Stratio

LinkSquare API

Example Guide for iOS

Version 1.0.0

Last updated: 2017-09-20

Stratio

Contents

1 개요

이 문서는 LinkSquare™ Professional 에 포함된 LinkSquare™ API for iOS 의 기능 및 사용 방법에 대해 설명합니다. LinkSquare API 는 LinkSquare™ 기기와 연결 및 데이터 수집 기능을 제공하는 소프트웨어 라이브러리 입니다. LinkSquare API for iOS 는 IOS 앱 개발에 사용할 수 있는 LinkSquare API 입니다.

2 요구 사항

2.1 운영체제

LinkSquare API 는 iOS 10.0 이상의 버전을 지원합니다.

2.2 기타 소프트웨어

LinkSquare API for iOS 를 사용한 앱 개발에는 아래 소프트웨어가 필요합니다.

- macOS 10.12 이상
- Xcode 8.3 이상

2.3 하드웨어

- LinkSquare 기기
- Wi-Fi 연결 가능한 Mac 기기
- iOS 기기

3 다운로드 및 설치

LinkSquare API for iOS 는 LinkSquare Professional 설치시 필요 프로그램과 같이 설치됩니다. 기본 설치 위치는 C:\Program Files\LinkSquare Professional\LSAPI\iOS 폴더 입니다.

4 주요 파일

4.1 LinkSquareAPI.framework_v1.0.0_Universal.zip

LinkSquare API for iOS 의 Framework 압축 파일입니다. iOS 및 mac 의 Simulator 에서 실행 되는 Universal Framework 입니다. 개발 시 Simulator 와 실제 iOS 기기에서 동작합니다.

4.2 LinkSquareAPI.framework_v1.0.0_iOS.zip

LinkSquare API for iOS 의 Framework 압축 파일입니다. 실제 iOS 기기에서만 실행 되는 Framework 입니다. 개발 후 App store 에 App 을 올리기 위해 필요한 architecture (armv7, arm64) 만 지원합니다.

5 LinkSquare API for iOS 사용하기

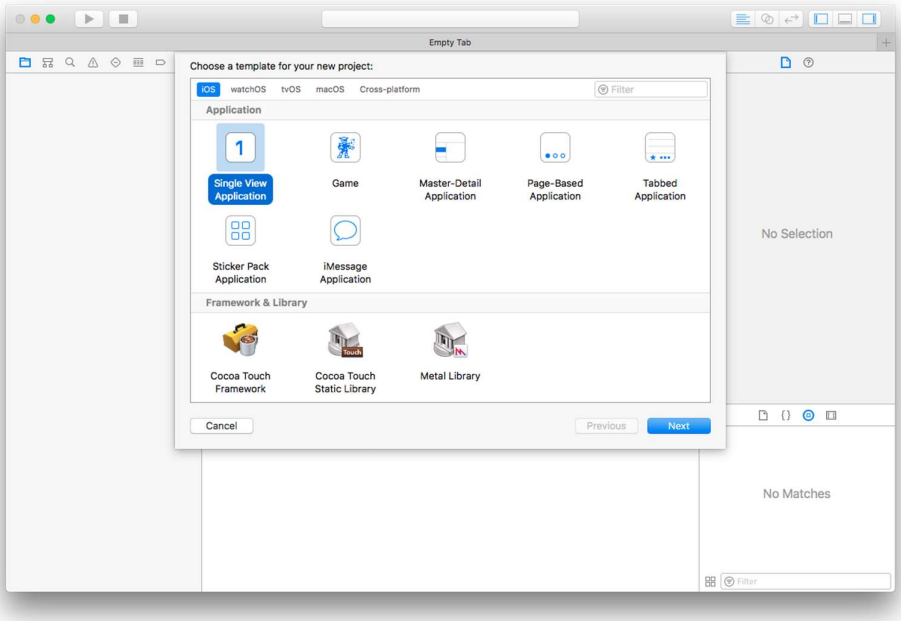
아래 내용은 처음 프로젝트 생성부터 시작하여 LinkSquare API for iOS 를 사용하기 위한 프로젝트 설정 방법을 설명합니다.

LinkSquare API 의 사용법은 예제 프로젝트를 참고하세요.

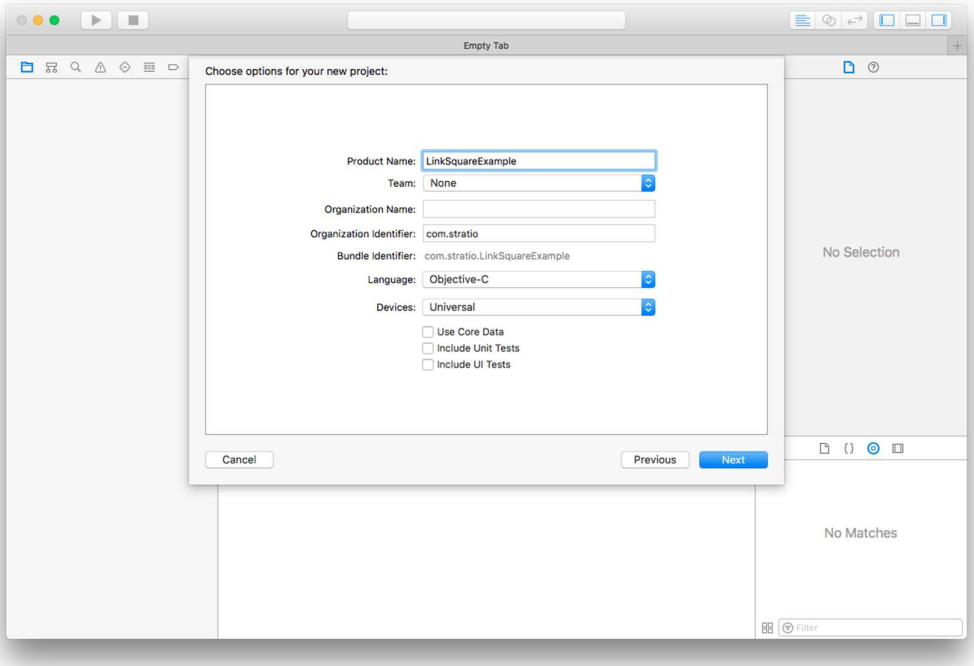
LinkSquare Professional 설치 시 Example 구성요소를 설치 하였으면 기본 설치 경로인 C:\Program Files\LinkSquare Professional\Examples\iOS\ 폴더에 예제 프로젝트의 압축파일이 있습니다. 해당 폴더의 LSAPI_iOS.zip 파일을 적당한 위치에 복사 한 후 압축을 해제한 다음 Xcode 에서 압축 해제한 폴더를 열면 아래 문서의 내용이 포함 되어 있는 예제 프로젝트를 확인 하실 수 있습니다. 설치된 예제 프로젝트의 빌드 및 실행은 5.4 항목을 참고하면 됩니다.

5.1 Project 생성

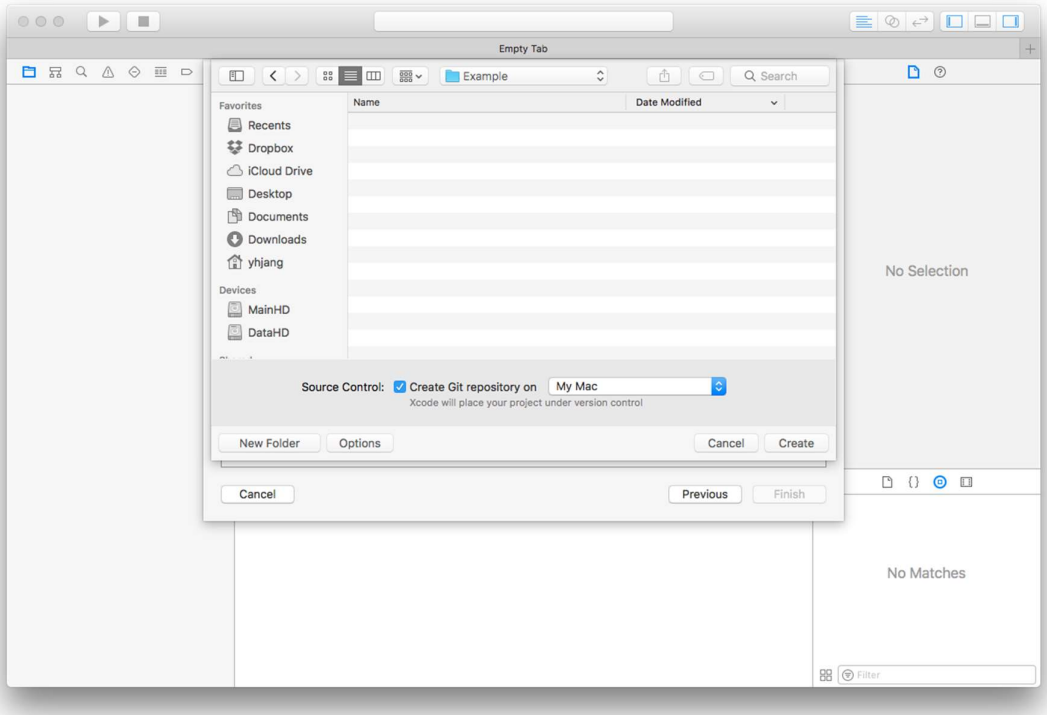
Xcode 를 실행하여 새 프로젝트를 생성합니다.



iOS 프로젝트 템플릿 중 Single View Application 을 선택합니다.



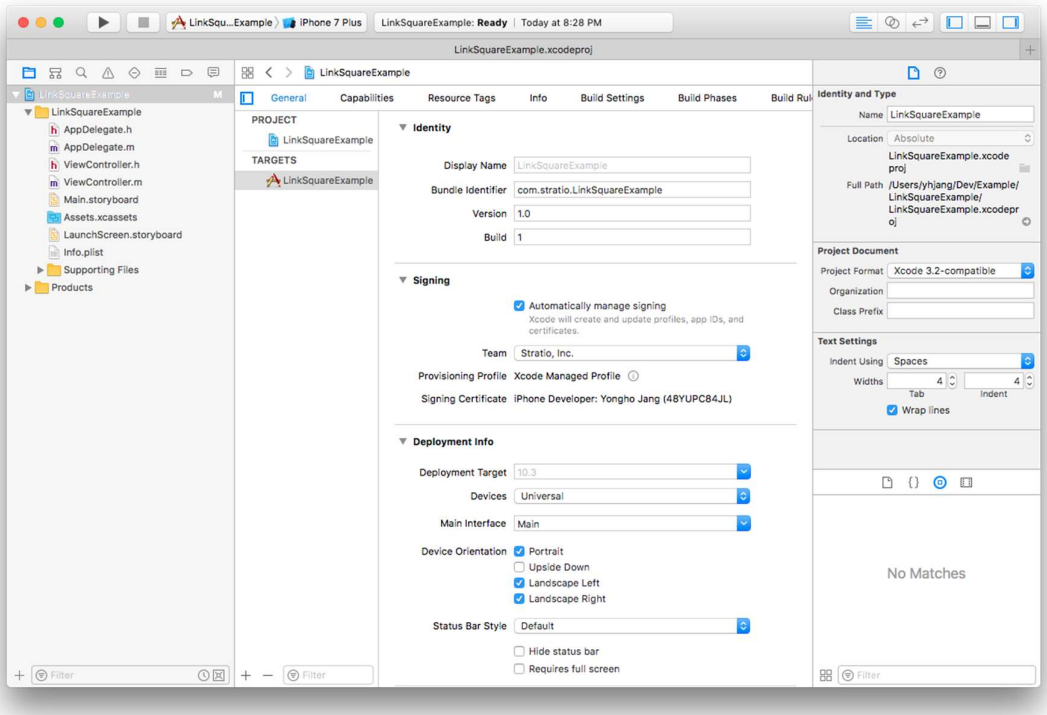
프로젝트 옵션을 설정 합니다. Product 이름과 Organization Name 등을 설정합니다. Next 버튼을 눌러 다음 화면으로 진행합니다. 본 예제 문서에서는 프로젝트 이름을 LinkSquareExample 로 설정하였습니다.



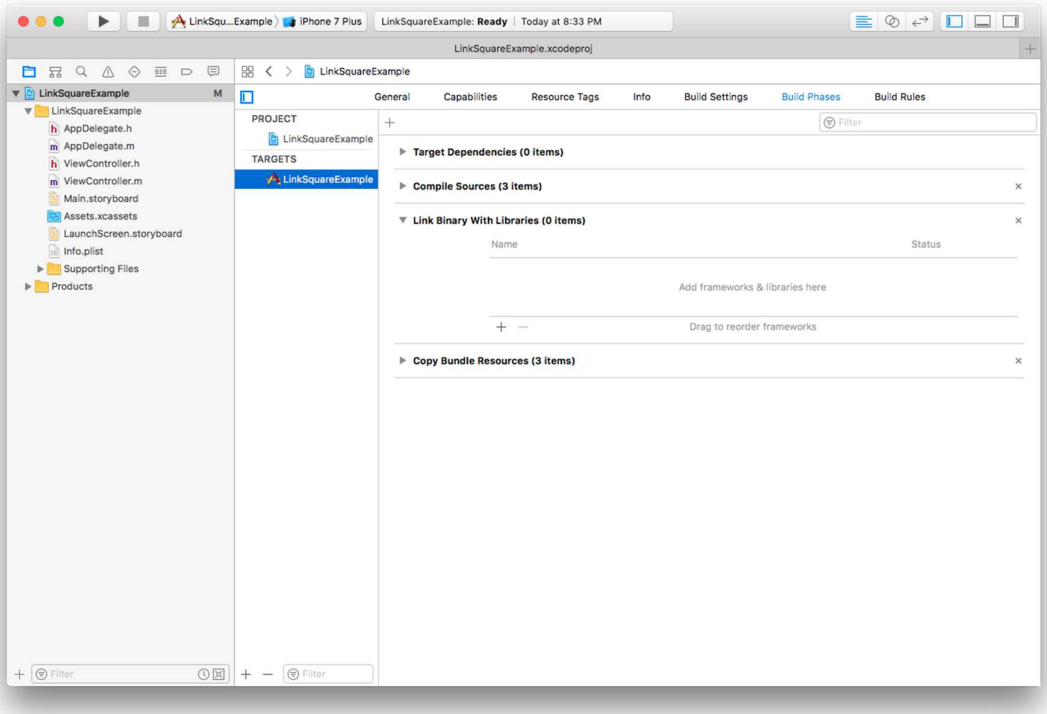
프로젝트를 저장할 위치를 선택합니다. 프로젝트 생성을 마쳤습니다.

5.2 LinkSquareAPI Framework 추가

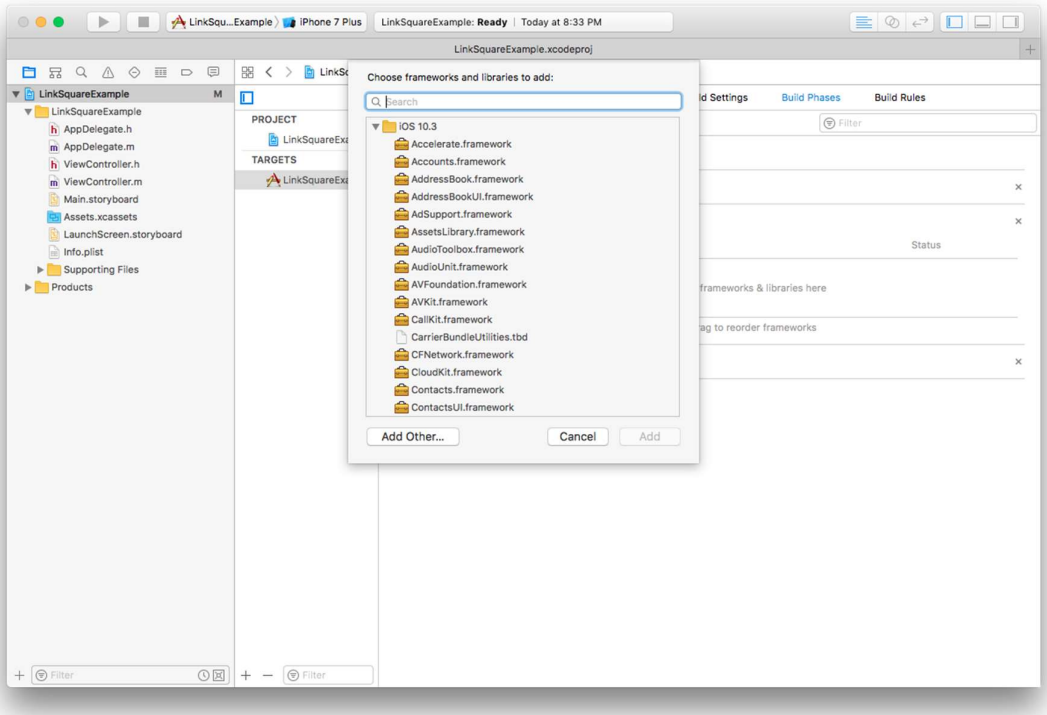
LinkSquare Professional 설치 경로의 LSAPI\iOS 폴더에 LinkSquareAPI.framework_v1.0.0_Universal.zip 파일이 있습니다. 압축을 해제하면 LinkSquareAPI.framework 이름의 Package 가 생성됩니다. 새로 만들어진 프로젝트에서 LinkSquareAPI Framework 를 사용 할 수 있도록 추가하는 작업이 필요합니다.



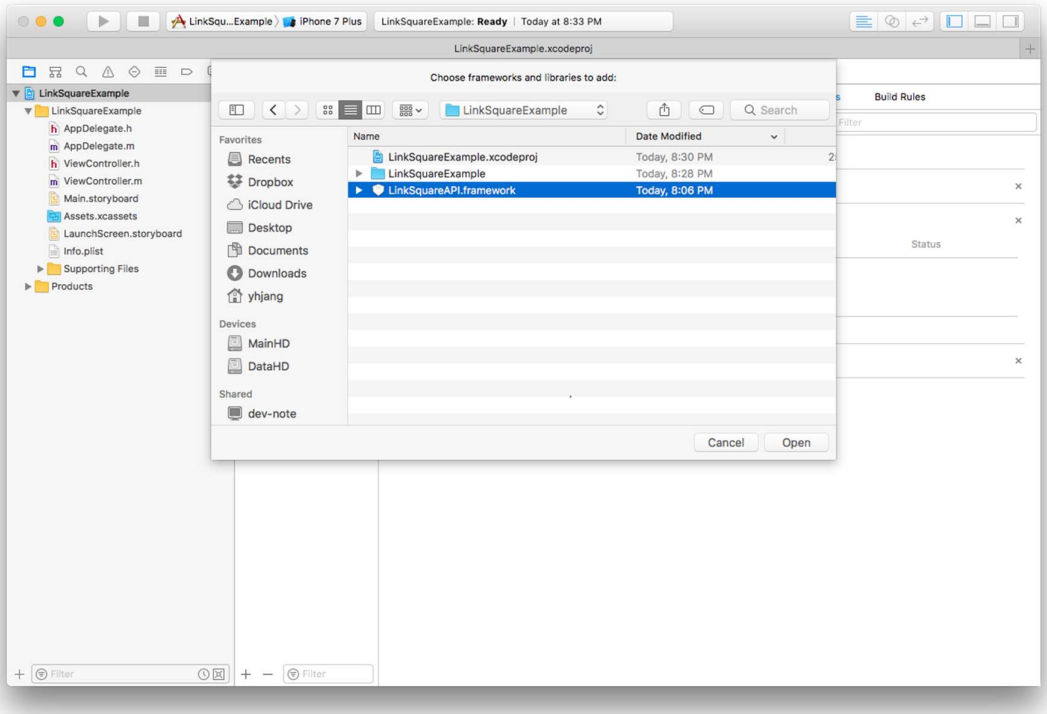
프로젝트에 LinkSquareAPI 를 추가 할 Target 을 선택합니다.

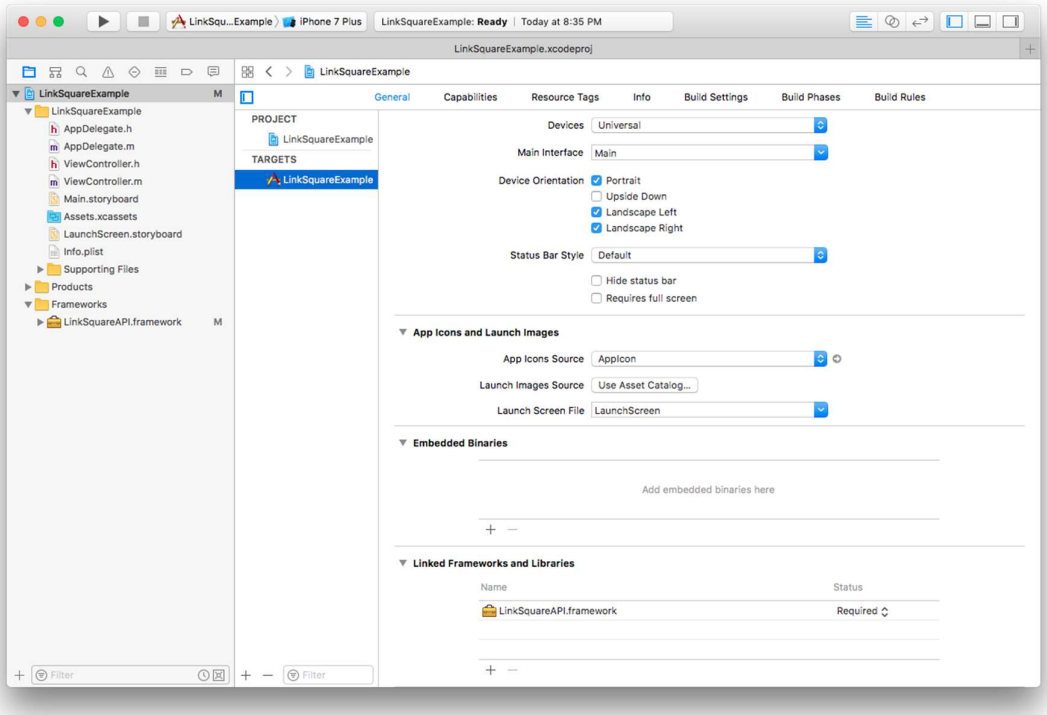


Build Phases 항목을 선택하고 아래의 Link Binary With Libraries 의 + 버튼을 누릅니다.

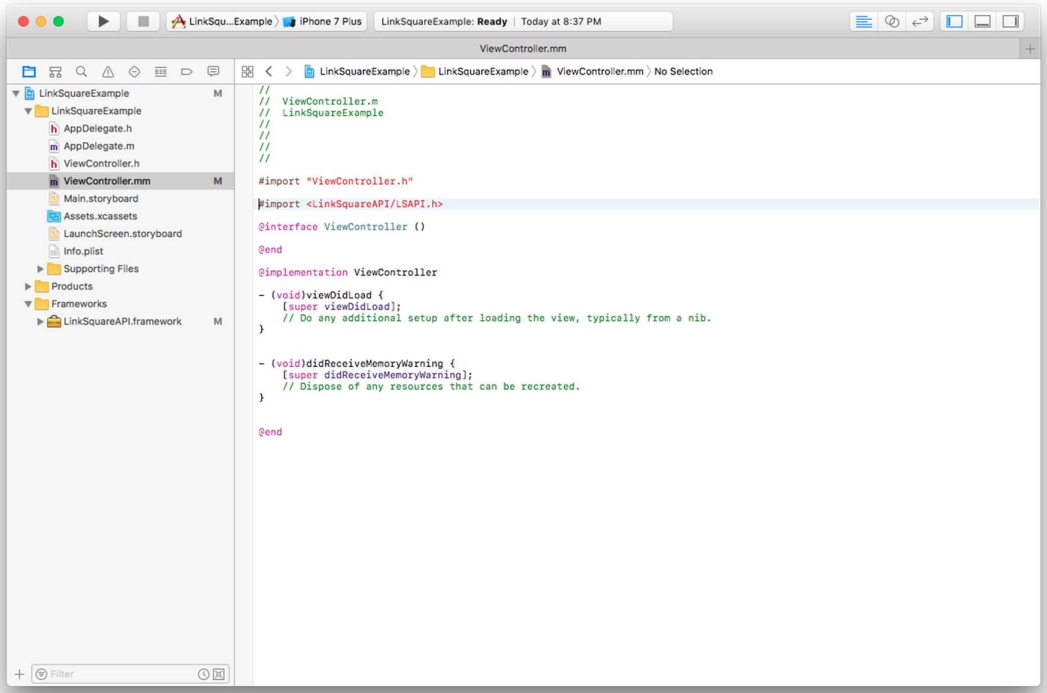


“Add Other…” 버튼을 눌러 LinkSquareAPI Framework 를 선택합니다.





General 항목을 선택하여 Embeded Binaries 의 + 버튼을 눌러 LinkSquareAPI Framework 가 빌드시 앱에 포함되도록 선택해 줍니다.



LinkSquare API 는 C++ 로 작성되어 있습니다. LinkSquare API 는 확장자가 ".mm"인 Objective-C++ 에서 사용할 수 있습니다. Framework 사용을 위해 #import <LinkSquareAPI/LSAPI.h> 로 헤더파일을 Import 합니다.

LinkSquare API 를 사용할 준비가 끝났습니다.

5.3 Example Project

LinkSquare API for iOS 사용법은 예제 프로젝트의 ViewController.mm 파일을 참고 하세요. 예제 프로젝트는 LinkSquare API 를 사용하여 LinkSquare 기기와 연결 하고 스캔을 진행하여 데이터를 얻는 샘플 코드입니다.

5.3.1 LinkSquare Event Callback 함수

LinkSquare API 에서 발생하는 Event 를 전달 받아 처리하기 위한 콜백 함수를 작성합니다. 함수의 원형을 LSAPI.h 파일에 정의 되어 있습니다.

```
typedef void(*DeviceEventCallbackFn)(void* userData, LSDeviceEventType type, int
value);
```

Callback 함수는 사용자가 LinkSquare 기기의 버튼이 눌렸을 때, LinkSquare 기기에서 일정 시간 이상 네트워크 응답을 못 받았을 때, LinkSquare 와 네트워크 연결이 끊겼을 때에 호출이 되며 상황에 따라 스캔을 진행하거나 사용자에게 적절한 알림을 주어야 합니다.

```
// LinkSquare API Callback
void LinkSquareEventCallback(void* userData, LSDeviceEventType type, int value)
{
    // casting userData to ViewController
    ViewController* viewController = (__bridge ViewController*)userData;

    if (type == LSDeviceEventType::ButtonEvent)
    {
        [viewController addLog:@"Scan Button Pressed.\n"];

        // Scan
        vector<LSFRAME*> frames;
        int ret = LSAPI::Scan(g_hLS, 3, 3, frames);
        if (ret != 1) {
            auto errorMsg = LSAPI::GetLSError();
            [viewController addLog:[NSString stringWithFormat:@"Scan Failed\n %@",
[NSString stringWithUTF8String:errorMsg]]];
            LSAPI::Close(g_hLS);
            return;
        }

        // Print scan data
        for (size_t i = 0; i < frames.size(); i++) {
            auto frame = frames[i];
            NSString* result = [NSString stringWithFormat:@"Frame No: %d, Length: %d, Light
Source: %d\n", frame->frame_no, frame->length, frame->light_source];
            [viewController addLog:result];
        }
    }
    else if (type == LSDeviceEventType::TimeoutEvent)
    {
        [viewController addLog:@"Device Network Timeout.\n"];
        LSAPI::Close(g_hLS);
    }
    else if (type == LSDeviceEventType::NetworkCloseEvent)
    {
        [viewController addLog:@"Device Network Closed.\n"];
        LSAPI::Close(g_hLS);
    }
}
```

DeviceEventCallbackFn 함수와 LSDeviceEventType 에 대한 제세한 설명은 LinkSquare API Reference 문서를 참조하세요.

5.3.2 LinkSquare API Initialize

LinkSquare API 사용하기 위해 제일 처음 Initialize 함수를 호출하여 초기화 작업을 진행합니다.

```
// LSAPI Initialize
g_hLS = LSAPI::Initialize();
if (g_hLS == NULL) {
    auto errorMsg = LSAPI::GetLSError();
    [_resultTextView insertText:[NSString stringWithUTF8String:errorMsg]];
    return;
}
```

Initialize 에 실패하게 되면 hLinkSquare 변수에 NULL 값이 반환되며 자세한 오류 메시지는 GetLSError 함수로 확인 할 수 있습니다.

5.3.3 Event Callback 함수 등록

작성한 Callback 함수를 LinkSquare API 에 등록합니다. 등록할 때 userData 에 전달되는 값은 Callback 호출 시 전달 됩니다. 예제에서는 Objective-C 객체를 전달하기 위해 __bridge 키워드를 사용합니다.

```
// Register Callback callback function
void* userData = (__bridge void *)self; // cast userData to (void *)
LSAPI::RegisterEventCallback(g_hLS, LinkSquareEventCallback, userData);
```

5.3.4 LinkSquare 연결

LinkSquare 기기에 연결 하기 위해 Connect 함수를 호출합니다. LinkSquare 기기의 IP 와 Port 번호를 입력합니다. 기본 Port 는 18630 입니다.

Connect 에 성공하면 result 변수에 1 을 반환 받습니다. Connect 에 실패 시 1 이외의 값을 받으며 자세한 오류 메시지는 GetLSError 로 확인 할 수 있습니다.

LinkSquare 기기와의 연결 상태는 IsConnected 함수로 확인 할 수 있습니다.

```
// Connect to LinkSqaure
int connectResult = LSAPI::Connect(g_hLS, DEFAULT_IP, DEFAULT_PORT);

if (connectResult == 1) {
    [_resultTextView insertText:@"Connected\n"];

    LSDeviceInfo deviceInfo;
    int result = LSAPI::GetDeviceInfo(g_hLS, &deviceInfo);
    if (result == 1)
```

```
    {
        NSString* alias = [NSString stringWithUTF8String:deviceInfo.Alias];
        NSString* SWVer = [NSString stringWithUTF8String:deviceInfo.SWVersion];
        NSString* HWVer = [NSString stringWithUTF8String:deviceInfo.HWVersion];
        NSString* DeviceID = [NSString stringWithUTF8String:deviceInfo.DeviceID];
        NSString* OPMode = [NSString stringWithUTF8String:deviceInfo.OPMode];
    }
}
else {
    auto errorMsg = LSAPI::GetLSError();
    [_resultTextView insertText:[NSString stringWithFormat:@"Connection Failed\n %@",
[NSString stringWithUTF8String:errorMsg]]];
}
```

연결이 성공하면 GetDeviceInfo() 메소드로 연결된 LinkSquare 기기의 세부정보를 알아 올 수 있습니다.

자세한 내용은 API Reference 문서를 참고하세요.

5.3.5 Scan

```
- (IBAction)doScan:(id)sender {
    [_resultTextView insertText:@"Scanning...\n"];

    // Number of LED Frames
    int ledFrames = 3;

    // Number of Bulb Frames
    int bulbFrames = 3;

    vector<LSFRAME*> resultFrames;
    int scanResult = LSAPI::Scan(g_hLS, ledFrames, bulbFrames, resultFrames);

    if (scanResult == 1) {
        [_resultTextView insertText:@"Scan Complete\nScan Results:\n"];
        NSString *scanResult = [self parseScanResult:resultFrames];
        [_resultTextView insertText:scanResult];
    }
    else {
        auto errorMsg = LSAPI::GetLSError();
        [_resultTextView insertText:[NSString stringWithFormat:@"Scan Failed\n %@",
[NSString stringWithUTF8String:errorMsg]]];
    }
}
```

LinkSquare 기기와 연결 후 Scan 함수를 이용하여 스캔 데이터를 얻어 올 수 있습니다.

광원에 따른 프레임 수를 설정하여 필요한 수의 데이터를 얻어 옵니다.

스캔 데이터는 LSFRAME 구조체로 전달 되며 std::vector<LSFRAME*> 타입의 참조 파라메터로 전달됩니다.

스캔 데이터는 다음 Scan 함수가 호출 되거나 Close 함수가 호출 되면 메모리에서 제거됩니다. 필요시 사용자가 데이터를 복사하여 사용하여야 합니다.

5.3.6 LSFRAME 구조체 데이터 사용

Scan 함수 호출로 얻은 std::vector<LSFRAME*>타입의 데이터는 일반적인 vector 컨테이너 사용법과 같이 사용할 수 있습니다.

```
NSString *result = [[NSString alloc] init];

for(std::vector<LSFRAME *>::iterator it = frames.begin(); it != frames.end(); ++it) {
    int f_no = (*it)->frame_no;
    int length = (*it)->length;
    unsigned char light = (*it)->light_source;
    result = [result stringByAppendingString:[NSString stringWithFormat:@"Frame No: %d,
Length: %d, Light Source: %d\n", f_no, length, light]];
}
```

For 문을 이용하여 모든 LSFRAME 의 length, light_source 정보를 출력 할 수 있습니다. 실제 프레임 데이터는 raw_data 혹은 data 멤버로 접근 할 수 있습니다. 자세한 사항은 LinkSquare API Reference 의 LSFRAME 구조체 항목 및 Scan 함수 항목을 참고하세요.

5.3.7 Close

프로그램 종료 전 LinkSquare 기기와의 연결을 종료하고 LinkSquare API 의 리소스 해제를 처기하기 위해 호출 합니다.

```
LSAPI::Close(g_hLS);
[_resultTextView insertText:@"Disconnected\n"];
```

5.4 빌드 및 실행

5.4.1 빌드

Xcode 의 Product 메뉴에서 Build 항목을 선택하여 프로젝트를 빌드 합니다.

5.4.2 LinkSquare 기기 연결

LinkSquare 기기는 Wi-Fi 를 사용하여 PC 와 연결됩니다. 이를 위해 두 가지 모드를 제공하고 있습니다.

IoT 모드는 사용중인 라우터에 LinkSquare 기기가 접속하여 같은 네트워크를 사용하는 PC 와 연결되는 모드입니다. AP 모드는 라우터가 없는 경우 LinkSquare 기기가 제공하는 AP 에 PC 가 직접 연결되는 모드입니다.

LinkSquare 기기의 전원을 겨면 버튼 LED 에 표시되는 색상에 따라 현재 작동 중인 모드를 확인 할 수 있습니다. IoT 모드일 경우 파란색으로 점멸합니다. AP 모드로 동작 중인 경우 초록색으로 점멸합니다.

두 모드 사이의 전환은 버튼을 누르면 가능합니다. 전원이 들어온 상태에서 LinkSquare 기기의 버튼은 한 번 누르면 다른 모드로 전환이 이루어 집니다

IoT 모드로 연결

IoT 모드는 LinkSquare 기기의 전원을 켠 다음 버튼의 LED 가 파란색으로 점멸합니다. 초록색일 경우 AP 모드로 작동 중입니다. 버튼을 한번 누르면 IoT 모드로 전환됩니다.

IoT 모드로 연결할 경우 LinkSquare 기기에 접속할 AP 정보를 설정해 주어야합니다. SetWLanInfo API 를 호출하여 설정하는 방법과 LS Config 프로그램을 이용하는 방법이 있습니다.

IoT 모드에서는 접속하려는 LinkSquare IP 를 개발자가 확인해야합니다. LS Config 를 사용하여 확인 하거나 라우터의 설정 페이지에서 확인 할 수 있습니다.

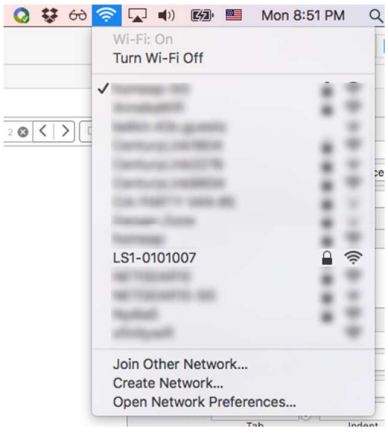
AP 모드로 연결

- LinkSquare 기기 전원 켜기

먼저 LinkSquare 기기의 버튼을 2 초동안 누르면 LinkSquare 기기의 전원이 켜집니다. 전원이 켜지면 LinkSquare 기기 버튼의 LED 가 초록색으로 점멸합니다.

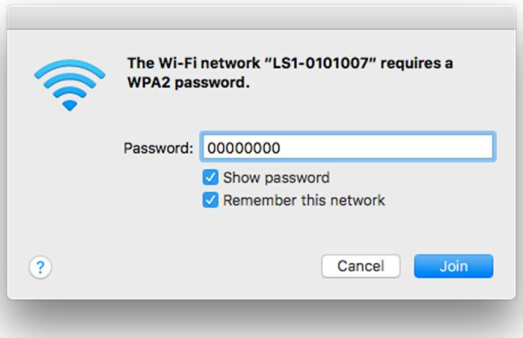
- PC 에서 네트워크 연결

Mac 기기 화면의 오른쪽 위 영역에서 Wi-Fi 네트워크 아이콘을 마우스 왼쪽 버튼으로 클릭하면 현재 Mac PC 에서 연결 가능한 네트워크 목록이 나타납니다.



AP 모드로 작동 중인 LinkSquare 는 PC 가 접속 할 수 있도록 자체의 AP 를 작동 중입니다. LinkSquare 의 AP 는 “LS1-“ 로 시작하는 이름으로 초기값은 LS1- 다음에 7 자리의 숫자가 있는 형태입니다. (예: LS1-0101001) PC 의 Wi-Fi 설정에서 해당 AP 가 표시되는 지 확인하고 LinkSquare AP 에 접속 합니다. Mac

장비에서 연결 가능한 네트워크 목록에서 LinkSquare 기기의 네트워크 이름을 찾아 선택한 다음 연결 버튼을 누릅니다.



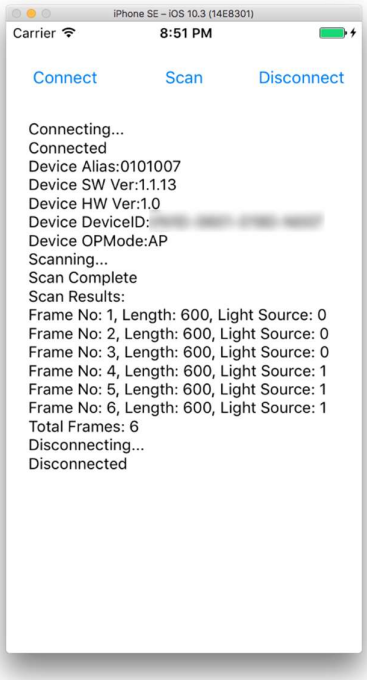
처음 네트워크 연결에는 암호를 입력 해야 합니다. 초기 암호는 "`00000000`" (숫자 0, 8 개)입니다.

5.4.3 실행

Product 메뉴의 Run 항목을 눌러 프로그램을 실행합니다. 테스트를 위해 LinkSquare 기기가 연결 가능해야 합니다.

5.4.4 화면 확인

성공적으로 실행되면 Connect 버튼을 눌러 LinkSquare 와 연결되고 Scan 을 누르면 스캔이 진행됩니다.



6 FAQ

6.1 “dyld: Library not loaded: @rpath/LinkSquareAPI.framework/LinkSquareAPI” 오류가 발생 할 때

LinkSquareAPI.framework 가 앱에 포함이 안되었을 때 발생합니다. Project 의 Target 설정의 Genernal 항목에서 아래쪽의 Embedded Binaries 와 Linked Frameworks and Libraries 부분에 LinkSquareAPI.framework 가 포함되어 있는지 확인하세요.

6.2 App Store 에 업로드시 Unsupported Architectures 로 실패할 때.

LinkSquareAPI.framework 는 개발 및 테스트를 위해 Simulator 에서 실행 될 수 있도록 i386 및 x86_64 아키텍쳐를 지원합니다. App Store 에 업로드 시에는 armv7, arm64 만 지원하는 LinkSquareAPI.framework_v1.0.0_iOS.zip 파일의 LinkSquareAPI.framework 를 사용하세요.

6.3 LinkSquare 기기와 연결이 안될 때

Mac 또는 iOS 기기의 Wi-Fi 가 LinkSquare 기기와 연결되어 있는지 확인합니다.연결된 네트워크의 이름이 “LS1-”로 시작하는지 확인 하세요.

LinkSquare 기기의 전원이 켜져 있는지 확인합니다. 전원이 켜져 있을 경우 LinkSquare 기기의 버튼 주변에 초록색 또는 파란색의 LED 가 점멸합니다.

LinkSquare 기기 연결 항목을 참고하세요.

7 기술 지원 안내

- LinkSquare Professional 기술 지원

LinkSquare Professional 제품에 관련 문의 사항은 아래 연락처로 문의 하십시오.

전화: 02 - 6205 - 7456

전자메일: contact@stratiotechnology.com

웹 사이트: http://linksquare.io/contact.html