# Ludovic Rousseau's blog

My activities related to smart card and Free Software (as in free speech).

Saturday, September 26, 2015

# PCSC sample in Swift

To continue the list of PC/SC wrappers initiated in 2010 with "PC/SC sample in different languages" I now present a sample in Swift using the Apple Crypto Token Kit API.

## Crypto Token Kit API

See my previous article "PCSC sample in Objective-C" to know more about Crypto Token Kit API.

#### Source code

Create a new Swift application in Xcode. You need to enable the App Sandbox and add/set the com.apple.security.smartcard entitlement to YES.

This code is just a, more or less direct, conversion of the Objective-C source code to Swift.

I used Xcode 7.0 that implements Swift version 2.1.

Swift is a language not yet known by my colorization tool: source-highlight. The colors may not be great.

```
import CryptoTokenKit
let mngr = TKSmartCardSlotManager.defaultManager()
// Use the first reader/slot found
let slotName = mngr!.slotNames[0]
print("slotName:", slotName)
// connect to the slot
mngr?.getSlotWithName(slotName, reply: {
    (slot: TKSmartCardSlot?) in
    // connect to the card
    let card = slot?.makeSmartCard()
   if (card != nil)
        // begin a session
        card?.beginSessionWithReply({
            (success: Bool, error: NSError?) in
            if (success)
                let aid : [UInt8] = [0xA0, 0x00, 0x00, 0x00, 0x62, 0x03, 0x01, 0x0C, 0
x06, 0x01]
                let data = NSData(bytes: aid, length: aid.count)
                card?.sendIns(0xA4, p1: 0x04, p2: 0x00, data: data, le: 0, reply: {
                    (data: NSData?, sw: UInt16, error: NSError?) in
                    if (error != nil)
                        print("sendIns error:", error!)
                    else
                        print("Response:", data!, String(sw, radix: 16))
                        // send 2nd APDU
                        let data = NSData(bytes: nil, length: 0)
                        card?.sendIns(0x0, p1: 0x00, p2: 0x00, data: data, le: 200, re
ply: {
                            (data: NSData?, sw: UInt16, error: NSError?) in
```



```
if (error != nil)
                                 print("sendIns error:", error!)
                             }
                            else
                                 print("Response:", data!, String(sw, radix: 16))
                                let newString = NSString(bytes: data!.bytes, length: d
ata!.length, encoding: NSASCIIStringEncoding)
                                print(newString!)
                        })
                })
            }
            else
                print("Session error:", error)
        })
    }
    else
    {
        print("No card found")
})
// wait for the asynchronous blocks to finish
sleep(1)
```

## Output

```
slotName: Gemalto PC Twin Reader
Response: <> 9000
Response: <48656c6c 6f20776f 726c6421> 9000
Hello world!
```

### Comments

See the previous article "PCSC sample in Objective-C" for comments about the Crypto Token Kit API.

Swift may be easier to learn and use than Objective-C. Using the Crypto Token Kit API is not different if you use Objective-C or Swift.

It looks like I am the first to provide a public code sample using the Crypto Token Kit API in Swift. Great! I hope that is a good Swift sample, I am not a Swift expert.

## Conclusion

As I wrote in "PCSC framework will stay in Mac OS X 10.11 El Capitan" the PC/SC API is not available from Swift. So the only option (for now) to use a smart card from Swift is to use the Crypto Token Kit API.

## [Update 5 Oct 2015]

The sample code was not correct in the error treatment.

I then discovered that if the card returns an "error" code then the sendIns method returns an error in the error parameter. Here is the execution output of the program with a card that does not contain the applet:

```
slotName: Gemalto PC Twin Reader
sendIns error: Error Domain=CryptoTokenKit Code=-3 "SmartCard returned error 6a82" Use
rInfo=0x600000060940 {NSLocalizedDescription=SmartCard returned error 6a82}
```

The card returns 0x6A82 (Wrong parameter(s) P1-P2, File not found) in SW (Status Word).

It may be difficult to differentiate errors at the card/reader communication level from "errors" returned by the card. A SW different from 0x9000 may be completely valid and expected by the application in some cases.



Labels: code, Mac OS X

Newer Post Home Older Post

#### Ludovic Rousseau b...



Bitcoin



License: by-nc-sa



This blog by Ludovic Rousseau is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

Simple theme. Powered by Blogger.