Sample Code

Testing and validating ad impression signatures and postbacks for SKAdNetwork

Validate your ad impressions and test your postbacks by creating unit tests using the StoreKit Test framework.

Download

iOS 16.4+ | iPadOS 16.4+ | Mac Catalyst 16.4+ | Xcode 14.3+

Overview

This sample code project provides examples of how to use the StoreKit Test framework to test individual units of an `SKAdNetwork` implementation. The sample code runs each type of test that `SKAdTestSession` makes possible.

First, it creates and validates ad impressions. In a typical ad attribution flow, the ad network cryptographically signs an ad impression. This unit test checks the signature and validates other aspects of the impression.

Next, the sample code creates a test postback and updates the conversion value. In a typical ad attribution flow, the system creates a postback after a user installs and opens an advertised app. In the unit test, you can see and control every aspect of the test postback, so you can test your app's conversion values and decide which postback to annotate as the winner.

Finally, the sample code sends and receives the test postbacks. In a typical ad attribution flow, the system sends a postback after a timer expires. In the unit test, you control when to send the postback so that you can test that your server receives the postback.

Configure the sample code project

To run the unit tests in your environment with your settings, you'll need the following:

- Your ad network identifier. Configure the value in your `Info.plist` as described in Configuring a Source App. In the sample code, use the same value everywhere that the ad network identifier is required.
- A source app identifier. In the testing environment, this value is always `0`.
- A cryptographic key pair, consisting of a public key and a private key, created with the Elliptic Curve Digital Signature Algorithm (ECDSA) with a `prime256v1` curve. In the sample code, use the private key when you generate a signature, and use the public key when you call `validateImpression`.
  - Set the `publicKey` variable in the sample code project to your public key.
  - For more information on generating a private and public key pair for signing ads, visit Registering an ad network.
- A local or remote server to listen for postbacks. Specify your server's URL when you create test postbacks as shown in the sample code's `testAddingPostbacks()` method.
- A device running iOS 16.4 or later. iOS Simulator doesn't support `SKAdNetwork`.

Create and validate the ad impression

In the `testImpressionValidity()` method, the unit test creates an instance of `SKAdImpression` to represent a view-through ad.

All of the properties are required. In the testing environment, the `sourceAppStoreItemIdentifier` is always `0`.

```
// Form the SKAdImpression instance and configure it.
let impression = SKAdImpression()
impression.version = "4.0"
impression.adNetworkIdentifier = "com.apple.test-1"
impression.sourceIdentifier = 3120
impression.advertisedAppStoreItemIdentifier = 525_463_029
impression.adImpressionIdentifier = "b7c9da2b-15c7-4f3b-9326-135f9630033d"
impression.sourceAppStoreItemIdentifier = 0
impression.timestamp = 1_676_057_605_705
impression.signature = "MEQCIAtBBiadCFlMOEOh3K43xyKaU1/sj/CtgDOB+Wm7J+29AiBDfreX67mn
```

The unit test then calls the `validateImpression` method on `SKAdTestSession` and checks for errors.

If `validateImpression` returns an error, the unit test fails, and the error contains further information. If `validateImpression` doesn't return an error, the unit test succeeds.

In the `testImpressionParametersValidity` method, the unit test creates a dictionary containing the parameters for a StoreKit-rendered ad and validates it using the `validateImpressionWithParameters` method. For more information about StoreKit-rendered ads, visit Signing and providing ads. The `testWebAdImpressionPayloadValidity` method creates and validates a web ad impression payload. For more information on web ads, visit Get a Signed Web Ad Impression Payload.

> **Note**
>
> For signature verification to succeed, provide the validate impression methods with the public key of the key pair used to generate the cryptographic signature for the ad impression.

Create the test postbacks

The unit test creates three winning postbacks in the `testAddingPostbacks` method using the `SKAdTestPostback` class. The fields of a postback are specific to each version of the `SKAdNetwork` API. This class provides an initializer that specifies the postback fields and the SKAd`Network` API version.

For ad postbacks using SKAd`Network` version 4.0 or later, test multiple conversion windows by creating an array of three postbacks corresponding to the three conversion windows using the `winningPostbacks` static method on SKAd`TestPostback`. Create a single winning or nonwinning postback using the initializer with the `sourceIdentifier` parameter. For more information on multiple conversion windows, visit Receiving postbacks in multiple conversion windows.

```
guard let testPostbacks = SKAdTestPostback.winningPostbacks(withVersion: .version4_0
                                                            adNetworkIdentifier: "cc
                                                            sourceIdentifier: "3120"
                                                            appStoreItemIdentifier:
                                                            sourceAppStoreItemIdenti
                                                            sourceDomain: nil,
                                                            fidelityType: 1,
                                                            isRedownload: false,
                                                            postbackURL: "TEST SERVE
    XCTFail("Failed to create postbacks.")
    return
}
```

After creating the SKAdTestPostback instances, the unit test calls the setPostbacks method on SKAdTestSession to add the newly created postbacks to the test session. Test postbacks added to the test session are available to the unit tests for further testing, including updating the conversion value or sending the postbacks. Test sessions can handle up to eight postbacks. Each call to the setPostbacks method overwrites the test postbacks in the test session.

```
try testSession.setPostbacks(testPostbacks)
```

Test updating the conversion value

A postback isn't ready to send to the server until it has a conversion value. The unit test updates the conversion value of the first test postback by calling the updatePostbackConversionValue method on SKAdNetwork. To confirm the conversion value after updating it, the unit test retrieves the postbacks using the postbacks property on SKAdTestSession and then checks the conversion value.

```
let fetchedPostbacks = testSession.postbacks
guard fetchedPostbacks.count == 3 else {
    XCTFail("Expecting 3 postbacks, received \(fetchedPostbacks.count).")
    return
}

// A test session's `postbacks` property maintains the order of the postbacks.
let firstPostback = fetchedPostbacks[0]
XCTAssertEqual(firstPostback.fineConversionValue, 42)
```

Send and receive the test postback

The unit test sends the postbacks with updated conversion values to the network server by calling the flushPostbacks method on SKAdTestSession The server URL is specified in the testAddingPostbacks method, where the test postbacks are created.

> **Important**
>
> Before attempting to send postbacks using the flushPostbacks method, check that the specified test server is running and accepting connections.

After sending the test postbacks, the unit test waits for a response from the server. The server responds with one SKAdTestPostbackResponse instance for each test postback. The

response contains information about the success or failure of the postback, the error details, and the HTTP response the server received, if any.

In the `testSendingPostback()` method, the unit test checks the response for the success or failure flag and the error object. The unit test passes only if it receives a success signal in combination with a `nil` error. Calling the `flushPostbacks` method removes the test postbacks from the test session.

```
testSession.flushPostbacks { responses, error in
    XCTAssertNil(error)
    guard let concreteResponses = responses else {
        XCTFail("No responses received.")
        return
    }
    for response in concreteResponses {
        let postbackResponse = response.value
        XCTAssertNil(postbackResponse.error)
        XCTAssertTrue(postbackResponse.didSucceed)
    }
}
```

See Also

Ad impression and postback testing

`class` SKAdTestSession

The class you use to test ad impressions and postbacks in Xcode.

`class` SKAdTestPostback

A test postback that contains ad conversion information in the testing environment.

`class` SKAdTestPostbackResponse

The status and error information for a postback that the system sends in the testing environment.

`struct` SKAdTestPostbackVersion

A constant that indicates the postback version.