Works with iPhone

External Accessory Development

iPhoneDevCon
28-September-2010
Michael Gile

FUZZYLUKE

Agenda

- The Made for iPod Program
- Intro to Apple Accessories
- Anatomy of an Accessory
- External Accessory Framework
- XCode - Info.plist Requirements
- Code - Device Comms Lifecycle
- Device Coding Strategies
- How Car Diagnostics Work
- Car Diagnostic App Demo
- Other Hardware Options

Please Ask Questions At Any Time

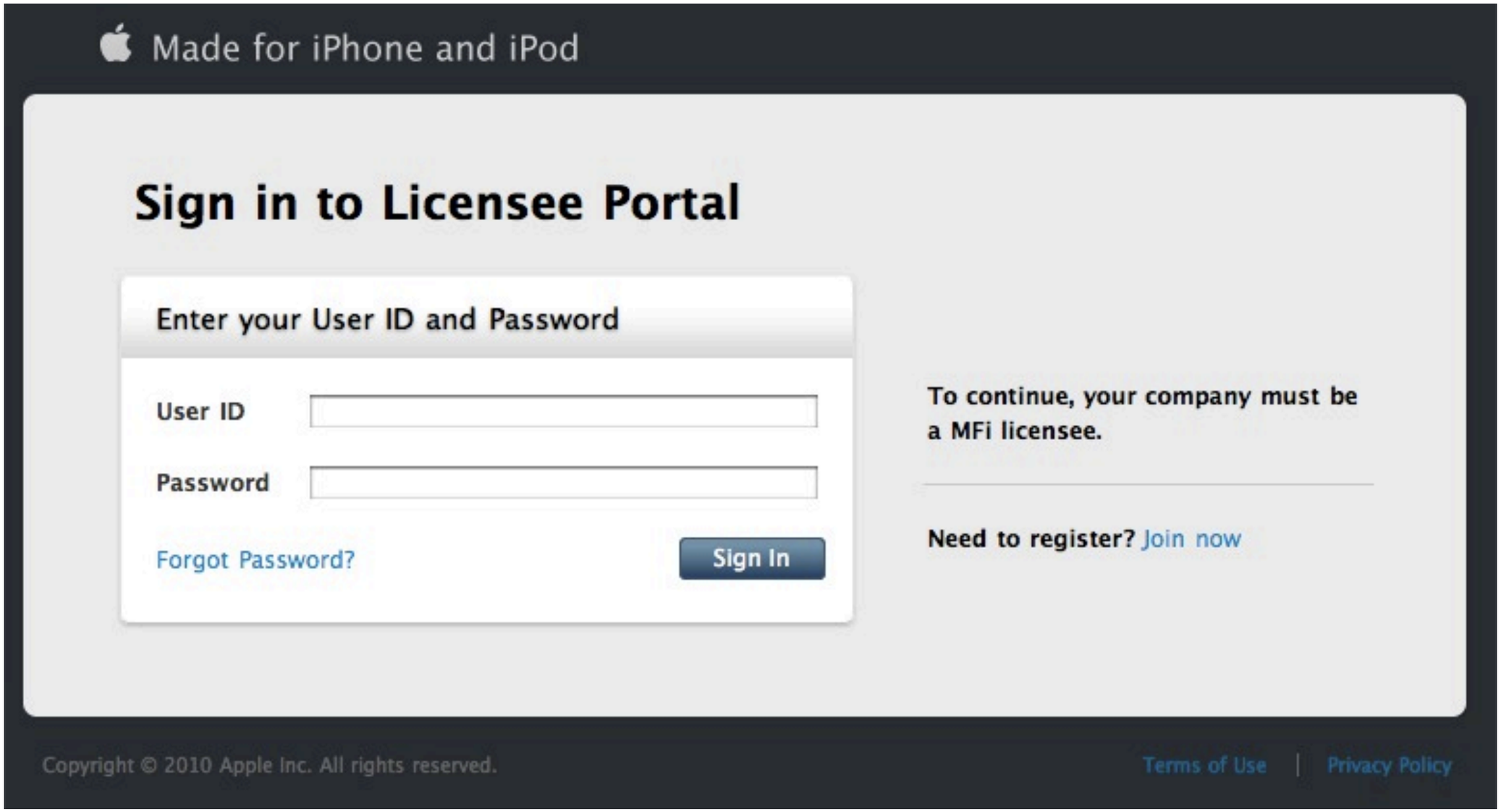
"The Made for iPod and Works with iPhone logos mean that an electronic accessory has been designed to connect specifically to iPod and iPhone and has been certified by the developer to meet Apple performance standards."

- http://developer.apple.com/ipod/

What's Possible?

Intro to Apple Accessories

- 6-10 Month Process
- Have Detailed Plan
- Submit For Apple Approval
- Design Hardware

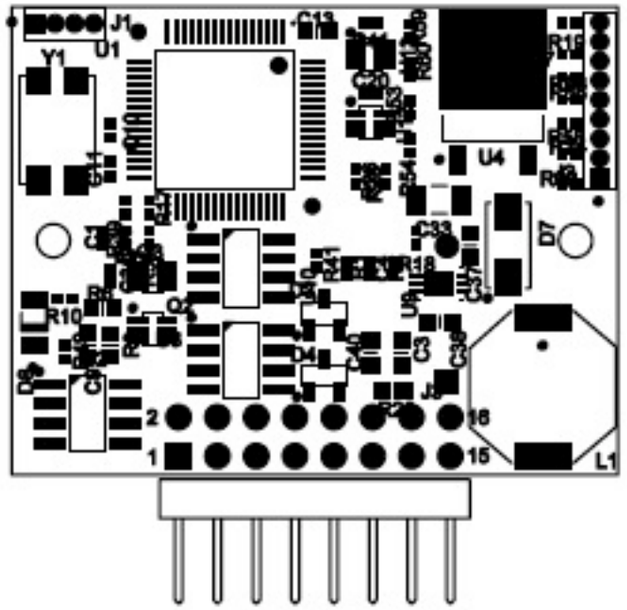


SDK-Like Portal

What do you get from MFI?

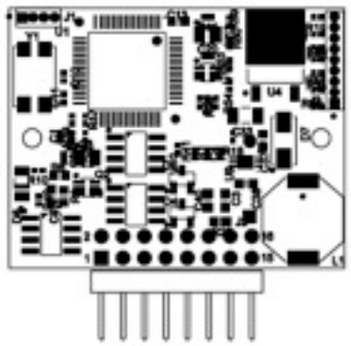
- No Cost To Join
- Portal Access
  - Detailed technical reference(s)
  - Fast Licensee Support
- Technology Evangelist Access
- Separate Tech Support

Anatomy of an Accessory



Step 0 - IAP

[REDACTED]



Step 1 - EASession

NSInputStream/NSOutputStream

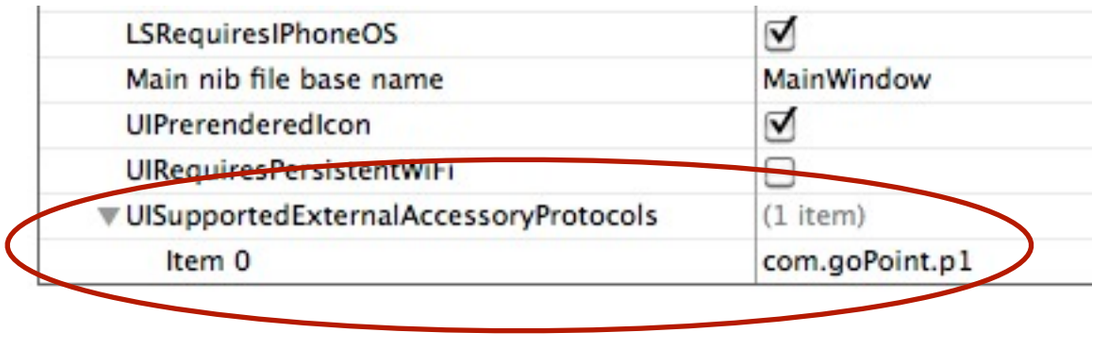
External Accessory Framework

EA Framework

- EAAccessoryManager
- EAAccessory
- EAAccessoryDelegate
- EASession

Info.plist Requirements

Let iPhoneOS know that your application supports the specified EAAcessory protocols



Connection Notifications

```
- (void) registerForNotifications {
   FLINFO(@"*** Registering for EA Notifications ***")
   [[NSNotificationCenter defaultCenter] addObserver:self
                                            selector:@selector(_accessoryConnected:)
                                                name:EAAccessoryDidConnectNotification
                                              object:nil];

   [[NSNotificationCenter defaultCenter] addObserver:self
                                            selector:@selector(_accessoryDisconnected:)
                                                name:EAAccessoryDidDisconnectNotification
                                              object:nil];

   [[EAAccessoryManager sharedAccessoryManager] registerForLocalNotifications];
}
```

```
- (void) _accessoryConnected:(NSNotification *)notification {
    FLTRACE_ENTRY
    EAAccessory* connectedAccessory = [[notification userInfo] objectForKey:EAAccessoryKey];
    [_connectedAccessoryList addObject:connectedAccessory];

    NSString* accessoryName = [[NSString alloc] initWithString:[connectedAccessory name]];
    FLDEBUG(@"Found external accessory: %@", accessoryName);
}
```

Disconnected - Notification & Delegate

```
- (void) _accessoryDisconnected:(NSNotification *)notification {

  FLTRACE_ENTRY
  EAAccessory* disconnectedAccessory = [[notification userInfo]
                                        objectForKey:EAAccessoryKey];

}
```

```
- (void) accessoryDidDisconnect:(EAAccessory *)accessory {

  FLDEBUG(@"the accessory was disconnected", nil)
  [self dispatchDelegate:@selector(scanToolDidDisconnect:)
              withObject:nil];

}
```

Device NSOperation

```
_streamOperation    = [[NSInvocationOperation alloc] initWithTarget:self
                                                           selector:@selector(runStreams)
                                                             object:nil];

_scanOperationQueue = [[NSOperationQueue alloc] init];
[_scanOperationQueue addOperation:_streamOperation];
[_scanOperationQueue setSuspended:NO];
```

- Operate Your Device In Background

Operation

```
- (void) runStreams {
    NSAutoreleasePool * pool  = [[NSAutoreleasePool alloc] init];
    NSRunLoop* currentRunLoop = [NSRunLoop currentRunLoop];
    NSDate* distantFutureDate = [NSDate distantFuture];
    @try {
       [self open];
       [self initScanTool];

       if ([self isEAScanTool]) {
          while (!_streamOperation.isCancelled &&
                 [currentRunLoop runMode:NSDefaultRunLoopMode
                                beforeDate:distantFutureDate]) {
             ;
          }
       }
       FLINFO(@"*** STREAMS CANCELLED ***")
    }
    @catch (NSException * e) {
       FLEXCEPTION(e)
    }
    @finally {
       [self close];
       [pool release];
    }
}
```

Open Session

```
- (BOOL) openSession {

  [_accessory setDelegate:self];

  if (!_session) {
     _session = [[EASession alloc] initWithAccessory:_accessory
                                         forProtocol:_protocolString];
  }

  if (_session) {
     [[_session inputStream] setDelegate:self];
     [[_session inputStream] scheduleInRunLoop:[NSRunLoop currentRunLoop]
                                       forMode:NSDefaultRunLoopMode];
     [[_session inputStream] open];
     [[_session outputStream] setDelegate:self];
     [[_session outputStream] scheduleInRunLoop:[NSRunLoop currentRunLoop]
                                        forMode:NSDefaultRunLoopMode];

     [[_session outputStream] open];
  }
  else     {
     FLERROR(@"creating session failed", nil)
  }

  return (_session != nil);
}
```

Handle Events

```
- (void)stream:(NSStream *)theStream handleEvent:(NSStreamEvent)streamEvent {
   switch (streamEvent) {
       case NSStreamEventNone:
           FLDEBUG(@"stream %@ event none", theStream);
           break;
       case NSStreamEventOpenCompleted:
           FLDEBUG(@"stream %@ event open completed", theStream);
           break;
       case NSStreamEventHasBytesAvailable:
           FLDEBUG(@"stream %@ event bytes available", theStream);
           [self handleReadData];
           break;
       case NSStreamEventHasSpaceAvailable:
           FLDEBUG(@"stream %@ event space available", theStream);
           [self writeCachedData];
           break;
       case NSStreamEventErrorOccurred:
           FLDEBUG(@"stream %@ event error", theStream);
           break;
       case NSStreamEventEndEncountered:
           FLDEBUG(@"stream %@ event end encountered", theStream);
           break;
       default:
           FLERROR(@"Received unknown NSStreamEvent: %0x04X", streamEvent);
           break;
   }
}
```

Close Session

```
- (void) closeSession {

  [[_session inputStream] removeFromRunLoop:[NSRunLoop currentRunLoop]
                                    forMode:NSDefaultRunLoopMode];
  [[_session inputStream] setDelegate:nil];
  [[_session inputStream] close];

  [[_session outputStream] removeFromRunLoop:[NSRunLoop currentRunLoop]
                                     forMode:NSDefaultRunLoopMode];
  [[_session outputStream] setDelegate:nil];
  [[_session outputStream] close];

  [_session release];
  _session    = nil;

  [_accessory setDelegate:nil];
  [_accessory release];
  _accessory = nil;

}
```

NSRunLoop Gotchas

```
while(!_streamOperation.isCancelled &&
      [currentRunLoop runMode:NSDefaultRunLoopMode
                   beforeDate:[NSDate distantFuture]]) {
        ;
}
```

- runMode:beforeDate: NOT run
- run not guaranteed to exit after removing input sources (i.e. your streams)

Coding Strategies

- Model Accessory With Finite State Machine
  - Constant Operation
  - Periodic Operation
    - Init / Operation(s) / Error / Stop
- Expect Malformed Data
- Expect Disconnects
- Expect More Than One Accessory

Our FSM

```
typedef enum  {
  STATE_INIT  =0,
  STATE_IDLE,
  STATE_WAITING,
  STATE_PROCESSING,
  STATE_ERROR,

  NUM_STATES
} ScanToolState;
```

Event Propagation

- Record It
- Report It
- Get Back To Work

What We Do

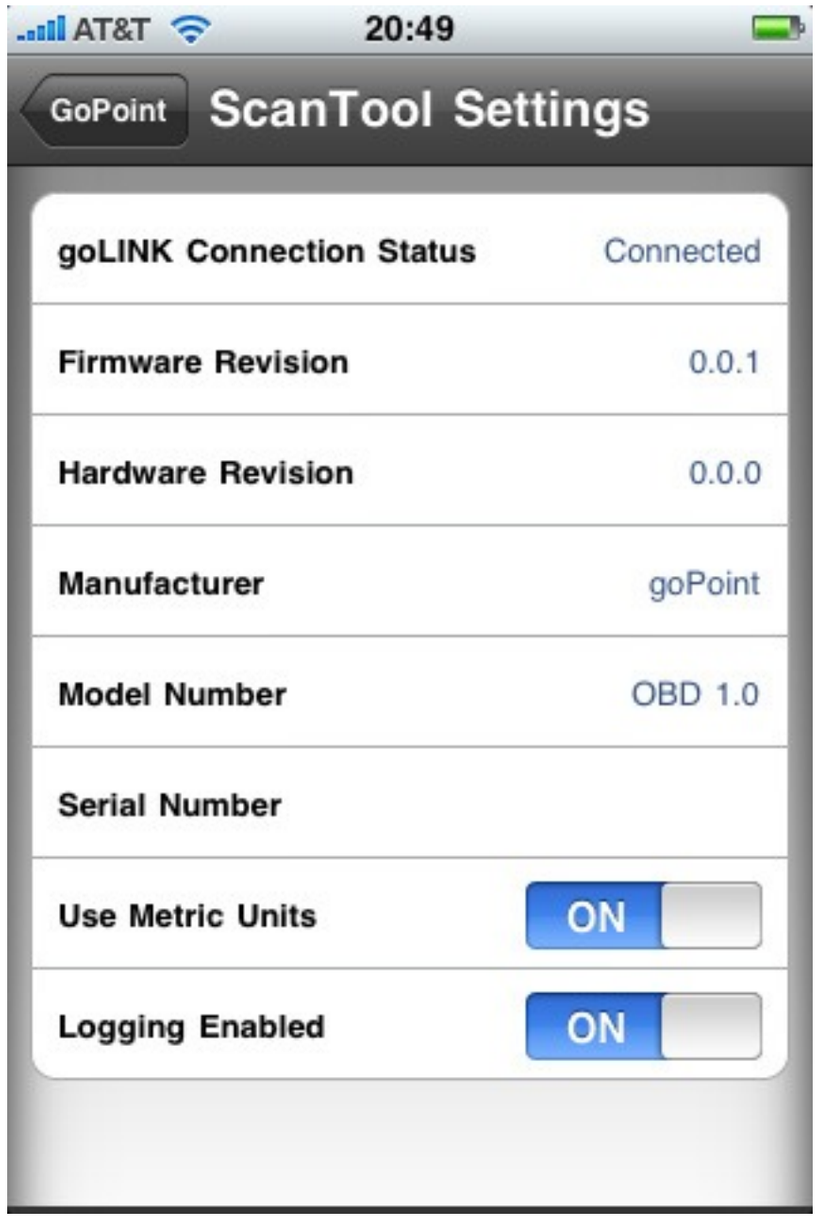
```
@protocol ScanToolDelegate <NSObject>
@optional
- (void)scanDidStart:(ScanTool*)scanTool;
- (void)scanDidPause:(ScanTool*)scanTool;
- (void)scanDidCancel:(ScanTool*)scanTool;
- (void)scanToolWillSleep:(ScanTool*)scanTool;
- (void)scanToolDidConnect:(ScanTool*)scanTool;
- (void)scanToolDidDisconnect:(ScanTool*)scanTool;
- (void)scanToolDidInitialize:(ScanTool*)scanTool;
- (void)scanToolDidFailToInitialize:(ScanTool*)scanTool;
- (void)scanTool:(ScanTool*)scanTool didSendCommand:(ScanToolCommand*)command;
- (void)scanTool:(ScanTool*)scanTool didReceiveResponse:(NSArray*)responses;
- (void)scanTool:(ScanTool*)scanTool didReceiveVoltage:(NSString*)voltage;
- (void)scanTool:(ScanTool*)scanTool didTimeoutOnCommand:(ScanToolCommand*)command;
- (void)scanTool:(ScanTool*)scanTool didReceiveError:(NSError*)error;
@end


[invocation performSelectorOnMainThread:@selector(invoke)
                             withObject:nil
                          waitUntilDone:NO];
```

Polling

Don't...Just Don't

EAAcessory Properties

Important Device Information



Easily Understood Data Structures

- This:

```
#pragma pack(1)
typedef struct golink_frame_header_t {
   uint8_t        fid;     // Frame ID
   uint8_t        address; // Frame Address
   uint8_t        length;  // Frame Length
} GoLinkFrameHeader;

#define GOLINK_FRAME_TYPE(buf) ((GoLinkFrameHeader*)buf)->fid
```

- Not This:

```
uint8_t msgbuf[256];
if(msgbuf[i+8] == 0x07) {
   ;
}
```

Debugging Strategies

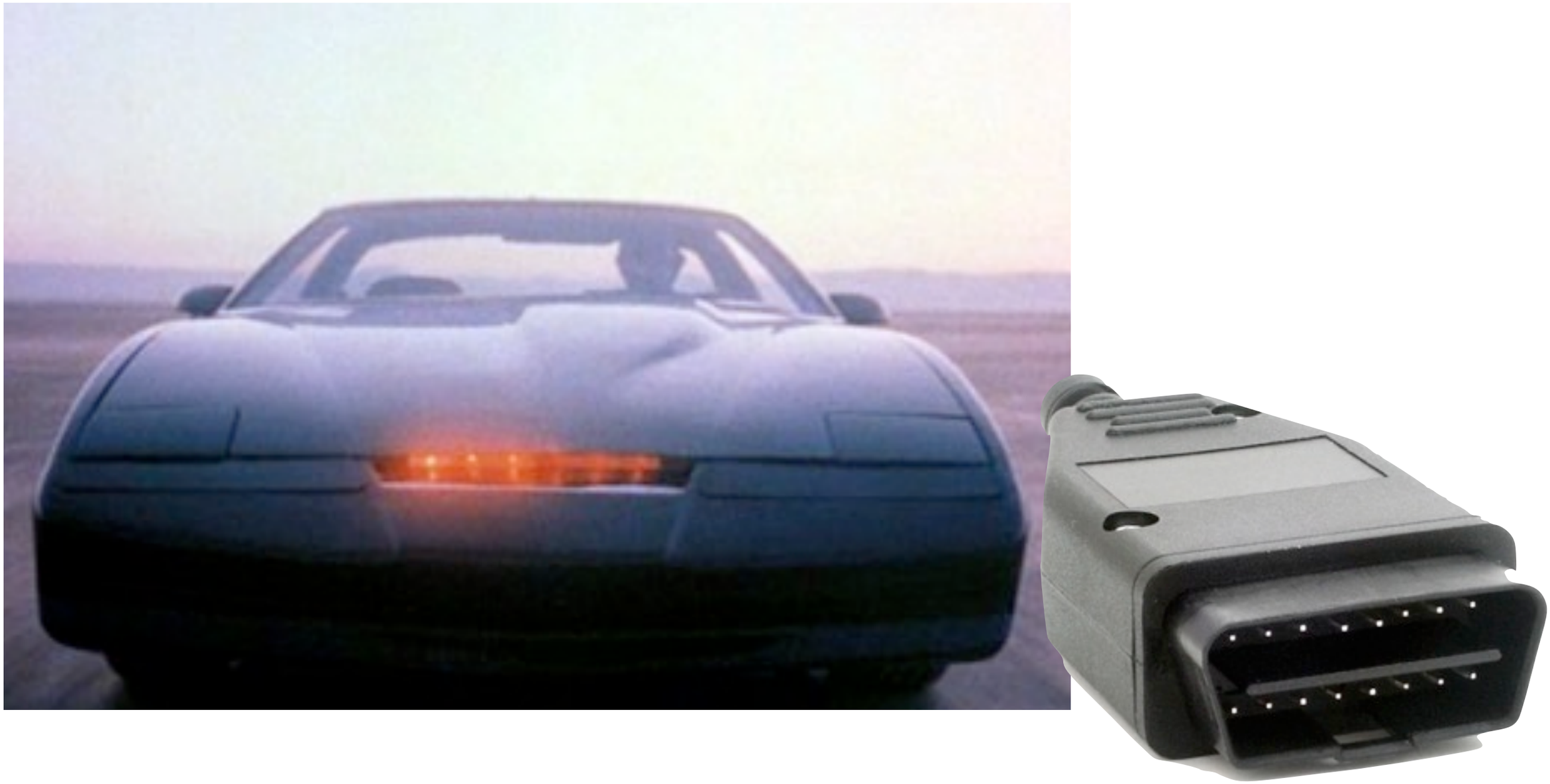
- You cannot debug an external accessory in simulator
- Build your hardware with USB breakout and debug on-device

NSLog To File

```
@implementation UIApplication (Logging)

+ (void) redirectConsoleLogToDocumentFolder:(NSString*)filename {

  NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
                                                       NSUserDomainMask, YES);

  NSString *documentsDirectory = [paths objectAtIndex:0];

  NSString *logPath = [documentsDirectory
                       stringByAppendingPathComponent:filename];

  freopen([logPath cStringUsingEncoding:NSASCIIStringEncoding],
                        "a+",stderr);
}
```

Publishing Gotchas

Don’t get bitten by review issues



What is OBD-2?

On-Board Diagnostics

Regulation enacted to require emissions testing by 3rd parties, regardless of manufacturer, year or model

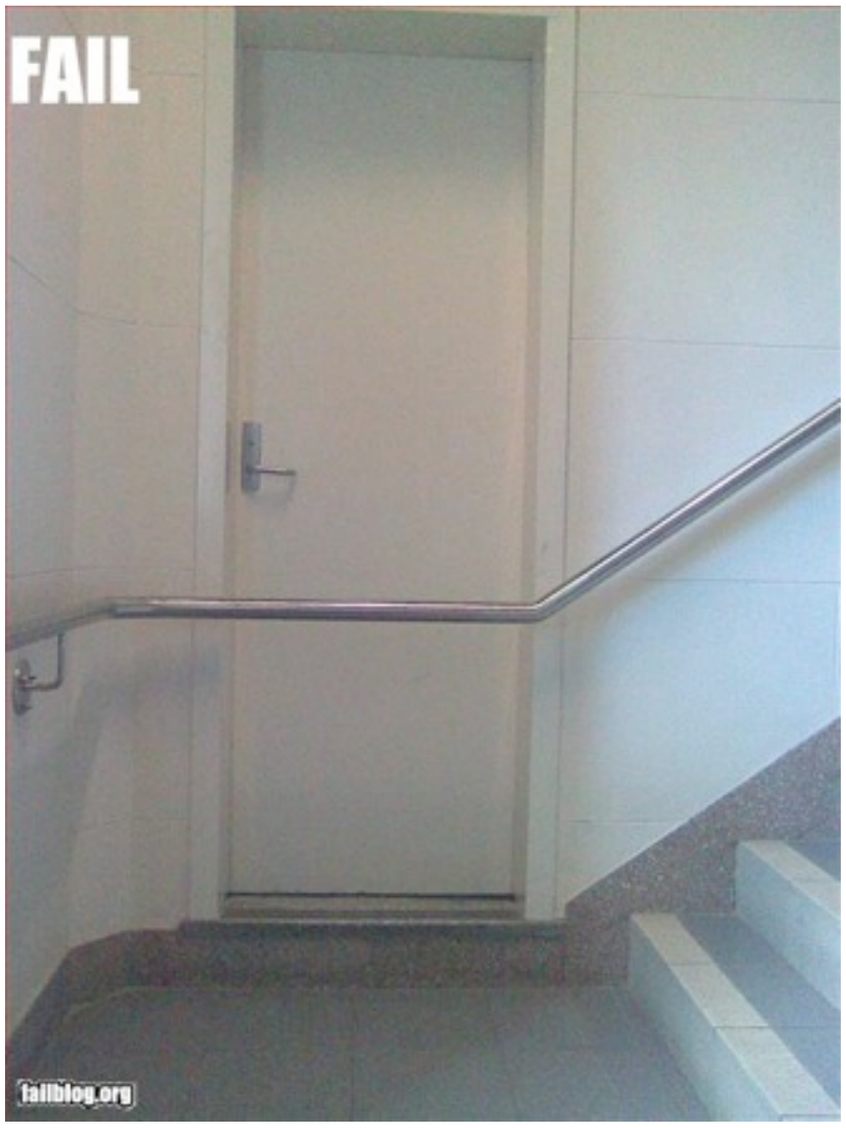
Standardized in 1996, available worldwide

How OBD-2 Works

- Standardized Command/Response Formats
- Standardized Categories (Mode)
- Standardized Sensors (PID)
- Mode 0x01, PID 0x0C ==

About That State Machine...

Cars do not behave!



Alternate Connectivity Options

- WIFI
- 3.5mm Headphone Jack

WIFI

- Cumbersome User Setup Process
- No Automatic Simultaneous Cellular Radio Data*

3.5mm Headphone Jack

- Simple Binary Read/Write
- Slow - 19.2 kbaud
- SDK Available - http://www.progical.com/

My Demo

Let me show you it

Q&A
Thank You

mgile@fuzzyluke.com
twitter: @mgile