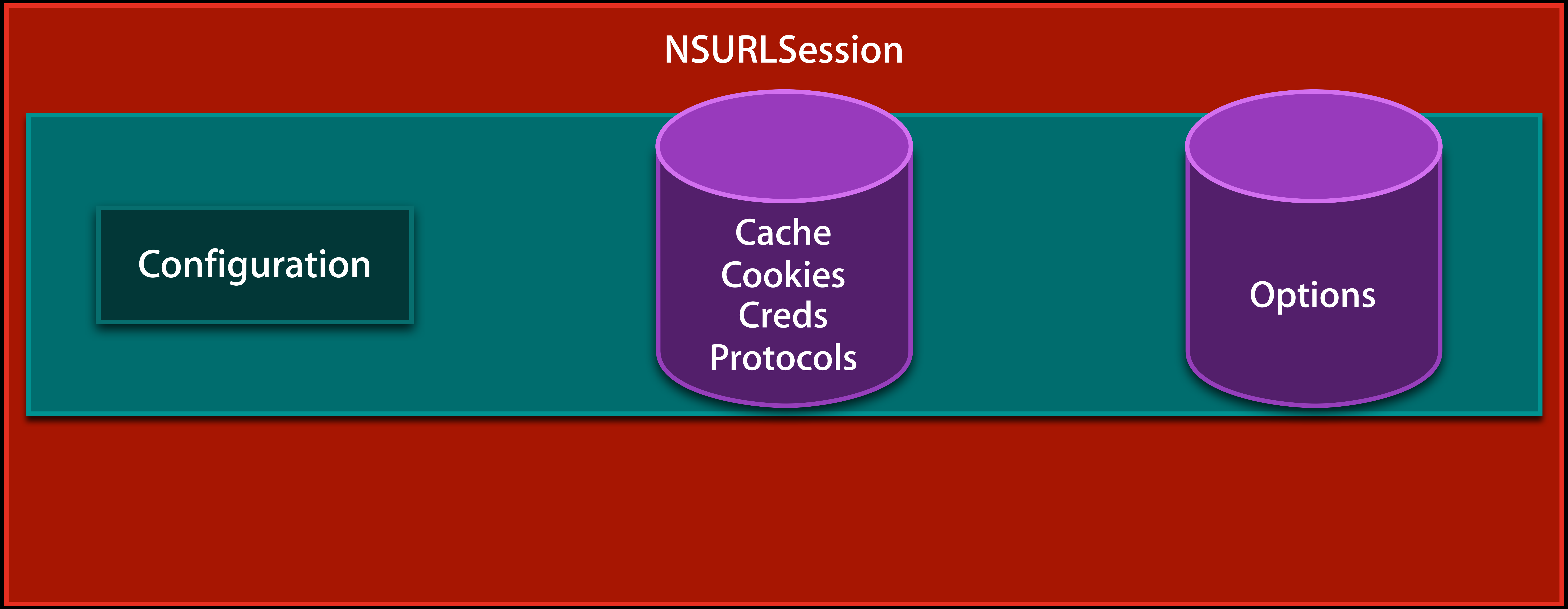
- Uploads/Downloads through the filesystem
- Encourages separation of Data from Meta-Data
  - NSURLRequst + payload
  - NSURLResponse + payload
- Out-of-process Uploads and Downloads
  - Uses same delegate model as in-process transfers
  - Optimizes battery life
  - Supports UIKit multitasking

NSURLSession

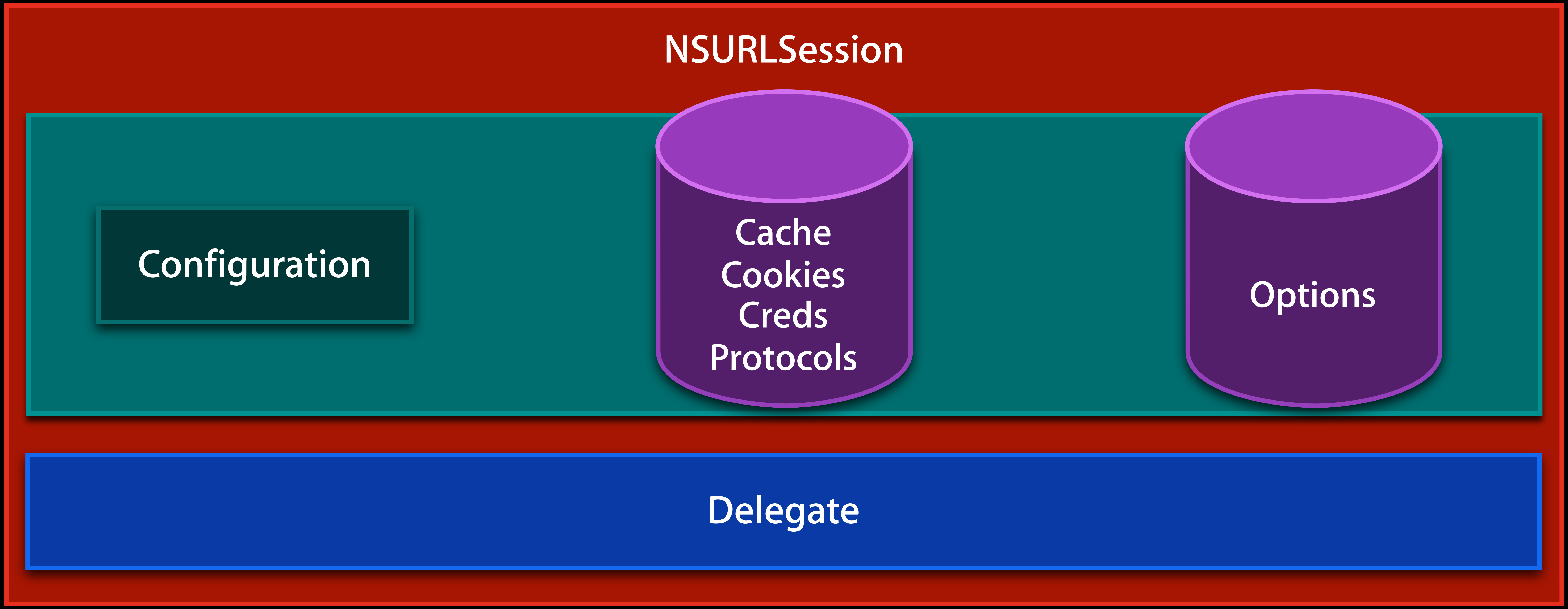
NSURLSession



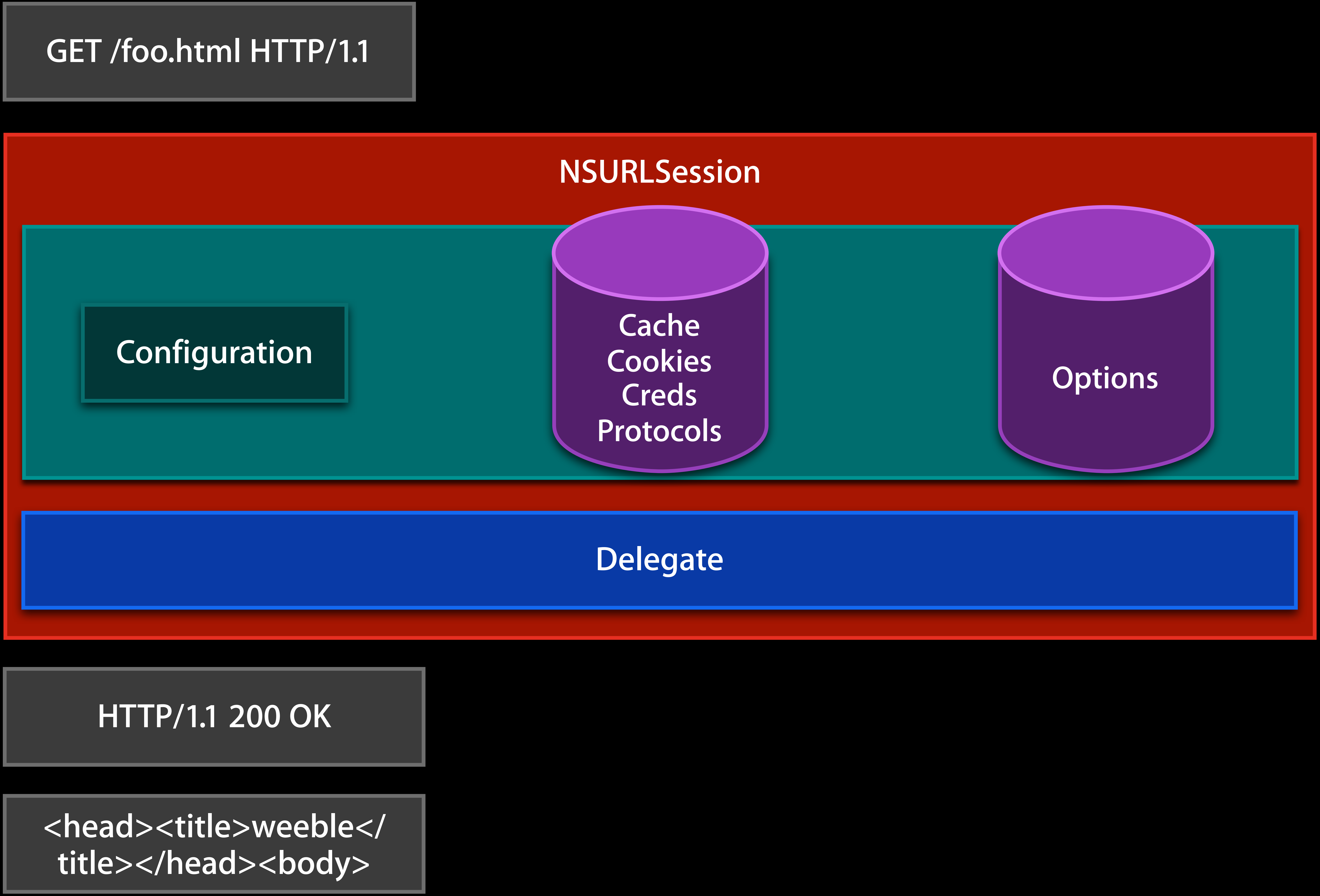
NSURLSession



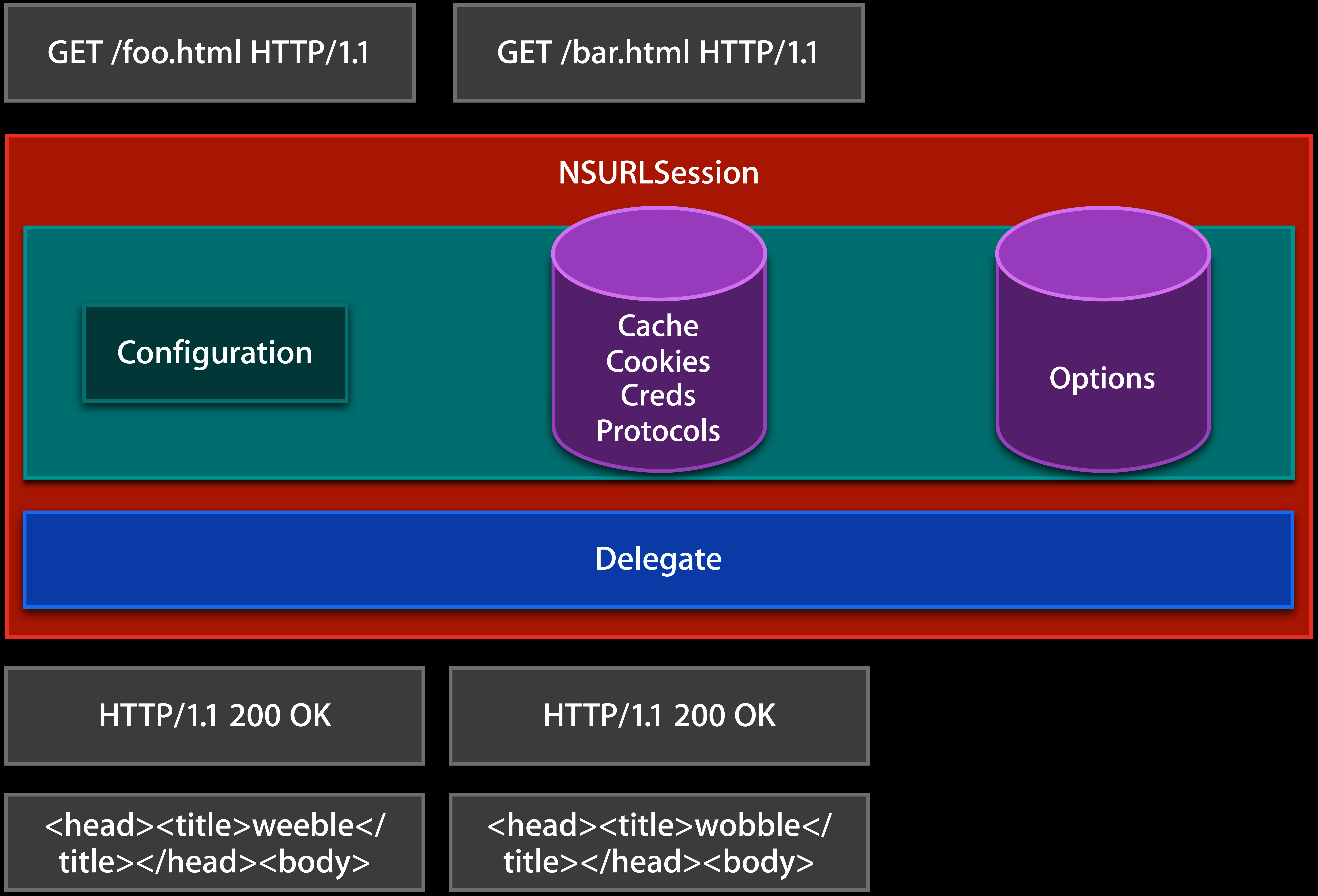
NSURLSession



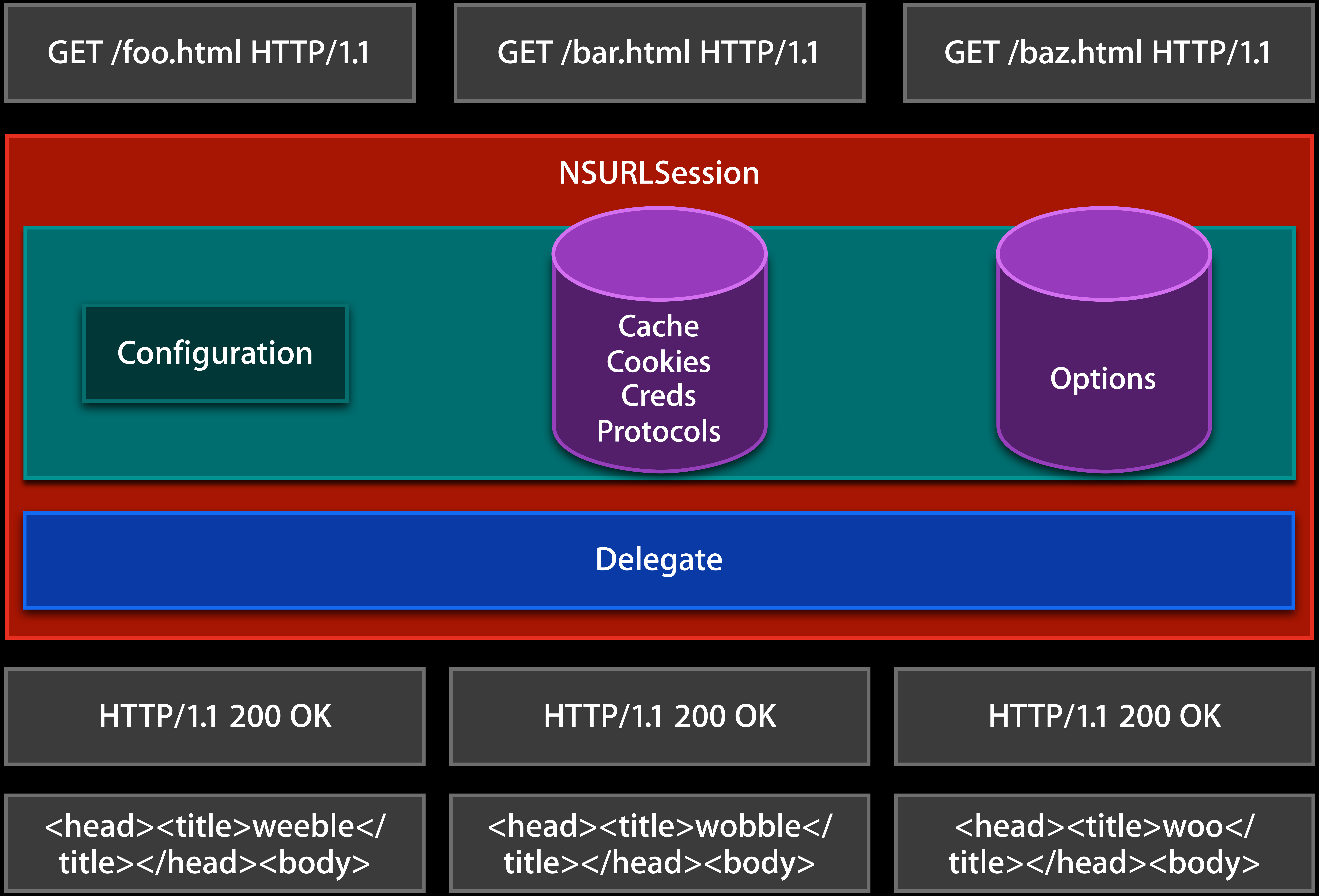
NSURLSession



NSURLSession



NSURLSession



NSURLSession API

- NSURLSessionConfiguration
  - Connection/HTTP policies
  - Cache, Credentials, Cookie storage
- NSURLSessionTask
  - Unit of “work” for a session
- NSURLSessionDelegate
- NSURLSession
  - Created with configuration, optional delegate
  - Creates one NSURLSessionTask per request
  - Long lived object

NSURLSession—Adoption

```
// NSURLConnection example:
id<NSURLConnectionDelegate> myDelegate = [[MyDelegate alloc] init];

for (int i = 0;  i < 10;  i++) {
    NSURL* myURL = [NSURL URLWithString:@”http://www.apple.com/”];

    NSURLRequest* myRequest = [NSURLRequest requestWithURL:myURL];
    [myRequest setAllowsCellularAccess:NO];

    NSURLConnection* conn;
    conn = [NSURLConnection connectionWithRequest:myRequest
                                         delegate:myDelegate];

}
```

NSURLSession–Adoption

```
// NSURLConnection example:
id<NSURLConnectionDelegate> myDelegate = [[MyDelegate alloc] init];

for (int i = 0;  i < 10;  i++) {
     NSURL* myURL = [NSURL URLWithString:@”http://www.apple.com/”];

     NSURLRequest* myRequest = [NSURLRequest requestWithURL:myURL];
     [myRequest setAllowsCellularAccess:NO];

     NSURLConnection* conn;
     conn = [NSURLConnection connectionWithRequest:myRequest
                                          delegate:myDelegate];

}
```

NSURLSession–Adoption

```
// NSURLConnection example:
id<NSURLConnectionDelegate> myDelegate = [[MyDelegate alloc] init];

for (int i = 0;  i < 10;  i++) {
     NSURL* myURL = [NSURL URLWithString:@"http://www.apple.com/"];

     NSURLRequest* myRequest = [NSURLRequest requestWithURL:myURL];
     [myRequest setAllowsCellularAccess:NO];

     NSURLConnection* conn;
     conn = [NSURLConnection connectionWithRequest:myRequest
                                          delegate:myDelegate];

}
```

NSURLSession–Adoption

```
// NSURLConnection example:
id<NSURLConnectionDelegate> myDelegate = [[MyDelegate alloc] init];

for (int i = 0;  i < 10;  i++) {
     NSURL* myURL = [NSURL URLWithString:@"http://www.apple.com/"];

     NSURLRequest* myRequest = [NSURLRequest requestWithURL:myURL];
     [myRequest setAllowsCellularAccess:NO];

     NSURLConnection* conn;
     conn = [NSURLConnection connectionWithRequest:myRequest
                                          delegate:myDelegate];

}
```

NSURLSession—Adoption

```
// NSURLConnection example:
id<NSURLConnectionDelegate> myDelegate = [[MyDelegate alloc] init];

for (int i = 0;  i < 10;  i++) {
     NSURL* myURL = [NSURL URLWithString:@”http://www.apple.com/”];

     NSURLRequest* myRequest = [NSURLRequest requestWithURL:myURL];
     [myRequest setAllowsCellularAccess:NO];

     NSURLConnection* conn;
     conn = [NSURLConnection connectionWithRequest:myRequest
                                          delegate:myDelegate];

}
```

NSURLSession—Adoption

```
// NSURLSession example:
id<NSURLSessionDelegate> myDelegate = [[MyDelegate alloc] init];

NSURLSessionConfiguration* myConfiguration =
                          [NSURLSession defaultSessionConfiguration];
myConfiguration.allowsCellularAccess = NO;

NSURLSession* mySession = [NSURLSession
                             sessionWithConfiguration:myConfiguration
                             delegate:myDelegate
                             delegateQueue:[NSOperationQueue mainQueue];

for (int i = 0;  i < 10;  i++) {
     NSURL* myURL = [NSURL URLWithString:@”http://www.apple.com/”];
     NSURLSessionDataTask* task;
     task = [mySession dataTaskWithHTTPGetRequest:myURL];
}
```

NSURLSession–Adoption

```
// NSURLSession example:
id<NSURLSessionDelegate> myDelegate = [[MyDelegate alloc] init];

NSURLSessionConfiguration* myConfiguration =
                          [NSURLSession defaultSessionConfiguration];
myConfiguration.allowsCellularAccess = NO;

NSURLSession* mySession = [NSURLSession
                             sessionWithConfiguration:myConfiguration
                             delegate:myDelegate
                             delegateQueue:[NSOperationQueue mainQueue];

for (int i = 0;  i < 10;  i++) {
     NSURL* myURL = [NSURL URLWithString:@”http://www.apple.com/”];
     NSURLSessionDataTask* task;
     task = [mySession dataTaskWithHTTPGetRequest:myURL];
}
```

NSURLSession–Adoption

```
// NSURLSession example:
id<NSURLSessionDelegate> myDelegate = [[MyDelegate alloc] init];

NSURLSessionConfiguration* myConfiguration =
                          [NSURLSession defaultSessionConfiguration];
myConfiguration.allowsCellularAccess = NO;

NSURLSession* mySession = [NSURLSession
                             sessionWithConfiguration:myConfiguration
                             delegate:myDelegate
                             delegateQueue:[NSOperationQueue mainQueue];

for (int i = 0;  i < 10;  i++) {
     NSURL* myURL = [NSURL URLWithString:@"http://www.apple.com/"];
     NSURLSessionDataTask* task;
     task = [mySession dataTaskWithHTTPGetRequest:myURL];
}
```

NSURLSession—Adoption

```
// NSURLSession example:
id<NSURLSessionDelegate> myDelegate = [[MyDelegate alloc] init];

NSURLSessionConfiguration* myConfiguration =
                          [NSURLSession defaultSessionConfiguration];
myConfiguration.allowsCellularAccess = NO;

NSURLSession* mySession = [NSURLSession
                             sessionWithConfiguration:myConfiguration
                             delegate:myDelegate
                             delegateQueue:[NSOperationQueue mainQueue]];

for (int i = 0;  i < 10;  i++) {
     NSURL* myURL = [NSURL URLWithString:@"http://www.apple.com/"];
     NSURLSessionDataTask* task;
     task = [mySession dataTaskWithHTTPGetRequest:myURL];
}
```

NSURLSession—Adoption

```
// NSURLSession example:
id<NSURLSessionDelegate> myDelegate = [[MyDelegate alloc] init];

NSURLSessionConfiguration* myConfiguration =
                          [NSURLSession defaultSessionConfiguration];
myConfiguration.allowsCellularAccess = NO;

NSURLSession* mySession = [NSURLSession
                             sessionWithConfiguration:myConfiguration
                             delegate:myDelegate
                             delegateQueue:[NSOperationQueue mainQueue];

for (int i = 0;  i < 10;  i++) {
     NSURL* myURL = [NSURL URLWithString:@”http://www.apple.com/”];
     NSURLSessionDataTask* task;
     task = [mySession dataTaskWithHTTPGetRequest:myURL];
}
```

NSURLSession—Adoption

```
// NSURLSession example:
id<NSURLSessionDelegate> myDelegate = [[MyDelegate alloc] init];

NSURLSessionConfiguration* myConfiguration =
                          [NSURLSession defaultSessionConfiguration];
myConfiguration.allowsCellularAccess = NO;

NSURLSession* mySession = [NSURLSession
                             sessionWithConfiguration:myConfiguration
                             delegate:myDelegate
                             delegateQueue:[NSOperationQueue mainQueue];

for (int i = 0;  i < 10;  i++) {
     NSURL* myURL = [NSURL URLWithString:@"http://www.apple.com/"];
     NSURLSessionDataTask* task;
     task = [mySession dataTaskWithHTTPGetRequest:myURL];
}
```

NSURLSessionConfiguration

- Per-session policies
  - Cache, Cookies, Credential stores
  - Cell usage, network service type
  - Number of connections
  - Resource and network timeouts
  - TLS protocols
  - HTTP proxies, cookies, pipelining, headers
  - Protocol handlers
- Storage subclasses
- Factory constructors for standard configurations

NSURLSessionConfiguration

- Default for access to global singleton storage, settings:

```
+(NSURLSessionConfiguration*) defaultConfiguration;
```

- Private storage, in-memory only storage:

```
+(NSURLSessionConfiguration*) ephemeralSessionConfiguration;
```

- Out-of-process session configuration, keyed to identifier string

```
+(NSURLSessionConfiguration*) backgroundSessionConfiguration:(NSString*)
                                                             identifier;
```

- Configuration objects are mutable, but copied when accessed

```
-(NSURLSessionConfiguration*) copy;
```

NSURLSessionTask

- Replaces NSURLConnection class
- Provides status and progress properties
- Cancel, Suspend, Resume
- Data and Upload tasks provided to differentiate
- Download task allows for capturing download state

```
–[NSURLSessionDownloadTask cancelByProducingResumeData:]
```

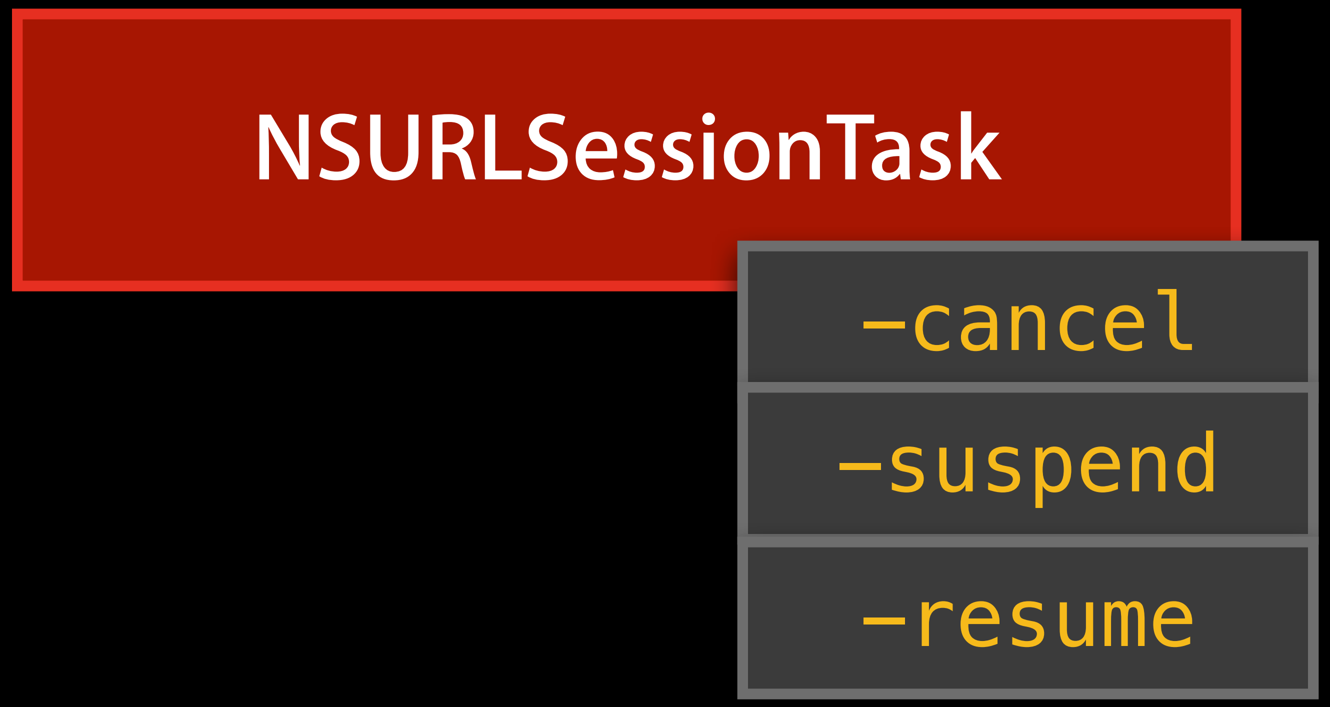
- NSURLSessionDelegate methods keyed to task type

```
–URLSession:task:didCompleteWithError:
–URLSession:dataTask:didReceiveData:
–URLSession:downloadTask:didFinishDownloadingToURL:
```

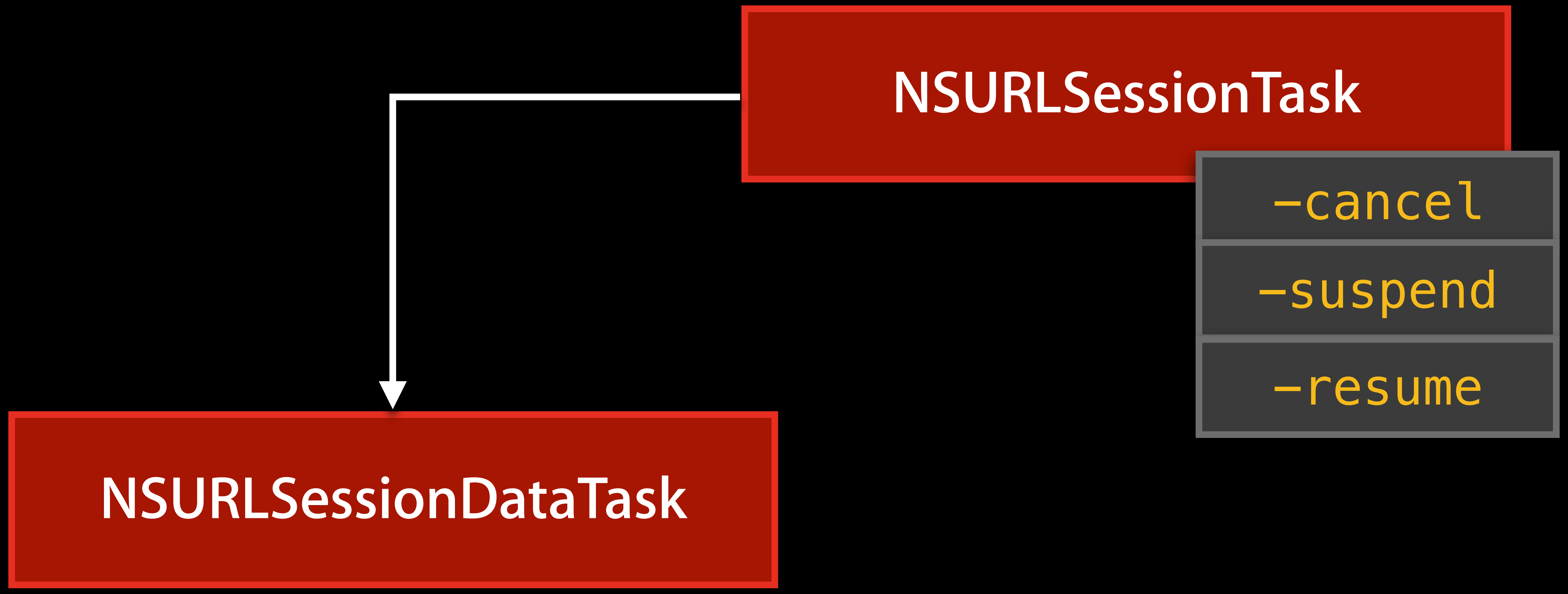
NSURLSessionTask

NSURLSessionTask

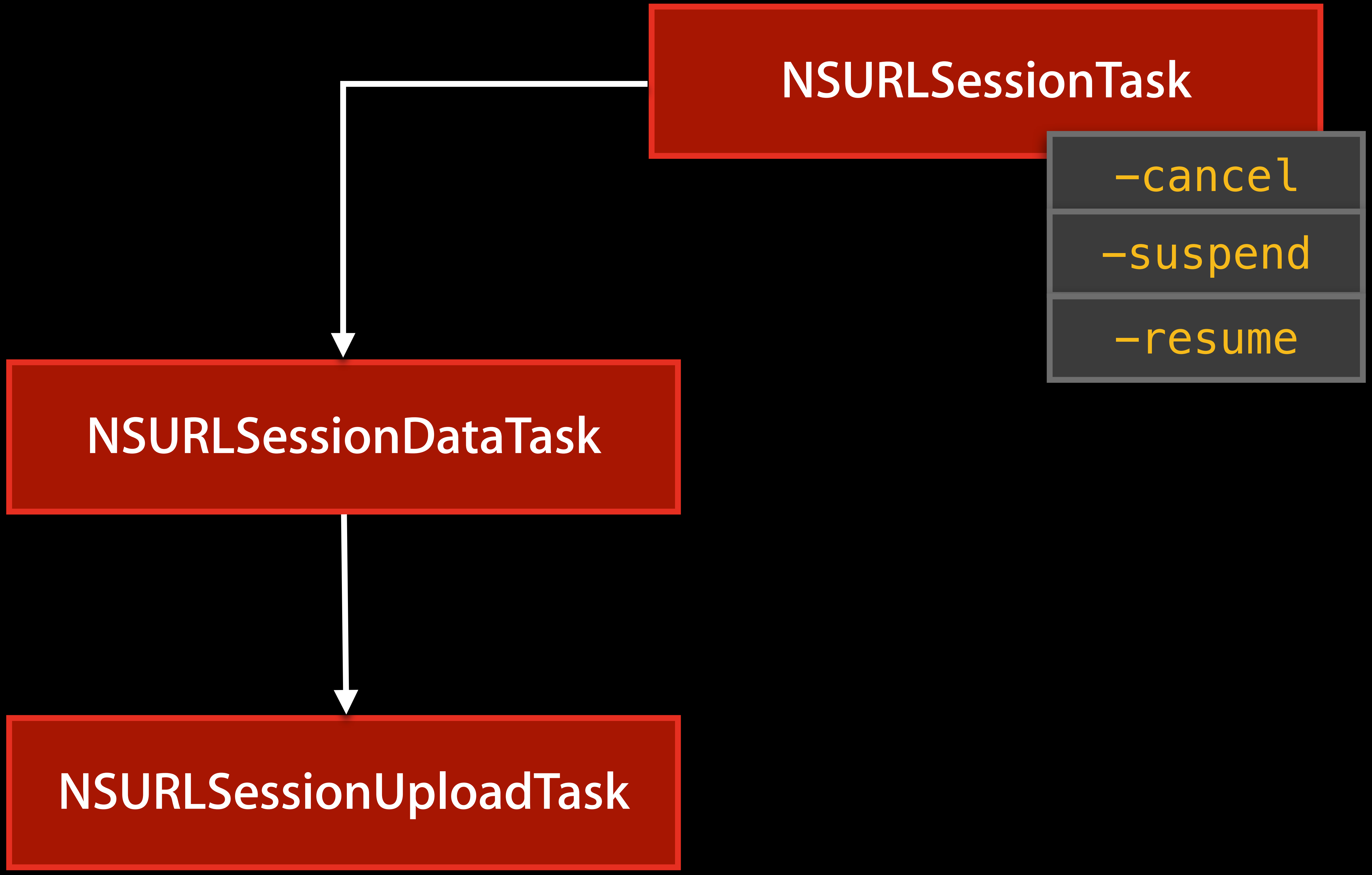
NSURLSessionTask



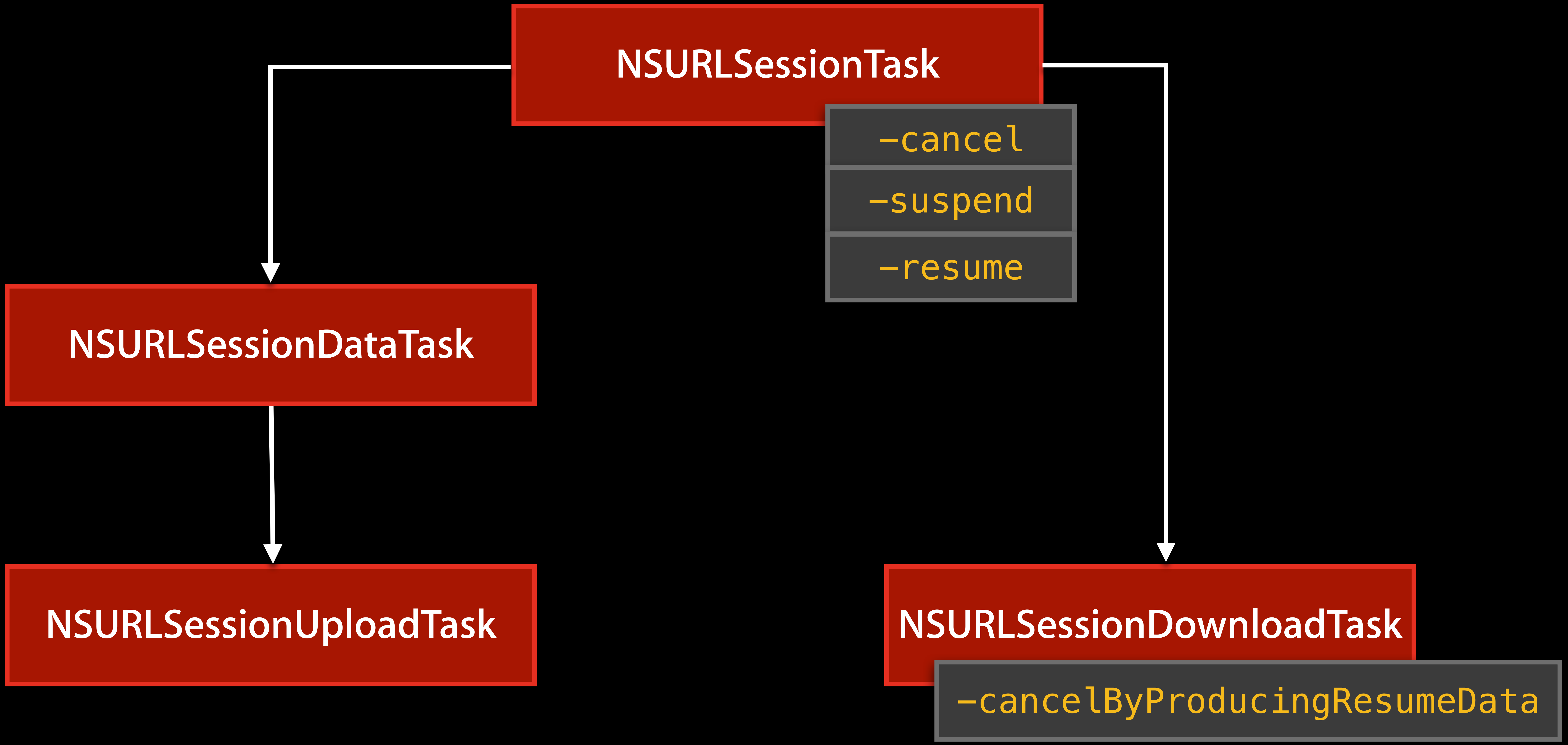
NSURLSessionTask



NSURLSessionTask



NSURLSessionTask



NSURLSessionDelegate

- Single delegate for all NSURLSession messages
    - Session, Task, DataTask, DownloadTask
- Strongly referenced until session invalidated
- Delegate messages may block loading
    - Invoke the completion handler to continue

NSURLSessionDelegate

Delegate messages for a session

```
–URLSession:didReceiveAuthenticationChallenge:completionHandler:
–URLSession:didBecomeInvalidWithError:
```

- For connection level auth
  - NTLM
  - Server Trust evaluation
  - Client Certificate
  - Kerberos implicitly handled

NSURLSessionTaskDelegate

Delegate messages for any task

`...task:willPerformHTTPRedirection:newRequest:completionHandler:`
`...task:didReceiveAuthenticationChallenge:completionHandler:`

- Request based challenges
- Basic, Digest, Proxies

`...task:didSendBodyData:totalBytesSent:totalBytesExpectedToSend:`
`...task:needsNewBodyStream:`

- Not needed if uploading from a `file://` or `NSData`
- May be called multiple times

`...task:didCompleteWithError:`

- Error will be nil for successful requests

NSURLSessionDataDelegate

Delegate messages for data tasks

`...dataTask:didReceiveResponse:completionHandler:`

- Allows you to turn a DataTask into a DownloadTask

`...dataTask:didBecomeDownloadTask:`

- No more messages for this data task

`...dataTask:didReceiveData:`

- Incremental data loading
  - `–[NSData enumerateByteRangesUsingBlock:]`

`...dataTask:willCacheResponse:completionHandler:`

- Default is to attempt to cache

NSURLSessionDownloadDelegate

Delegate messages for download tasks

`...downloadTask:didFinishDownloadingToURL:`

- Open or move the file during the callback

`...downloadTask:didWriteData:totalBytesWritten:totalBytesExpectedToWrite:`
`...downloadTask:didResumeAtOffset:expectedTotalBytes:`

- Resume offset may be less than previous reported totalBytesWritten

NSURLSession

- Default Session shares NSURLConnection stack

```
[NSURLSession sharedSession].configuration.HTTPCookieStorage ==
[NSHTTPCookieStorage sharedStorage]
```

- Custom sessions with private configuration
- Invalidation required for your sessions

```
–URLSession:didBecomeInvalidWithError:
```

- Creates Data, Upload, Download task objects
- Asynchronous convenience APIs
  - Can share delegate for auth
  - Cancelable

NSURLSession

Custom configuration

- Copy or create a configuration
- If using the task convenience routines, no delegate required
- Private Browsing example:

```
NSURLSessionConfiguration* myConfig = [NSURLSessionConfiguration
                                       ephemeralConfiguration];
NSURLSession* mySession = [NSURLSession
                           sessionWithConfiguration:myPrivateSession];
NSURL* myPrivateURL = [NSURL URLWithString:@”http://apple.com/secret/”];
[mySession dataTaskWithHTTPGetRequest:myPrivateURL
           completionHandler:^(NSData* data, NSURLResponse* response,
                               NSError* error) { [self gotSecret:data]; }
```

NSURLSession

Custom configuration

- Copy or create a configuration
- If using the task convenience routines, no delegate required
- Private Browsing example:

```
NSURLSessionConfiguration* myConfig = [NSURLSessionConfiguration
                                       ephemeralConfiguration];
NSURLSession* mySession = [NSURLSession
                           sessionWithConfiguration:myPrivateSession];
NSURL* myPrivateURL = [NSURL URLWithString:@"http://apple.com/secret/"];
[mySession dataTaskWithHTTPGetRequest:myPrivateURL
           completionHandler:^(NSData* data, NSURLResponse* response,
                               NSError* error) { [self gotSecret:data]; }
```

NSURLSession

Custom configuration

- Copy or create a configuration
- If using the task convenience routines, no delegate required
- Private Browsing example:

```
NSURLSessionConfiguration* myConfig = [NSURLSessionConfiguration
                                        ephemeralConfiguration];
NSURLSession* mySession = [NSURLSession
                           sessionWithConfiguration:myPrivateSession];
NSURL* myPrivateURL = [NSURL URLWithString:@"http://apple.com/secret/"];
[mySession dataTaskWithHTTPGetRequest:myPrivateURL
           completionHandler:^(NSData* data, NSURLResponse* response,
                               NSError* error) { [self gotSecret:data]; }
```

NSURLSession

Custom configuration

- Copy or create a configuration
- If using the task convenience routines, no delegate required
- Private Browsing example:

```
NSURLSessionConfiguration* myConfig = [NSURLSessionConfiguration
                                        ephemeralConfiguration];
NSURLSession* mySession = [NSURLSession
                            sessionWithConfiguration:myPrivateSession];
NSURL* myPrivateURL = [NSURL URLWithString:@"http://apple.com/secret/"];
[mySession dataTaskWithHTTPGetRequest:myPrivateURL
           completionHandler:^(NSData* data, NSURLResponse* response,
                               NSError* error) { [self gotSecret:data]; }
```

NSURLSession

Custom configuration

- Copy or create a configuration
- If using the task convenience routines, no delegate required
- Private Browsing example:

```
NSURLSessionConfiguration* myConfig = [NSURLSessionConfiguration
                                        ephemeralConfiguration];

NSURLSession* mySession = [NSURLSession
                           sessionWithConfiguration:myPrivateSession];
NSURL* myPrivateURL = [NSURL URLWithString:@"http://apple.com/secret/"];
 [mySession dataTaskWithHTTPGetRequest:myPrivateURL
           completionHandler:^(NSData* data, NSURLResponse* response,
                               NSError* error) { [self gotSecret:data]; }
```

NSURLSession

Custom configuration

- Copy or create a configuration
- If using the task convenience routines, no delegate required
- Private Browsing example:

```
NSURLSessionConfiguration* myConfig = [NSURLSessionConfiguration
                                        ephemeralConfiguration];
myConfig.allowsCellularAccess = NO;
NSURLSession* mySession = [NSURLSession
                           sessionWithConfiguration:myPrivateSession];
NSURL* myPrivateURL = [NSURL URLWithString:@"http://apple.com/secret/"];
 [mySession dataTaskWithHTTPGetRequest:myPrivateURL
           completionHandler:^(NSData* data, NSURLResponse* response,
                               NSError* error) { [self gotSecret:data]; }
```

NSURLSession

Data Task Creation

- Delegate-based tasks

```
-(NSURLSessionDataTask*) dataTaskWithRequest:(NSURLRequest*) request;
-(NSURLSessionDataTask*) dataTaskWithHTTPGetRequest:(NSURL*) url;
```

- Asynchronous Task Conveniences

```
-(NSURLSessionDataTask*) dataTaskWithRequest:(NSURLRequest*) request
                           completionHandler:(void (^) (NSData* data,
                                              NSURLResponse* response,
                                              NSError* error)
                                             completionHandler;
```

NSURLSession

Upload Task Creation

- Upload tasks

```
-(NSURLSessionUploadTask*) uploadTaskWithRequest:(NSURLRequest*) request
                                        fromFile:(NSURL*) file;

-(NSURLSessionUploadTask*) uploadTaskWithRequest:(NSURLRequest*) request
                                        fromData:(NSData*) data;

-(NSURLSessionUploadtask*) uploadTaskWithStreamedRequest:(NSURLRequest*) r;
```

  - Your delegate must implement `-needsNewBodyStream:`

- Asynchronous Upload Conveniences

```
    ...:fromFile:completion:^(NSData*, NSURLResponse*, NSError*) completion;
    ...:fromdata:completion:
```

NSURLSession

Download Task Creation

- Download tasks

```
-(NSURLSessionDownloadTask*) downloadTaskWithRequest:(NSURLRequest*) request;
-(NSURLSessionDownloadTask*) downloadTaskWithResumeData:(NSData*) data;
```

- Asynchronous Download Conveniences

```
...downloadTaskWithRequest:completionHandler:^(NSURL* fileURL
                                               NSURLResponse* response,
                                               NSError* error)
                                               completionhandler;
...downloadTaskWithResumeData:completionHandler:
```

- Connection errors produce resume data too

```
[[error userInfo] objectForKey:NSURLSessionDownloadTaskResumeData]
```

NSURLSession

Background Transfers

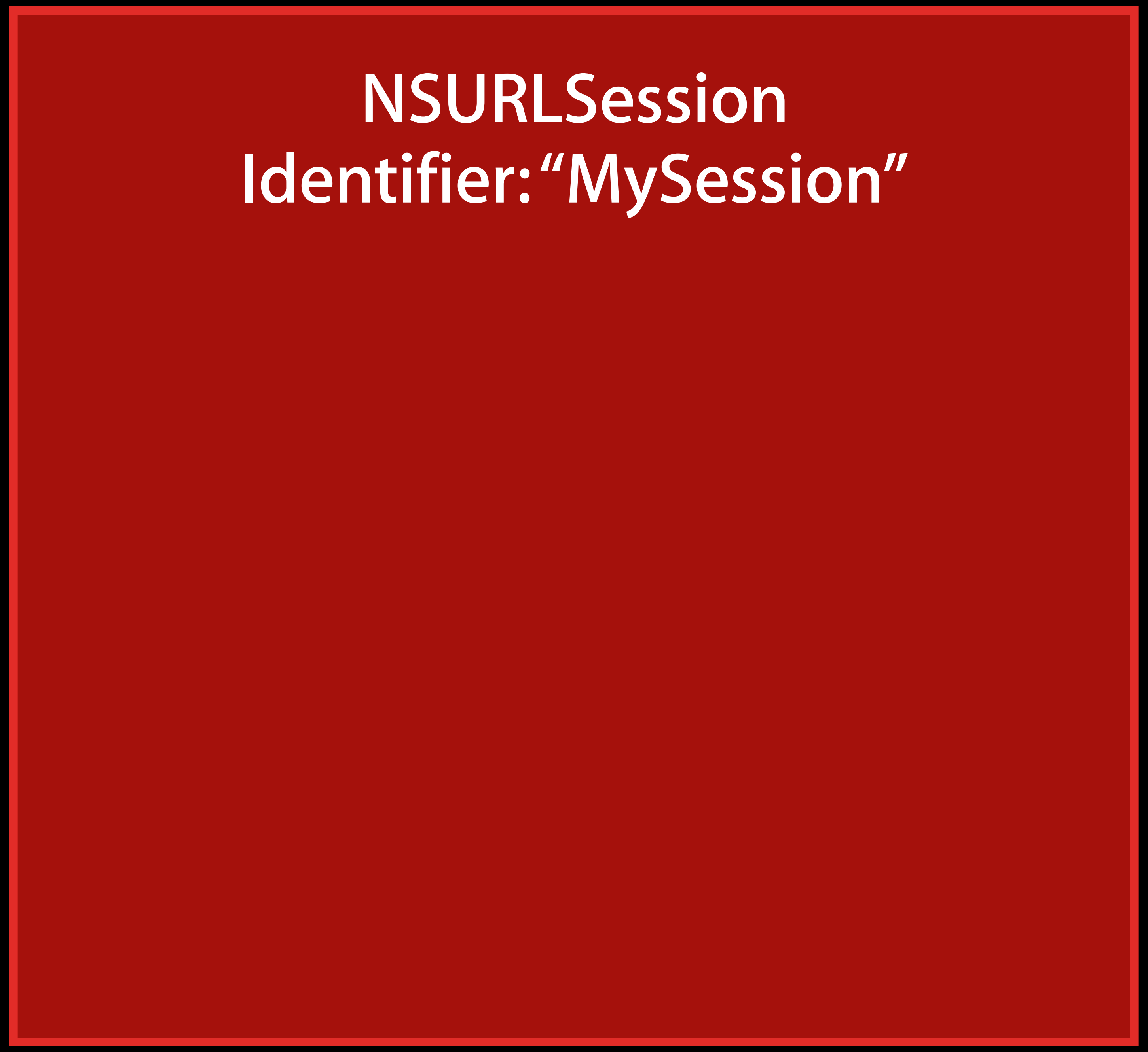
- Supports upload and download using HTTP(S)
- Requires a delegate for event delivery
    - Uses same Upload and Download task delegates as in-process
- Redirections are always taken
- `–discretionary` configuration property
    - Available on iOS, only applies to background transfers
    - Optimizes for power and network

NSURLSession

Background Transfers

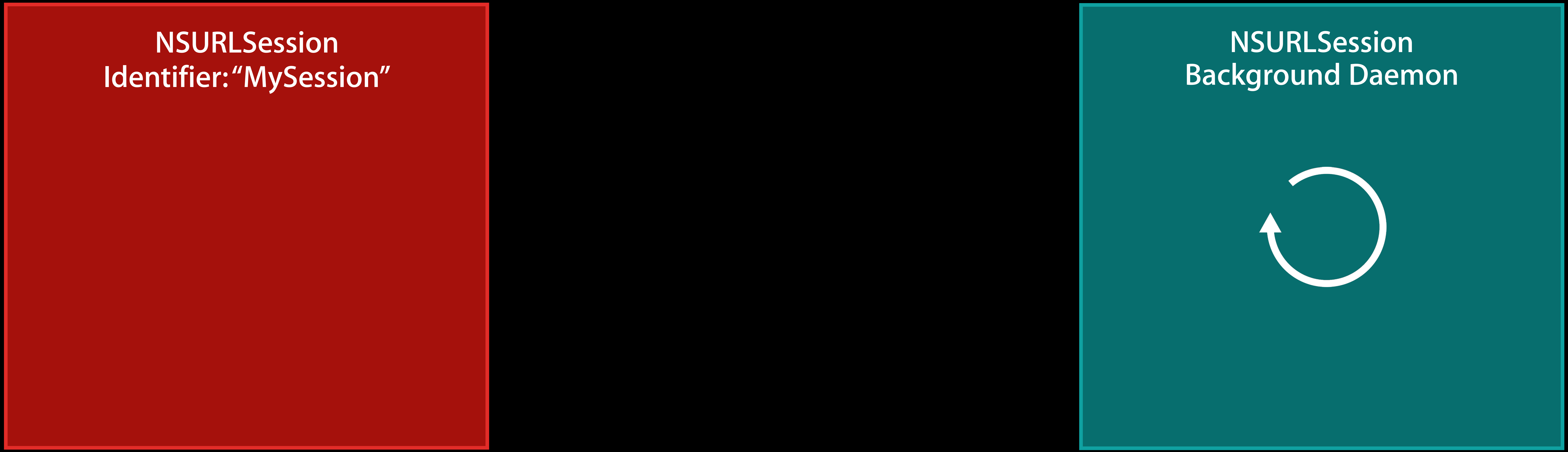
NSURLSession

Background Transfers



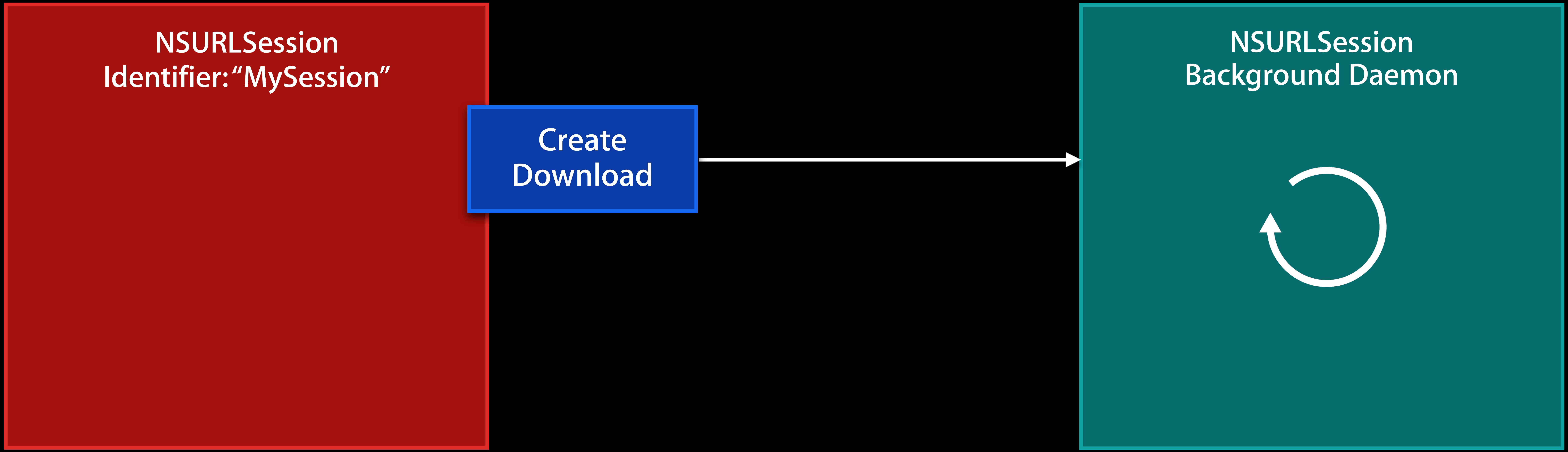
NSURLSession

Background Transfers



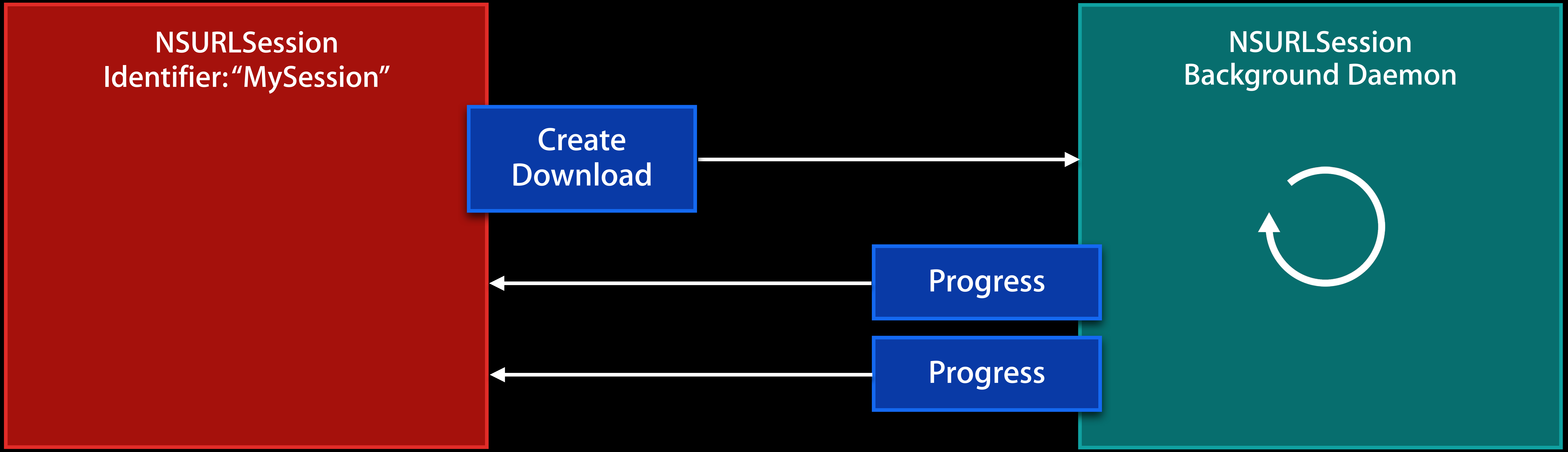
NSURLSession

Background Transfers



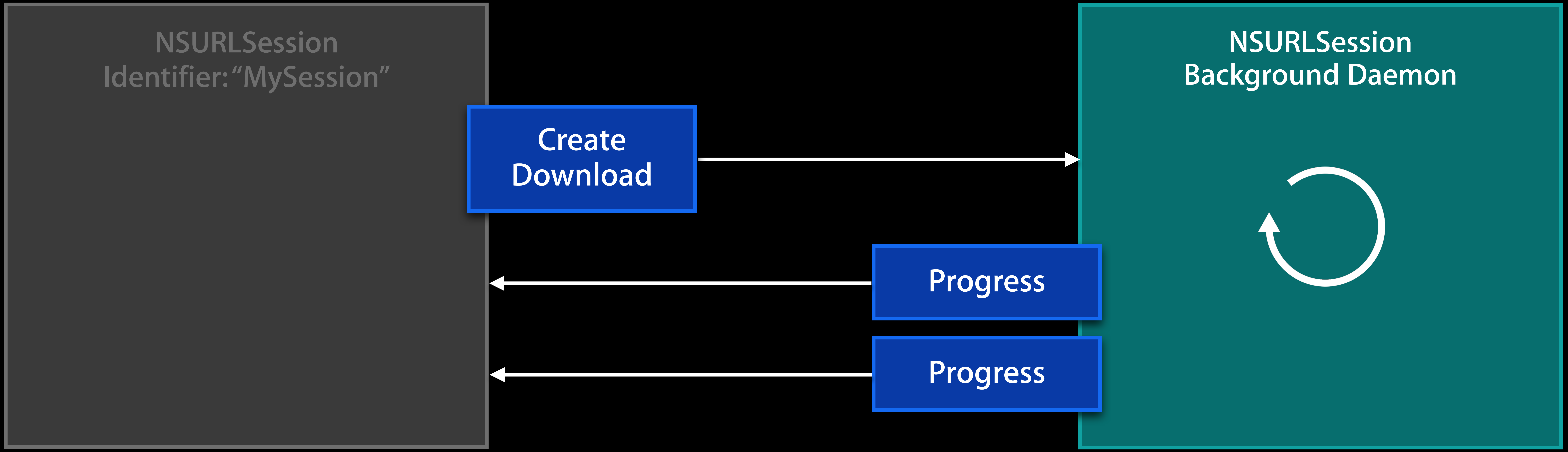
NSURLSession

Background Transfers



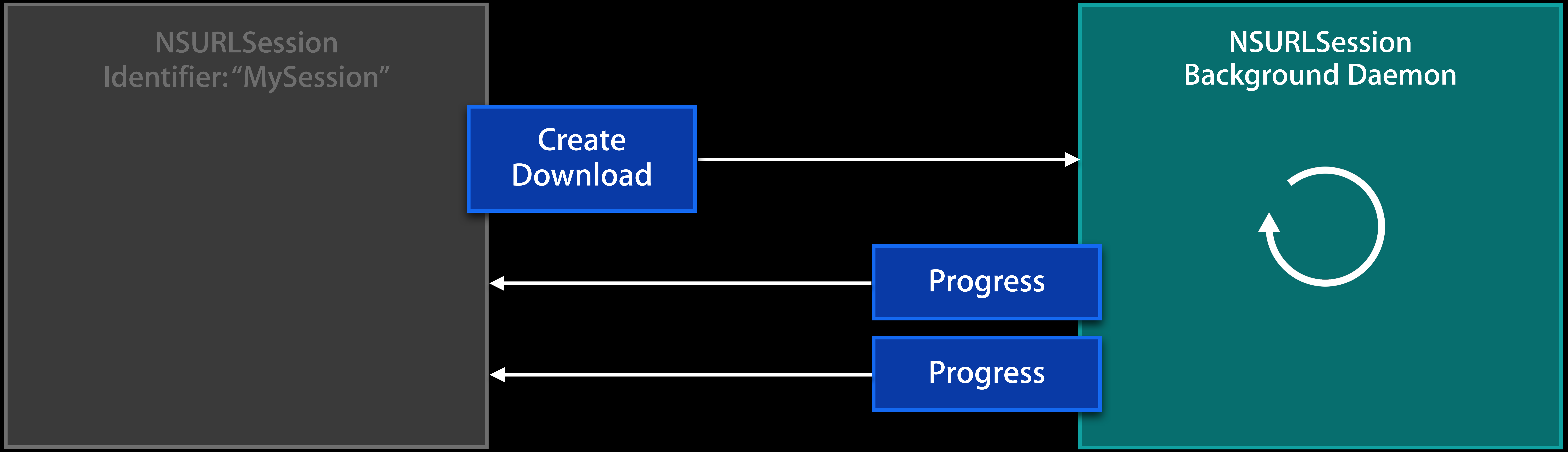
NSURLSession

Background Transfers



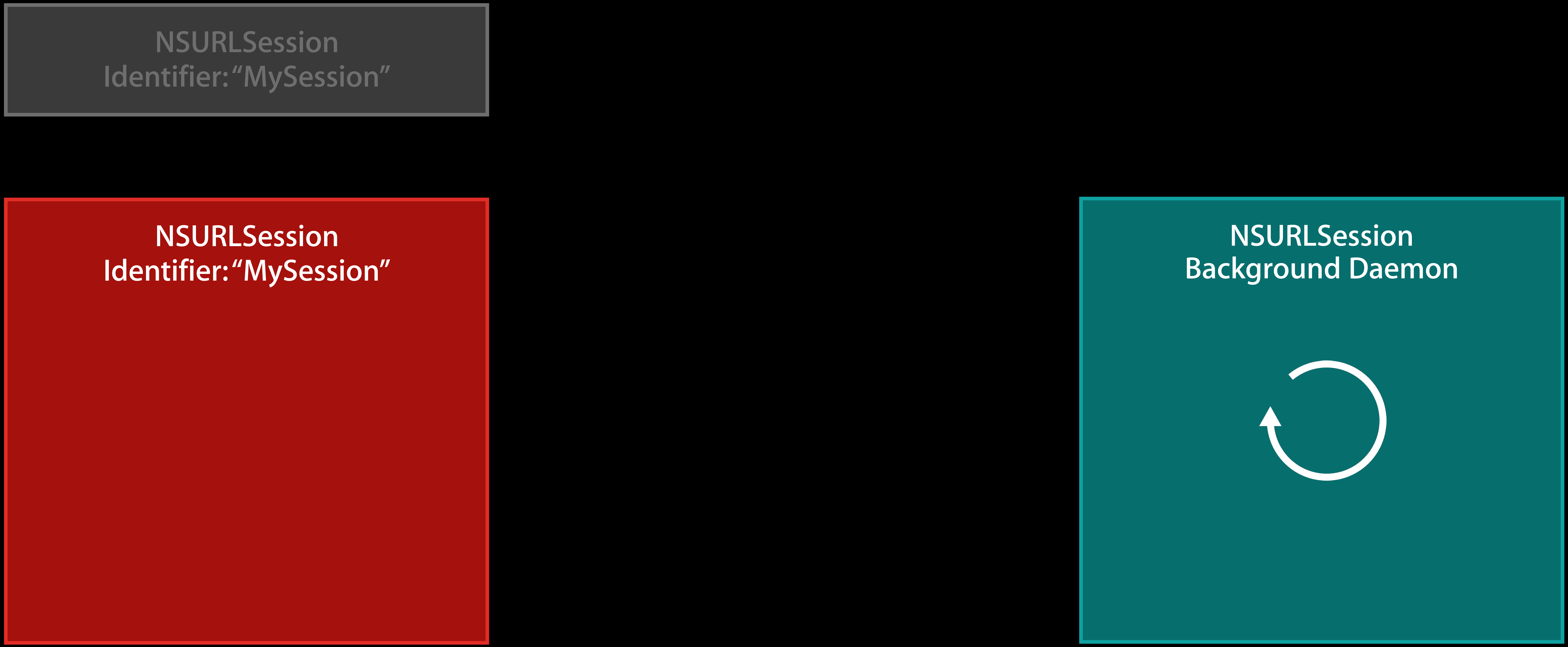
NSURLSession

Background Transfers



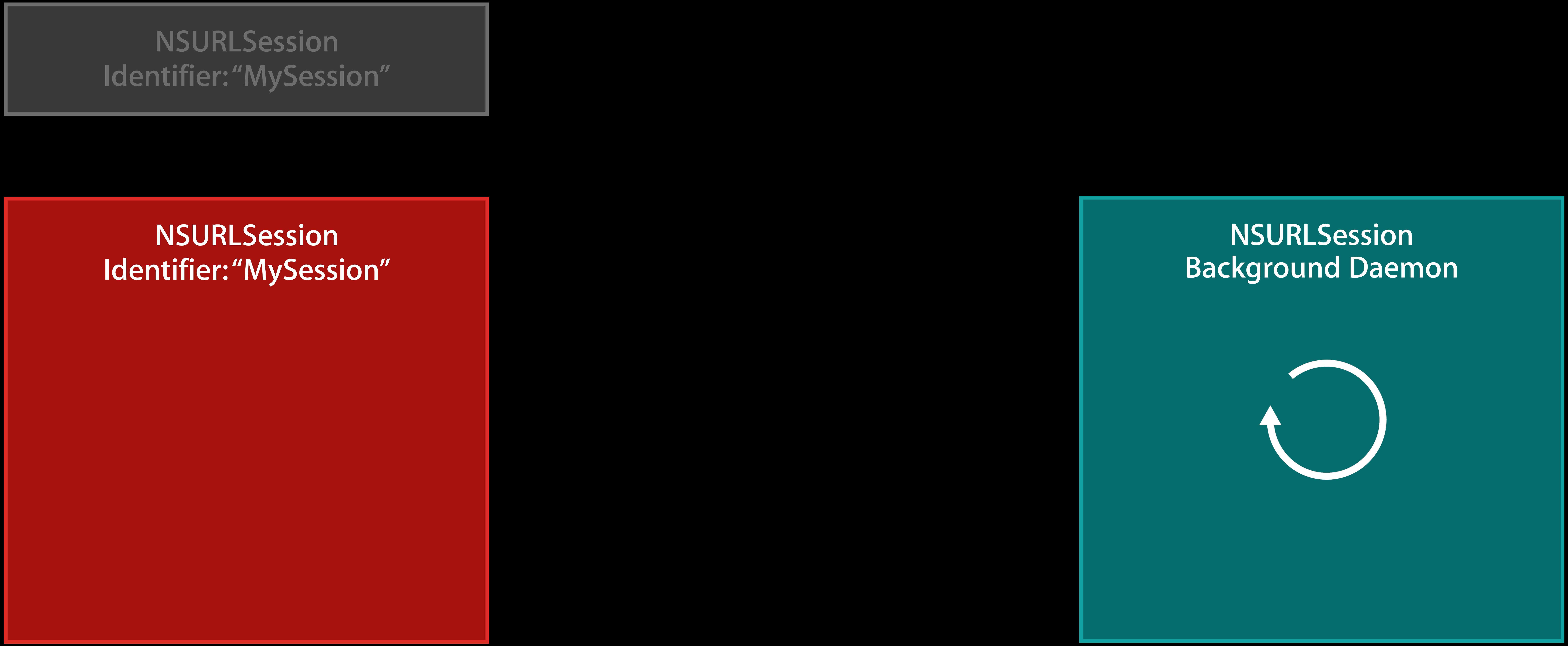
NSURLSession

Background Transfers



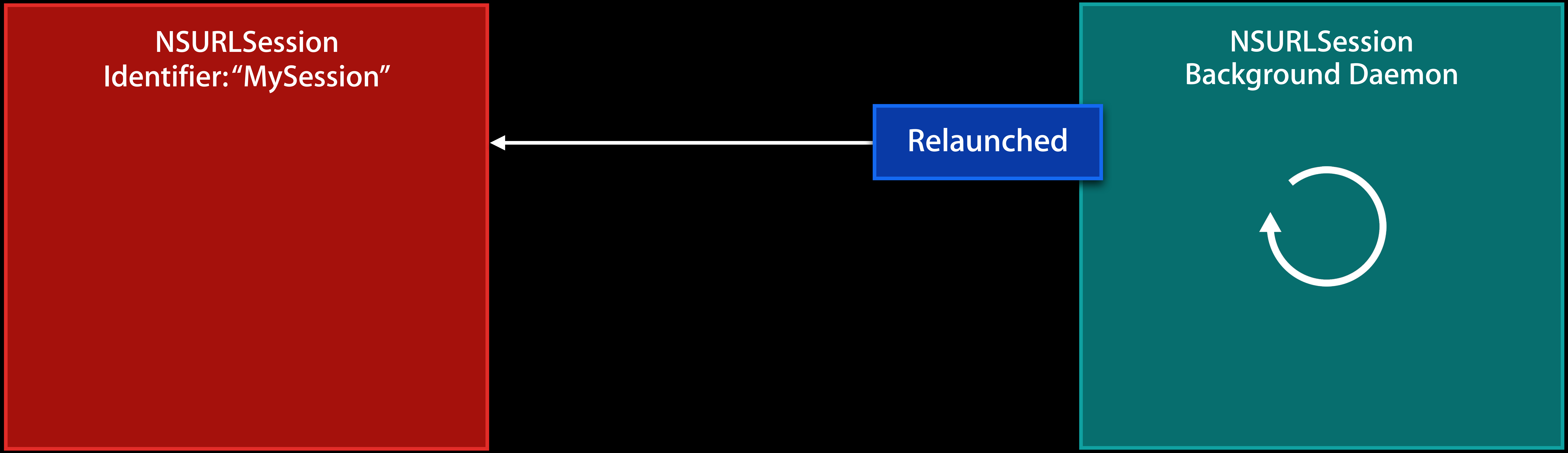
NSURLSession

Background Transfers

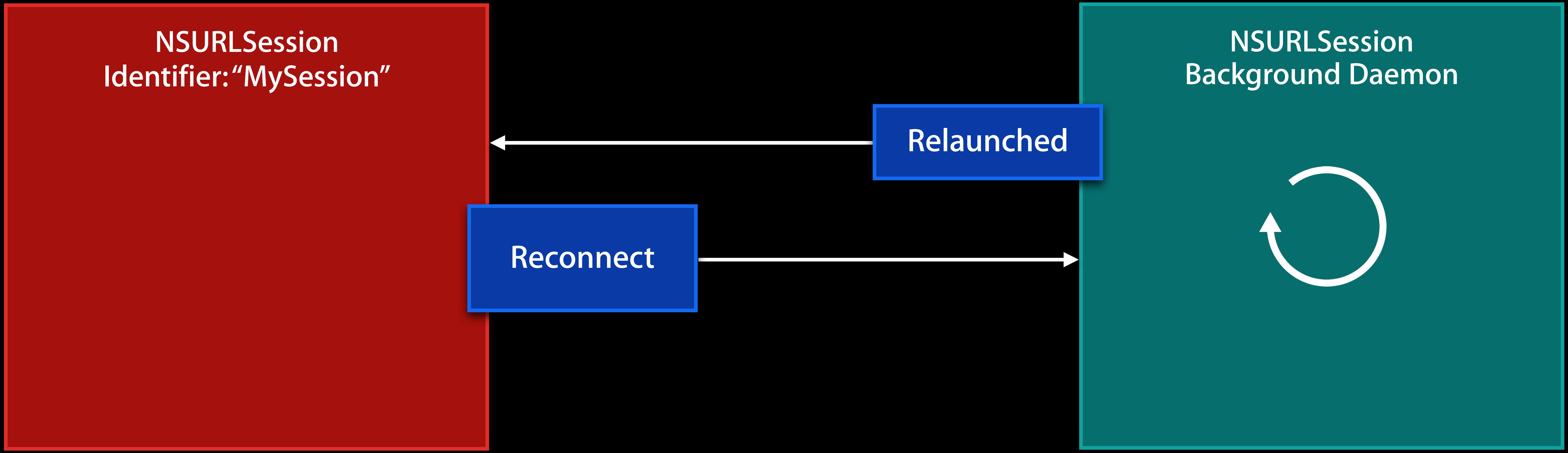


NSURLSession

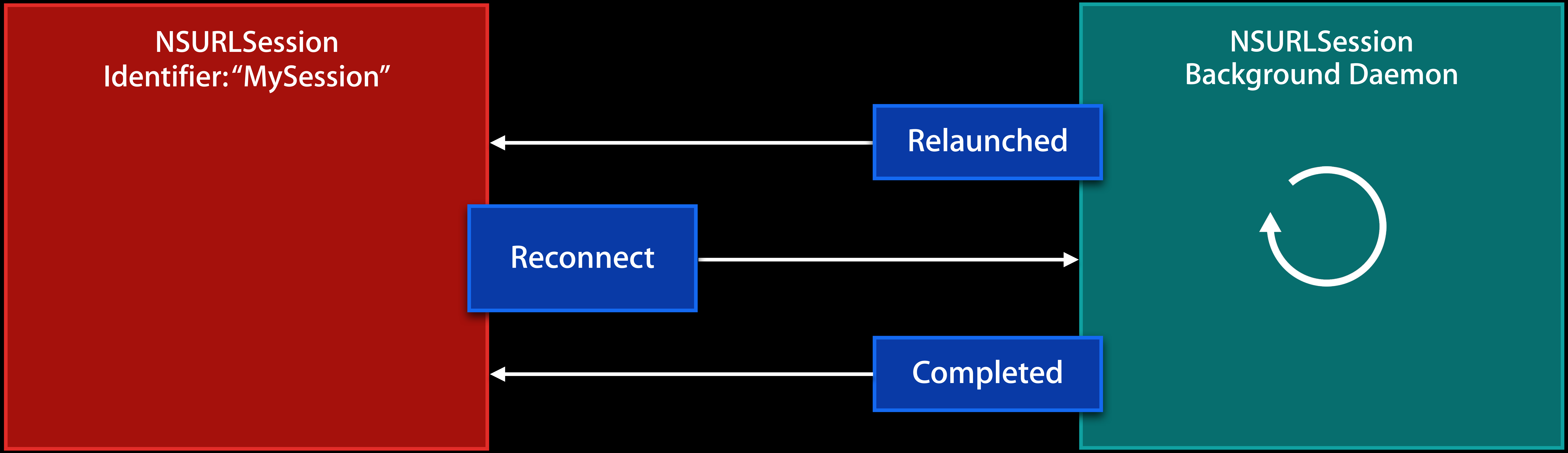
Background Transfers



NSURLSession
Background Transfers



NSURLSession
Background Transfers



NSURLSession

Out-of-process Transfers

- Delegate messages received while you’re running
- Your app will be launched in the background…
  - to service auth requests
  - when all tasks complete
- Creating a session from same identifier “reconnects" you to existing background session

```
-(void) getTasksWithCompletionHandler:^(NSArray* dataTasks,
                                        NSArray* uploadTasks,
                                        NSArray* downloadTasks)
                                        completion;
```

Demo

NSURLSession - Background Requests

Dan Vinegrad
Software Engineer

NSURLSession vs. NSURLConnection

- Connection based auth schemes
- HTTP configuration options
- Private, subclassable storage
- Background, out-of-process transfers
- API baseline

What's New in Foundation Networking

- New NSURLSession API
  - iOS 7, OS X 10.9
  - Out-of-process background transfers
- Framework Enhancements
  - NSNetServices
  - Single sign-on
  - iCloud credential syncing

NSNetServices

NEW

- Browse for and connect to Bonjour services

```
-(BOOL) getInputStream:(NSInputStream**) inputStreamPtr
            outputStream:(NSOutputStream**) outputStreamPtr;
```

- New property: `includesPeerToPeer`
  - Browsing and publishing on Peer to Peer Wi-Fi and Bluetooth
  - Peer to Peer Wi-Fi new in iOS 7

NSNetServices

NEW

- New option: `NSNetServiceListenForConnections`

```
-(void) [NSNetService publishWithOptions:(NSNetServiceOptions) options
```

- Binds IPv4 and IPv6 listening sockets
- Invokes delegate on incoming connections:

```
-(void) netService:(NSNetService*) service
   didAcceptConnectionWithInputStream:(NSInputStream*) is
                     outputStream:(NSOutputStream*) os;
```

Authentication
Single sign-on

NEW

- Kerberos Authentication
- Available in MDM environments
- Device Managers specify:
  - Applicable URLs
  - Applications
- Kerberos authentication challenges are handled by the system
- See "Extending Your Apps for Enterprise and Education Use" session

iCloud Credential Syncing

- Credentials synced between devices through iCloud
- Credential persistence option:

```
NSURLCredentialPersistenceSynchronizable
```

- Credential storage API:

```
-(void) removeCredential:(NSURLCredential*) credential
      forProtectionSpace:(NSURLProtectionSpace*) protectionSpace
                 options:(NSDictionary*) options;

Key: NSURLCredentialStorageRemoveSynchronizableCredentials
```

Removes a credential across all participating devices

Summary

- NSURLSession
    - New API for iOS 7.0 and OS X 10.9
    - Replaces NSURLConnection
    - Extensive Customization
    - Out-of-process background transfers

Summary

- NSNetServices
  - Peer to Peer support
  - Server creation
- NSURLAuthentication
  - Kerberos single sign-on
  - iCloud credential syncing

More Information

Paul Danbold
Core OS Technologies Evangelist
danbold@apple.com

Documentation
Foundation Class Reference
http://developer.apple.com/

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Managing Apple Devices	Pacific Heights Tuesday 11:30AM	🟩
What’s New with Multitasking	Presidio Tuesday 2:00PM	🟦
Extending Your Apps for Enterprise and Education Use	Nob Hill Tuesday 3:15PM	🟩
Nearby Networking with Multipeer Connectivity	Mission Wednesday 10:15AM	🟦
What’s New in State Restoration	Mission Thursday 3:15PM	🟦

Labs

Foundation Networking Lab	Core OS Lab B Wednesday 10:15AM
Multipeer Connectivity Lab	Core OS Lab A Wednesday 11:30AM
Cocoa and Foundation Lab	Frameworks Lab A Wednesday 11:30AM
Networking Lab	Core OS Lab A Thursday 9:00AM
Multitasking Lab	Services Lab B Thursday 9:00AM
Multipeer Connectivity Lab	Core OS Lab B Friday 9:00AM

WWDC2013