SDK Device Collector iOS 2.5 Guide

Contents

Introduction 4
  SDK Device Collector iOS 4
Using the SDK Device Collector iOS 5
  Adding the SDK in a Project 5
  Configuring a Project to Link with SDK library 6
  Referencing Documentation 9
  Implementing the SDK Inside Your Application 9
  Calling collect() 11
  Error Codes 12
  Reference Implementation Application 12
  iOS Implementation Q and A 13

Introduction

A Software Development Kit, also called a “devkit” or SDK, is a set of development tools that allows software engineers to create customized applications for a particular software package, software framework, hardware platform, or operating system. This allows developers to create applications specific to their business needs that will interact with the product or products developed by the SDK creators.

SDK Device Collector iOS

The SDK Device Collector iOS provides a static library which can be linked with an iOS application to perform Device Collection interaction with Kount for native iOS applications on Apple’s iPhone and iPad platforms. The SDK includes a collection of development tools that allow you to build your own customized applications in order to submit information to the Risk Inquiry System (RIS) server. The static library is compiled for iOS version 4.0 and newer. It is a Mach-O universal binary for architectures armv6, armv7, armv7s and i386.

Additional information regarding SDKs can be found in the Kount RIS Software Development Kit Guide. Also, please consult the **README** file for the specifications of the SDK.

NOTE: For this release and iOS7 applications, merchants should disable mac address collection as Apple has effectively short circuited this capability. To turn mac address collection off in Version 2.0 of the iOS SDK, the merchant will need to pass the DC_COLLECTOR_DEVICE_ID flag within an array to the skip function of the SDK. An example of how to do this is in the Reference implementation. It looks like this:

```
NSMutableArray *skipList = [[NSMutableArray alloc]init];
[skipList addObject:DC_COLLECTOR_DEVICE_ID];
[self.deviceCollector setSkipList:skipList];
```

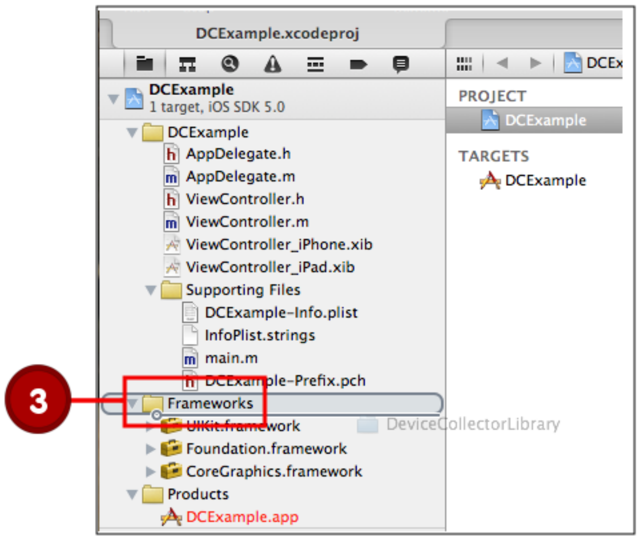
Using the SDK Device Collector iOS

The following information will assist you in acquiring, adding, and and configuring the SDK in a project.

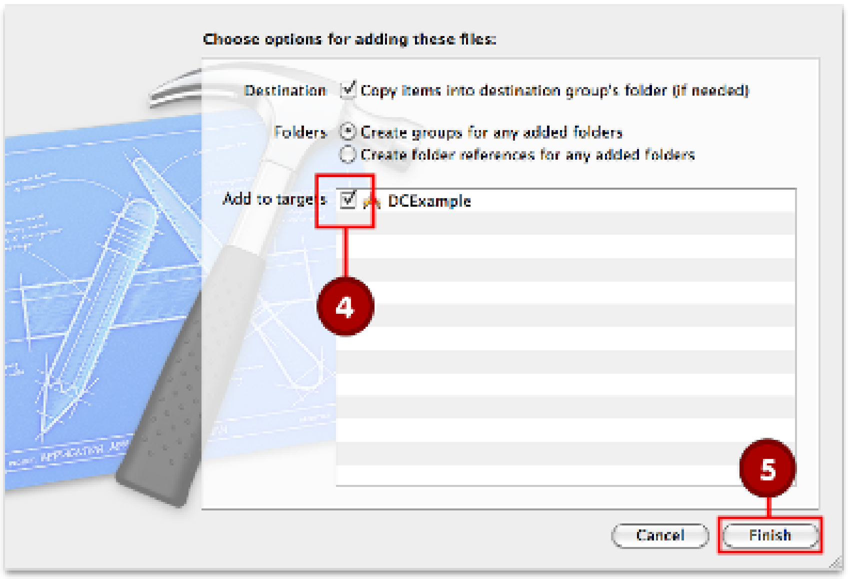
Adding the SDK in a Project

To use the Device Collector with native iOS applications, perform the following steps:

1. Contact a Merchant Services representative to request the latest version of the SDK.
2. Unpack the SDK zip file by double clicking the archive.
3. Drag and drop the **DeviceCollectorLibrary** folder on to the **Frameworks** section of your project.



4. Select the check box next to your target(s).
5. Click **Finish**.


Choose options for adding these files:

Destination ✓ Copy items into destination group's folder (if needed)

Folders [✓ Create groups for any added folders] Create folder references for any added folders

Add to targets ✓ DCExample

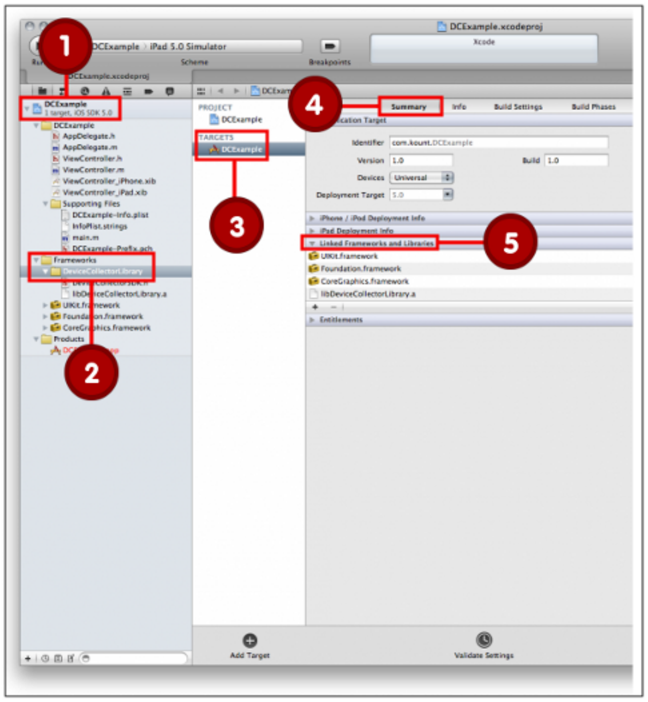
Configuring a Project to Link with SDK library

To configure your project to link with the SDK library, the following frameworks and libraries are required:

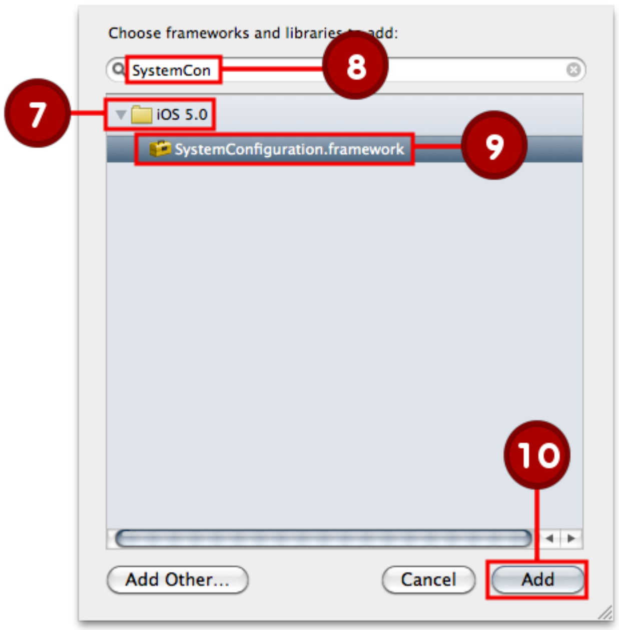
- libDeviceCollectorLibrary.a
- UIKit.framework
- SystemConfiguration.framework
- CoreLocation.framework

In the project, perform the following:

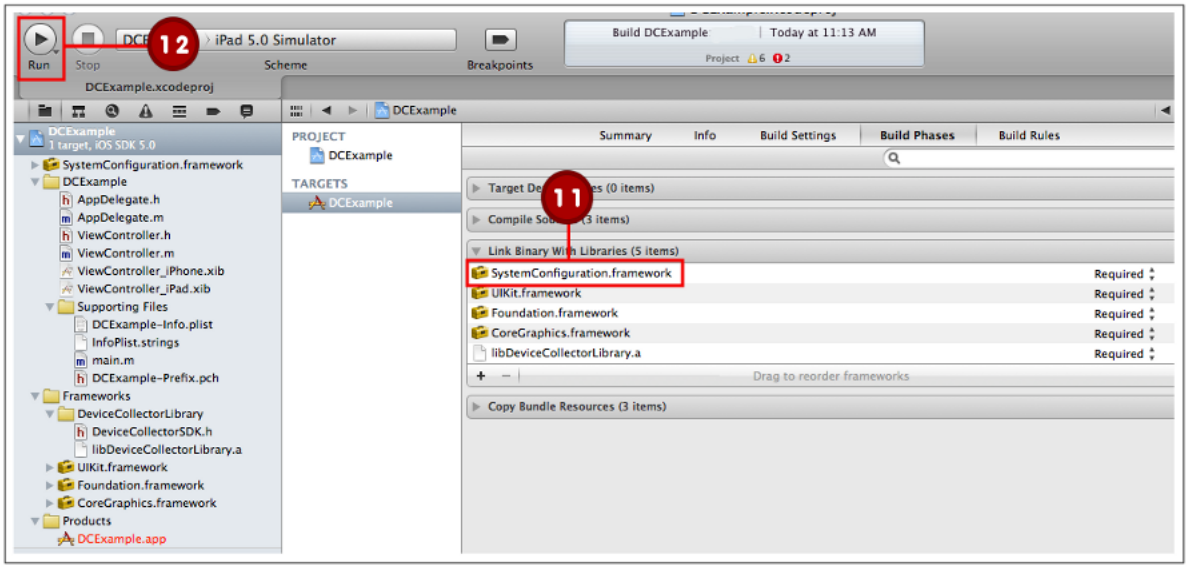
1. In the project menu, select the desired project and if necessary, expand it.
2. Under **Frameworks**, select **DeviceCollectorLibrary**.
3. Under **Targets**, select the desired target.
4. Select the **Summary** tab.
5. Expand **Linked Frameworks and Libraries** to verify that everything required is present.



6. If any of the required frameworks or libraries are not present, click the “plus” sign to search for and add the missing elements.
7. Expand the desired folder if necessary to produce the full list.
8. Begin typing the name of the desired framework in the search field to filter the list.
9. When the desired framework appears in the list, select it.
10. Click **Add**.


Choose frameworks and libraries to add:
SystemCon
iOS 5.0
SystemConfiguration.framework
Add Other...
Cancel
Add
7
8
9
10

11. Verify that the required framework has been added.
12. Click **Run**.



Referencing Documentation

The following documents are referenced in this guide. You should understand these documents before proceeding:

- **Technical Specifications Guide: Data Collector and Risk Inquiry System (RIS)** - This guide covers in detail information about the Data Collector and RIS. Even if the merchant is not implementing a checkout page within a web browser, the Merchant URL Image with the 302 redirect to the Kount Server URL is still required, and should be setup prior to implementing the Mobile SDK Library. This document provides the information necessary to setup both the Data Collector Merchant URL and the RIS request used to evaluate the purchase request.
- **The iOS API:** (`https://developer.apple.com/devcenter/ios/index.action`)

Implementing the SDK Inside Your Application

Implementing the SDK inside your application consists of two basic steps:

1. Creation and Configuration
2. Calling the collect() method

This starts the collection process.

The **Creation** and **Configuration** process involves creating the DeviceCollector object, and setting up the object with the following values:

- **Merchant ID** - This is provided to you by Kount.
- **Collector URL** - This is your Data Collector URL the 302 redirects to the Data Collector (something like : https://mysite.com/logo.htm). For more information on the Collector URL, see the Data Collector documentation.
- **The DeviceCollectorSDKDelegate implementation** - To listen for status changes by the collector. The SDK collects data in the background after collect() is called. To learn the state of the collector, pass in an implementation of this delegate to get updates. A delegate is optional and the collector operates properly if a delegate is not set.

When you want the collector to gather information about the device, once the DeviceCollector is created and configured, call the collect() method and pass in the sessionId . The collect() method spawns a background thread to collect device information. The collect() method should not be called until a transaction is considered "imminent". For examples around what "imminent" means, see the section on using the collect() method.

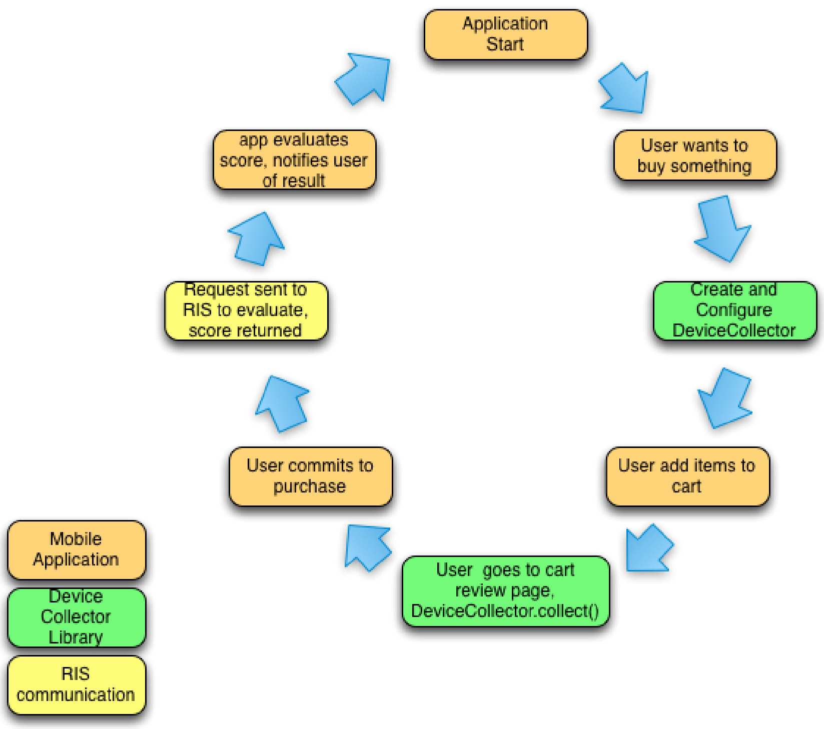
When the DeviceCollector collect process starts, it fires the onCollectorStart() method of the delegate if a delegate exists.

Information gathered by the DeviceCollector is only germane to the device and its environment and gathers no information about the user.

Once collection completes, the onSuccess() method fires on an implemented delegate. If the DeviceCollector failed, the exception onError(int, NError) fires. Succeed or fail, the collector process completes and no further action is taken.

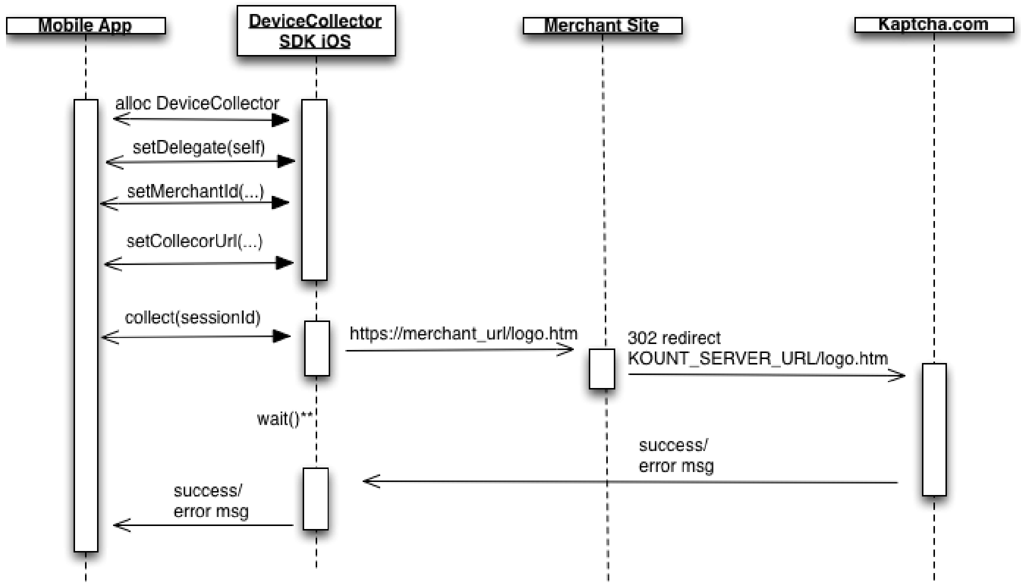
The step-by-step process of adding the SDK to your application may appear as follows:

- Implement a call to the library at the appropriate point in your application.
- Import DeviceCollector header
- Add DeviceCollectorSDKDelegate protocol to class declaration
- Implement DeviceCollectorSDKDelegate protocol methods in your class
- Instantiate collector
- Register your object to receive protocol notifications
- Configure collector (merchantId, collectorUrl)
- When close to "checking out" call [collector collect:sessionId]


Application Start
User wants to buy something
Create and Configure DeviceCollector
User add items to cart
User goes to cart review page, DeviceCollector.collect()
User commits to purchase
Request sent to RIS to evaluate, score returned
app evaluates score, notifies user of result
Mobile Application
Device Collector Library
RIS communication

NOTE: The session id used in the DeviceCollector must be the same that is passed to the RIS call when the purchase is being evaluated.

Here is a sequence diagram showing the actions between the application, the DeviceCollector Library, the merchant site, and the Kount site:



**wait times out after a period of time to prevent runaway threads

Calling collect()

It is important to collect the most up-to-date information possible just before a payment method is used. In the realm of shopping carts on a web site, this is normally at the checkout step. On the checkout page, a merchant displays order information the user would like to submit and also displays and gathers last minute information like method of payment or billing/shipping information. Work flow models differ, but generally there is a page that allows the user to confirm their purchase prior to the transaction submission to the processing server. Our suggestion is to execute the collect() method in this review step ("prior to" or "as" the review step is being displayed).

The collection process starts when you call the collect(sessionId) method of the DeviceCollector. This process runs asynchronously in the background and should not stop or interrupt your application.

The collect() call should only happen once per transaction.

If your application resets (i.e. due to an orientation change) you should keep track of whether or not you have already called collect(). The collector will notify your listener, if implemented, of the changes in status as things move along, which you can react to or ignore.

The calling application does not need to wait for the completion of the DeviceCollector. If the DeviceCollector does not complete prior to a call to the Risk Inquiry System (RIS) for the same transaction (session ID), then the DeviceCollector information will not be used. This prevents any "gating" effects to the application from the collector.

Error Codes

The Error codes for the DeviceCollector library are listed in the header file:

- #define DC_ERR_NSERROR 1 // NSError trapped. See error for details
- #define DC_ERR_NONETWORK 2 // Network access not available
- #define DC_ERR_INVALID_URL 3 // Invalid collector URL
- #define DC_ERR_INVALID_MERCHANT 4 // Invalid Merchant Id
- #define DC_ERR_INVALID_SESSION 5 // Invalid Session Id
- #define DC_ERR_VALIDATION_FAILURE 6 // Device collection failed

Reference Implementation Application

Kount has produced a bare-bones Reference Implementation application that shows the fundamentals needed to use the SDK. It is included in the SDK. The reference implementation is not meant to be used in a production environment. For assistance with production code, please contact your Merchant Services representative.

The user interface (UI) for **iOS 6.x** and **7.0** have a different “look and feel.” Reference Implementations for both iOS versions are included in this SDK for merchants to see examples of how to implement for either version. These examples **are not** meant for production use and can be ignored if examples are not needed by the merchant.

iOS Implementation Q and A

QUESTION: Why do I get an ErrorCode 6 on iOS when attempting to collect data?

ANSWER: Have you verified that your **Merchant ID** and **CollectorURLs** are set correctly? The **Merchant ID** must be the 6 digit number supplied to you from Kount, and the **Collector URL** (with HTTPS) must be a fully qualified URL that points to a resource, not just a server (i.e. **https://mysite.com/logo.htm**, not just **https://mysite.com**) Also, have you setup your Data Collector URL within your framework to 302 redirect to the KOUNT_SERVER_URL (i.e. **from https://mysite.com/logo.htm** -> **https://KOUNT_SERVER/logo.htm**) If you turn debugging on, you should see it send a message to the logs showing the final URL it is trying to use. If your input is:

- Merchant ID: **123456**
- Collector URL(HTTPS) : **https://mysite.com/logo.htm**
- Session ID: **ABCD123456789ABCDFFEEDD123456732**

Then the final final URL posted to the logs (in debug mode) should look something like this:

https://mysite.com/logo.htm?m=123456&s=ABCD123456789ABCDFFEEDD123456732

If this still does not work, try copying and pasting this URL to a web browser and ensure the logo you have provided merchant services displays on the return page. If you do not get your logo to display:

1. Check to ensure you correctly setup the Data Collector redirect on your servers

   (i.e. **from https://mysite.com/logo.htm** -> **https://KOUNT_SERVER/logo.htm**)

2. If your redirect is working, the there may be an issue with your account. Contact Merchant Services to troubleshoot. Please include the URL you are using above.

Kount®