Integrating the Jamf Certificate SDK into Your iOS App

Version 1.0.0

Contents

Introduction

Target Audience

This guide is designed for developers that are familiar with iOS app development.

What's in This Guide

This guide provides instructions for integrating the Jamf Certificate SDK with your apps and instructions for using the Jamf Certificate SDK Sample App for testing purposes.

In addition, this guide provides a library, code samples, and descriptions for you to develop your apps with the Jamf Certificate SDK and a reference for the required Managed App Configuration.

Important Concepts

Before using this guide, make sure you are familiar with Managed App Configuration. For more information, see the following websites:

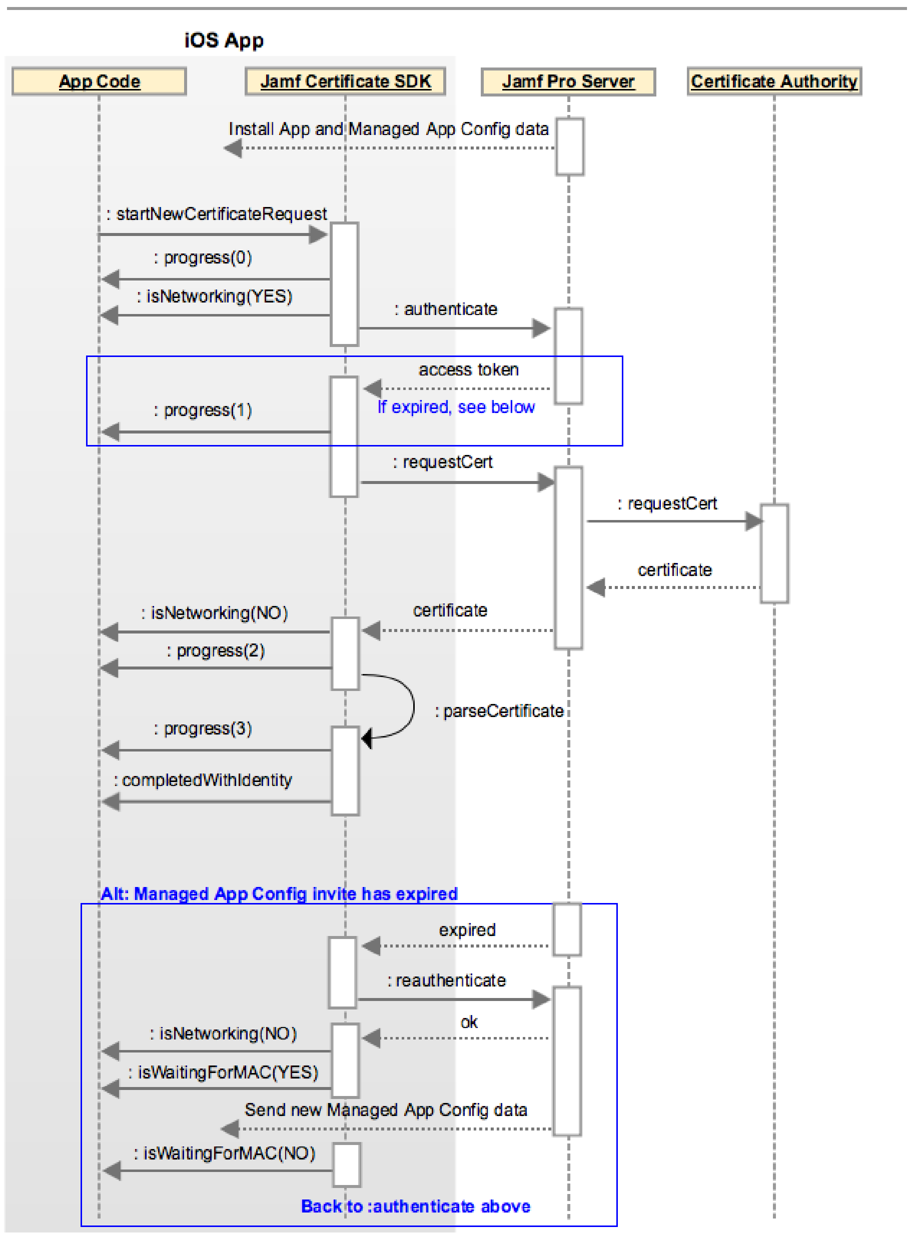
- https://developer.jamf.com/app-config
- https://www.appconfig.org/ios/

Overview

When the Jamf Certificate SDK is integrated with your iOS app, it provides a secure process that allows the app to request a certificate from a certificate authority (CA) via Jamf Pro. Certificates can be used to establish identities that support certificate-based authentication to perform Single Sign-On (SSO) or other actions specific to your environment.

To ensure successful communication between the app and the CA server, your app must be distributed using Jamf Pro. This allows the required Managed App Configuration to be applied to the app during distribution. The Managed App Configuration includes the required information needed by the SDK to authenticate with Jamf Pro. This enables the SDK to request the full certificate and private key as a .p12 bundle that Jamf Pro gets from the CA server.

The following diagram illustrates how a certificate is transferred from the CA to the app using Jamf Pro:



In addition, you can use the included Jamf Certificate SDK Sample App to test the certificate communication in your environment. The Sample App includes two types of tests: an Actual test and a Simulated test. The Actual test makes network calls to the CA server using Jamf Pro and Managed App Configuration. The Simulated test includes an embedded .p12 file for local testing so you can simulate certificate communication without a Jamf Pro instance or the required Managed App Configuration.

General Requirements

To develop your iOS apps with the Jamf Certificate SDK and distribute them to mobile devices, ensure you have the following:

	Requirements	Related Information
App Development	To develop your apps with the Jamf Certificate SDK, you need Xcode 9.0 or later with iOS SDK 10.0 or later	For more information about Xcode, see the following website: https://developer.apple.com/xcode/
App Distribution	Apps developed with the Jamf Certificate SDK can only be distributed to mobile devices with iOS 10 or later. **Note**: The version of iOS required for your app may vary. In addition, the following conditions are required to distribute apps developed with the Jamf Certificate SDK: ▪ The app must be distributed using Jamf Pro. ▪ The app must be managed with Jamf Pro and have a Managed App Configuration applied to it. ▪ The devices the app is distributed to must be enrolled with Jamf Pro.	For more information about app distribution using Jamf Pro, see App Distribution in the *Jamf Pro Administrator's Guide*. For more information about device management using Jamf Pro, see the following website: https://www.jamf.com/products/jamf-pro/device-management/ For more information about Managed App Configuration, see the following websites: ▪ https://developer.jamf.com/app-config ▪ https://www.appconfig.org/ios/
Network Communication	Standard HTTPS ports (typically 443) are used for communication between apps developed with the Jamf Certificate SDK and Jamf Pro. All other communication requirements between devices and Jamf Pro are standard with Jamf Pro.	For all other communication requirements between mobile devices and Jamf Pro, see the Network Ports Used by Jamf Pro Knowledge Base article.
Jamf Pro Environment	Active Directory Certificate Services (AD CS) must be added as a PKI Provider in Jamf Pro. This allows you to use AD CS as the certificate authority (CA).	For more information about adding a PKI Provider to Jamf Pro, see PKI Certificates in the *Jamf Pro Administrator's Guide*. For more information about adding AD CS as the PKI Provider in Jamf Pro, see the Integrating with Active Directory Certificate Services (AD CS) Using Jamf Pro technical paper for more information.

Integrate with the Jamf Certificate SDK

Before you can distribute your iOS apps developed with the Jamf Certificate SDK, you must integrate the Jamf Certificate SDK with your app code.

Requirements

Ensure the general requirements for app development are met. For more information, see General Requirements.

Integrating with the Jamf Certificate SDK

1. Follow the instructions in the following Apple documentation to embed the Jamf Certificate SDK in your app:
   Embedding Frameworks In An App
2. Implement the `CertificateRequestDelegate` protocol in your code.
   Note: The delegate messages are not called on the main thread. To update any user interface, you must dispatch back to the main thread.
3. Create an instance of a subclass of `CertificateRequestBase` passing it a reference to your `CertificateRequestDelegate` protocol, and keep a strong reference to the request object.
   The available subclasses are `CertificateRequestEmbeddedP12` (for testing) and `CertificateRequestWorkflow` (for production).
4. Call `-startNewCertificateRequest` on the object created in step 3.
   This call initiates the request asynchronously and returns immediately.
   Your delegate will be called with progress (if implemented by your delegate), when networking starts /stops, any errors, and a completion message.

After your app is developed with the SDK, the app can be distributed to a device using Jamf Pro. This allows the required Managed App Configuration to be applied to the app during distribution. For more information, see the "Distribute In-House Apps Developed with the Jamf Certificate SDK" section in the Integrating with Active Directory Certificate Services (AD CS) Using Jamf Pro technical paper.

Test Certificate Communication Using the Jamf Certificate SDK Sample App

You can test the certificate communication in your environment using the Jamf Certificate SDK Sample App that is included with the SDK. You can run the following types of tests depending on your environment:

- **Actual**–The Actual test type makes network calls to the Jamf Pro server to request a certificate from a certificate authority (CA). This test ensures that communication is established between the app and Jamf Pro, and that the Managed App Configuration is sending the required information needed by the app to the CA server.
  Note: This test requires access to a Jamf Pro instance.
- **Simulated**–The Simulated test type includes an embedded .p12 file for local testing so you can simulate certificate communication without a Jamf Pro instance or the required Managed App Configuration. You can use settings in the Sample App to simulate a slow network connection and an error state.
  Note: This test does not require access to a Jamf Pro instance.

Test progress, errors, and results are displayed for both types of tests in the Action Log of the Sample App interface.

In addition, you can schedule a local notification that prompts to renew the certificate.

Requirements

To test certificate communication using the Actual test type, you need access to a Jamf Pro instance so you can distribute the Sample App to a device using Jamf Pro. This allows you to apply the required Managed App Configuration to the app during distribution. For more information, see the "Distribute In-House Apps Developed with the Jamf Certificate SDK" section in the Integrating with Active Directory Certificate Services (AD CS) Using Jamf Pro technical paper.

To test certificate communication using the Simulated test type, the Sample App can be distributed to a device through any means that fit your environment.

In addition, ensure the general requirements for distributing apps are met. For more information, see General Requirements.

Testing Certificate Communication Using the Jamf Certificate SDK Sample App

1. Open the Sample App on the device it is installed on.
2. Do one of the following:
   - To use the actual test type, tap **Actual** and then tap **Run Test**.
     The progress of the actual test is displayed in the Action Log, along with any errors that may occur.

  - To use the simulated test type, tap **Simulated**, apply your desired settings, and then tap **Run Test**.
    The progress of the simulated test is displayed in the Action Log.
    If you selected the **Error simulation** setting, the errors are displayed in the Action Log. For a list of potential errors, see CertificateRequestErrorDomain Constants Reference.

3. (Optional) To schedule a local notification to prompt a user to renew the certificate, tap **Schedule Notification**.
   This notification is displayed after **Schedule Notification** is tapped.
   - If the app is running in the background or closed when the notification is displayed, the app must be reopened to start the certificate renewal process. Tapping the notification also reopens the app.
   - If the app is running in the foreground, the certificate automatically renews. The progress of the certificate renewal process is displayed in the Action Log.

   The certificate renews using the Simulated test type.
4. To reset a test, tap **Reset**.
   Note: The certificate used for the test remains in the keychain until **Reset** is tapped.
5. To rerun a test, tap **Rerun Test**.

Jamf CertificateRequest Reference

This section provides a library, code samples, and descriptions for you to develop your iOS apps with the Jamf Certificate SDK.

Class References

- CertificateRequestBase
- CertificateRequestEmbeddedP12
- CertificateRequestWorkflow

Protocol References

- CertificateRequestProtocol
- CertificateRequestDelegate

Constant References

- CertifcateRequestSDKVersionNumber
- CertificateRequestSDKVersionString
- JAMFSecurityErrorDomain
- CertificateRequestErrorDomain

CertificateRequestBase Class Reference

Inherits From	`NSObject`
Conforms To	`CertificateRequestProtocol`
Declared In	`CertificateRequestBase.h`

This class is the base class for all `CertificateRequestProtocol` objects. Subclasses will have a concrete implementation of requesting certificates.

Properties

delegate

The object that will be called back with progress, errors, and completion.

```
@property (weak, readonly) id<CertificateRequestDelegate> delegate
```

Declared In

`CertificateRequestBase.h`

maxNumberOfSteps

This is the maximum number of steps in the progress of a certificate request. May be approximate until the actual call of `-startNewCertificateRequest`:

```
@property (assign, readonly) NSUInteger maxNumberOfSteps
```

Declared In

`CertificateRequestBase.h`

Instance Methods

initWithDelegate:

Create an object.

```
- (nullable instancetype)initWithDelegate:
(id<CertificateRequestDelegate>)delegate
```

Parameters

`delegate`	Your `delegate` that will be called as the certificate request workflow proceeds.

Return Value

An initialized object; may be nil if memory is full

Declared In

`CertificateRequestBase.h`

Back to Jamf CertificateRequest Reference

CertificateRequestEmbeddedP12 Class Reference

Inherits From	`CertificateRequestBase : NSObject`
Conforms To	`CertificateRequestProtocol`
Declared In	`CertificateRequestEmbeddedP12.h`

This class reads the certificate from an embedded .p12 file, and can force errors for testing.

Properties

secondsBetweenSteps

Use this option to artificially delay the completion of each step. Each step up to `maxNumberOfSteps` will take this many seconds. This allows you to test things that are hard to test otherwise, such as progress. Defaults to zero (full speed).

```
@property (assign) NSUInteger secondsBetweenSteps
```

Declared In

`CertificateRequestEmbeddedP12.h`

Instance Methods

initWithDelegate:p12File:p12Password:

Create an object.

```
- (nullable instancetype)initWithDelegate:
(id<CertificateRequestDelegate>)delegate p12File:(NSURL)p12URL
p12Password:(NSString)pwd
```

Parameters

`delegate`	Your delegate that will be called as the certificate request workflow proceeds
`p12URL`	A URL to a file with a .p12 certificate that will be returned
`pwd`	The password for the .p12 file

Return Value

An initialized object; may be nil if memory is full

Declared In

`CertificateRequestEmbeddedP12.h`

Back to Jamf CertificateRequest Reference

CertificateRequestWorkflow Class Reference

Inherits From	`CertificateRequestBase : NSObject`
Conforms To	`CertificateRequestProtocol`
Declared In	`CertificateRequestWorkflow.h`

This class is the main class that should be instantiated to make a certificate request.

Properties

networkingTimeout

Use this option to specify a custom timeout in seconds for individual network requests. Defaults to 30 seconds.

```
@property (assign) NSUInteger networkingTimeout
```

Declared In

`CertificateRequestWorkflow.h`

pollingInterval

Use this option to specify a custom interval in seconds to poll for certificate creation. The first retrieval will happen after this interval, which means this also specifies the minimum amount of time required for the certificate request. Minimum value is 1 second. Defaults to 5 seconds.

```
@property (assign) NSUInteger pollingInterval
```

Declared In

`CertificateRequestWorkflow.h`

pollingTimeout

Use this option to specify an overall timeout in seconds to poll for certificate creation. After this much time, if the server has still not returned a certificate then the certificate request will fail with a timeout error. Defaults to 180 seconds.

```
@property (assign) NSUInteger pollingTimeout
```

Declared In

`CertificateRequestWorkflow.h`

Instance Methods

initWithDelegate:

Create an object.

```
- (nullable instancetype)initWithDelegate:
(id<CertificateRequestDelegate>)delegate
```

Parameters

`delegate`	Your delegate that will be called as the certificate request workflow proceeds.

Return Value

An initialized object; may be nil if memory is full

Declared In

`CertificateRequestWorkflow.h`

initWithDelegate:testSettings:

Create an object with given settings instead of relying on Managed App Config. This is a testing method.

```
- (nullable instancetype)initWithDelegate:
(id<CertificateRequestDelegate>)delegate testSettings:(NSDictionary)
testSettings
```

Parameters

`delegate`	Your delegate that will be called as the certificate request workflow proceeds.
`testSettings`	The settings that would otherwise be provided by Managed App Config.

Return Value

An initialized object; may be nil if memory is full

Declared In

`CertificateRequestWorkflow.h`

Back to Jamf CertificateRequest Reference

CertificateRequestProtocol Protocol Reference

Declared In	`CertificateRequestProtocol.h`

This is the protocol used for requesting and renewing certificates from Jamf Pro.

Properties

maxNumberOfSteps

This is the maximum number of steps in the progress of a certificate request. May be approximate until the actual call of `startNewCertificateRequest`:

```
@property (assign, readonly) NSUInteger maxNumberOfSteps
```

Declared In

`CertificateRequestProtocol.h`

Instance Methods

cancelRequest

Cancels the current request, if any. The completion handler WILL be called on the delegate.

```
- (void)cancelRequest
```

Declared In

`CertificateRequestProtocol.h`

startNewCertificateRequest

This will start a request to the Jamf Pro server for a new certificate. The delegate will be called with progress, errors, and the new identity.

Note: Only one such request should be in progress at a given time for a single `CertificateRequest` object.

```
- (void)startNewCertificateRequest
```

Declared In

`CertificateRequestProtocol.h`

Back to Jamf CertificateRequest Reference

CertificateRequestDelegate Protocol Reference

Conforms To	`NSObject`
Declared In	`CertificateRequestDelegate.h`

Third-party apps should have a class that implements these methods for notification on progress, errors, and completion.

Instance Methods

certificateRequest:completedWithIdentity:

When the request to Jamf Pro is completed, this method will be called. If there were errors, the identity will be nil.

```
- (void)certificateRequest:(id<CertificateRequestProtocol>)request
completedWithIdentity:(nullable SecIdentityRef)identity
```

Parameters

`request`	The object that started the request
`identity`	An identity that encapsulates the info from the server

Declared In

`CertificateRequestDelegate.h`

certificateRequest:errorOccurred:

When any kind of error occurs in the process for requesting certificates, this delegate method will be called.

```
- (void)certificateRequest:(id<CertificateRequestProtocol>)request
errorOccurred:(NSError)error
```

Parameters

`request`	The object that initiated the request
`error`	An error object

Declared In

`CertificateRequestDelegate.h`

certificateRequest:isUsingNetwork:

When the CertificateRequest SDK begins accessing the network and when it ends accessing the network, this method is called if implemented.

```
- (void)certificateRequest:(id<CertificateRequestProtocol>)request
isUsingNetwork:(BOOL)isUsingNetwork
```

Parameters

`request`	The object that started the request
`isUsingNetwork`	Whether or not the request is currently using the network

Declared In

`CertificateRequestDelegate.h`

certificateRequest:isWaitingForMAC:

When the CertificateRequest SDK detects no settings in the Managed App Config key, or if the invitation in the Managed App Config has expired this will be called to let you know that it is waiting for a new MAC to be delivered from the Jamf Pro server. Will be called again when the MAC comes in; the timing on this can vary greatly.

```
- (void)certificateRequest:(id<CertificateRequestProtocol>)request
isWaitingForMAC:(BOOL)waitingForMAC
```

Parameters

`request`	The object that started the request
`waitingForMAC`	Whether or not the request is currently waiting for settings in the Managed App Config

Declared In

`CertificateRequestDelegate.h`

certificateRequest:progress:

As the steps are completed during the request to Jamf Pro, this method can be called. The current progress can be checked against the request.maxNumberOfSteps to show progress percentage.

Note: Each step may take a different (unknown) amount of time as the actual request will be using network resources.

```
- (void)certificateRequest:(id<CertificateRequestProtocol>)request
progress:(NSUInteger)current
```

Parameters

`request`	The object that started the request
`current`	The current progress

Declared In

`CertificateRequestDelegate.h`

Back to Jamf CertificateRequest Reference

CertificateRequestSDKVersionNumber Constants Reference

The `CertificateRequestSDKVersionNumber` is a double and `CertificateRequestSDKVersionString` is the text string of the number with the current release of the Jamf Certificate SDK being the version number.

Back to Jamf CertificateRequest Reference

CertificateRequestSDKVersionString Constants Reference

The `CertificateRequestSDKVersionString` is the text string of the number for the `CertificateRequestSDKVersionNumber` with the current release of the Jamf Certificate SDK being the version number.

Back to Jamf CertificateRequest Reference

JAMFSecurityErrorDomain Constants Reference

`JAMFSecurityErrorDomain` is used for a set of error codes from the Jamf Certificate SDK that are interacting with Apple's Security framework. An error code has the following prefix: `errSec`

For more information, see the following website:
https://developer.apple.com/documentation/security/1542001-security_framework_result_codes

Back to Jamf CertificateRequest Reference

CertificateRequestErrorDomain Constants Reference

Declared In	`CertificateRequestErrorDomain.h`

CertificateRequestErrors

The error domain for the SDK code.

Definition

```
typedef NS_ERROR_ENUM(kCertificateRequestErrorDomain) {
kErrorBadInitializationParams = 1,kErrorCertificateMalformed = 2,
kErrorCertificateWrongAutomaticPassword = 3,
kErrorIncorrectManagedAppConfigData = 4,
kErrorInvalidResponseFromJamfProServer = 5};
```

Constants

`kErrorBadInitializationParams`

A `CertificateRequestEmbeddedP12` object was initialized with bad parameters.

Declared In `CertificateRequestErrorDomain.h.`

`kErrorCertificateMalformed`

The certificate coming from Jamf Pro has been malformed in transit.

Declared In `CertificateRequestErrorDomain.h.`

`kErrorCertificateWrongAutomaticPassword`

Occurs when the .p12 from the server was encrypted with a different password than the SDK has chosen.

Declared In `CertificateRequestErrorDomain.h.`

`kErrorIncorrectManagedAppConfigData`

The Managed App Configuration does not contain enough information to contact Jamf Pro.

Declared In `CertificateRequestErrorDomain.h.`

`kErrorInvalidResponseFromJamfProServer`

The Jamf Pro server responded with information that the SDK cannot process.

Declared In `CertificateRequestErrorDomain.h.`

Back to Jamf CertificateRequest Reference