MobileIron®

MobileIron® AppConnect for iOS App Wrapping Developers Guide

iOS AppConnect Wrapper Version 1.9

December 20, 2013

Contents

Chapter 1

Introducing AppConnect for iOS wrapped apps

MobileIron AppConnect secures and manages enterprise apps on mobile devices. These secure enterprise apps are called *AppConnect apps* or *secure apps.*

You can create an AppConnect app for iOS two ways:

- Wrapping the app

  The MobileIron AppConnect wrapping technology creates a secure app without any further app development.
- Using the AppConnect for iOS SDK (software development kit)

  An app developer uses the SDK to create a secure app, or turn an existing app into a secure app.

For information about choosing between wrapping and the SDK, see "Wrapping versus using the AppConnect for iOS SDK" on page 9.

Note: You **cannot** wrap an app if either of the following are true:

- You got the app (IPA file) from the Apple App Store.
- The app is a 64-bit app. For 64-bit apps, the Architectures build setting in Xcode is "Standard Architectures (including 64-bit").

AppConnect for iOS wrapped app overview

Wrapped app features

Secure enterprise apps that are created using the AppConnect wrapping technology can:

- Tunnel network connections to servers behind an enterprise's firewall.

  This capability means that device users do not have to separately set up VPN access on their devices to use the app.
- Enforce data loss prevention.

  The VSP administrator decides whether an app can copy to the iOS pasteboard, use the document interaction feature (Open In), or use print capabilities. AppConnect for iOS uses this information to limit the app's functionality to prevent data loss through these features.
- Receive app-specific configuration information from the VSP.

  This capability requires some additional app development. It means that device users do not have to manually enter configuration details that the app requires. By automating this process for the device users, each user has a better experience when installing and setting up apps. Also, the enterprise has fewer support calls.

Note: If an app uses iOS data protection APIs, and the device has a device passcode, then the app's data is encrypted.

MobileIron AppConnect components for wrapped apps

Wrapped AppConnect apps work with the following MobileIron components:

MobileIron component	Description
VSP	The MobileIron server which provides security and management for an enterprise's devices, and for the apps and data on those devices. An administrator configures the security and management features using a web portal.
Standalone Sentry	The MobileIron server which provides secure network traffic tunneling from your app to enterprise servers.
Mobile@Work for IOS	The MobileIron app that runs on an iOS device. It interacts with the VSP to get current security and management information for the device. It interacts with the AppConnect library to communicate necessary information to your app.
The AppConnect app wrapper	Provided by the AppConnect wrapping technology, the app wrapper provides AppConnect capabilities to your app. For example, it includes the AppConnect library for communicating with Mobile@Work.

Using a wrapped app

An iOS device user can use a wrapped AppConnect app only if:

- The device user has been authenticated through the MobileIron VSP.

  The user must use the Mobile@Work for iOS app to register the device with the MobileIron VSP. Registration authenticates the device user. Once registered, the device user can use a secured enterprise app.
- The VSP administrator has authorized the device user to use the app.
- The device user has entered a passcode for using secure enterprise apps.

  This passcode is the AppConnect passcode. However, when presented to device users, it is called the secure apps passcode. The VSP administrator configures whether the AppConnect passcode is enabled, and also configures rules about its complexity.

  Note: The AppConnect passcode is not the same as the passcode used to unlock the device.

Product versions required

To deploy a wrapped AppConnect app, you need the following products:

MobileIron VSP 5.5 or later

AppConnect apps require VSP 5.5 or later.

VSP 5.7 is required if the VSP administrator configures your app to authenticate to an enterprise server using Single Sign On using Kerberos Constrained Delegation.

Standalone Sentry 4.5 or later

AppConnect apps require Standalone Sentry 4.5 or later if the VSP administrator configures the app to tunnel network connections.

Standalone Sentry 4.7 is required if the VSP administrator configures your app to authenticate to an enterprise server using Single Sign On using Kerberos Constrained Delegation.

Mobile@Work version 5.5 or later

AppConnect apps require the device to be running Mobile@Work for iOS, version 5.5 or later. Mobile@Work is available on the Apple App Store.

Note: To run AppConnect apps on iOS 7 devices, Mobile@Work 5.7.4 is required.

iOS 5.0 or later

AppConnect apps are supported on devices running iOS 5.0 or later.

On iOS 7 devices, running AppConnect apps requires the following:

- Mobile@Work 5.7.4 for iOS is running on the device.
- The app is wrapped with AppConnect for iOS Wrapper version 1.8 or later.

Note:

- Apps wrapped with AppConnect for iOS Wrapper version 1.8 or later can also run on iOS 5 or iOS 6 devices, running Mobile@Work 5.5 or later.
- Mobile@Work 5.7.4 on an iOS 7 device displays an error message when an app wrapped with AppConnect for iOS Wrapper versions prior to 1.8 attempts to check in. The error message indicates that the app cannot be secured and cannot access secure data until a new version is installed.

- The following table summarizes which AppConnect for iOS Wrapper versions and Mobile@Work versions are compatible with each iOS version:

	Mobile@Work 5.5	Mobile@Work 5.6.x	Mobile@Work 5.7, 5.7.1, 5.7.3	Mobile@Work 5.7.4
Wrapper versions prior to 1.8	iOS 5 and 6	iOS 5 and 6	iOS 5 and 6	iOS 5 and 6
Wrapper version 1.8 and later	iOS 5 and 6	iOS 5 and 6	iOS 5 and 6	iOS 5, 6, and 7

Chapter 2

Wrapping versus using the AppConnect for iOS SDK

Whether to use the AppConnect wrapping technology or the AppConnect for iOS SDK to create an AppConnect app depends on your situation.

Wrap the app if:

- You do not have the resources to modify an app using the SDK.

  For example, if you have an in-house app, but no developer to modify it, wrap the app. Similarly, if you no longer have the source code for the app, wrap the app.
- You do not have a need for the control that the SDK provides.

  Using the SDK, you have finer control of how the app handles AppConnect status changes. For example, if the user becomes unauthorized to use the app, you have more control of how to handle application logic, secure data removal, and user notification.
- Your app is an in-house app, distributed using the VSP's enterprise app storefront.

  Wrapped apps are not compliant with Apple's terms and conditions, and cannot be distributed using the Apple App Store. Furthermore, you cannot wrap an app (IPA file) that you get from the Apple App Store.

Use the SDK if:

- You want to distribute the app through the Apple App Store.

  However, you can also distribute apps created with the SDK as in-house apps, using the VSP's enterprise app storefront.
- You have the resources to modify an app using the SDK.

  If you have development resources and the app's source code, using the SDK is possible.
- You want the app to have finer control of policy changes than app wrapping provides.

  If the user becomes unauthorized to use the app, you have more control of how to handle application logic, secure data removal, and user notification.

  For example, if your app handles both secure and unsecured data, the SDK allows you to choose what app data to remove when the VSP retires the app. Retiring an app means that the user is unauthorized to use the app and that all the app's secure data should be deleted.

Chapter 3

Securing and managing a wrapped iOS AppConnect app

A VSP administrator configures how mobile device users can use secure enterprise applications. The administrator sets the following app-related settings that impact your wrapped app's behavior:

- "Authorization" on page 11
- "AppConnect passcode policy" on page 12
- "Tunneling" on page 12
- "Data loss prevention policies" on page 13
- "Handling app-specific configuration from the VSP" on page 13
- "Log messages based on log levels" on page 14

The following steps show the flow of information from the VSP to a wrapped app:

1. The VSP administrator decides which app-related settings to apply to a device or set of devices.
2. The VSP sends the information to the Mobile@Work app.
3. Mobile@Work passes the information to the wrapped AppConnect app. Mobile@Work and the AppConnect app wrapper enforce the app-related settings.

Authorization

The VSP administrator determines:

- whether or not each device user is authorized to use each secure enterprise app.

  When an unauthorized user launches the app, Mobile@Work displays a message to the user, and the app exits.
- the situations that cause an authorized device user to become unauthorized.

  These situations include, for example, when the device OS is compromised. Mobile@Work reports device information to the VSP. The VSP then determines whether to change the user to unauthorized based on security policies on the VSP.

  When a user becomes unauthorized, Mobile@Work displays a message to the user, and the app exits.
- the situations that retire the app.

  Retiring an app means that the user is not authorized to use it and the app's data is deleted. Mobile@Work displays a message to the user, and the app exits. Furthermore, the AppConnect app wrapper removes data associated with the app. Specifically, the app wrapper removes all data in the application's sandbox and in the application's keychain. It also resets the application's default settings.

Note: When an app is retired, the app wrapper removes the app’s data. When a user is unauthorized but the app is not retired, the app cannot run, so the user cannot access the data. However, the app wrapper does not remove the data. The reason is that an unauthorized user can become authorized again, and therefore the data should become available again.

AppConnect passcode policy

The VSP administrator determines:

- whether the AppConnect passcode is enabled, which requires the device user to enter a passcode to access any secure enterprise apps. One AppConnect passcode controls all secure enterprise apps on the device.
- the complexity of the AppConnect passcode.
- the inactivity timeout for the AppConnect passcode.

The AppConnect app wrapper and Mobile@Work enforce an AppConnect passcode as follows:

- The VSP notifies Mobile@Work when the VSP administrator has enabled an AppConnect passcode. Mobile@Work prompts the user to set the passcode the next time that the device user launches or switches to a secure enterprise app.
- Mobile@Work prompts the user to enter the passcode when the user subsequently launches or switches to a secure enterprise app but the inactivity timeout has expired.
- Mobile@Work prompts the user to enter the passcode when the inactivity timeout expires *while* the user is running a secure enterprise app.
- Mobile@Work prompts the user to set the passcode the next time the device user launches or switches to a secure enterprise app after the VSP has notified Mobile@Work that the passcode’s complexity rules have changed.

Tunneling

A secure enterprise app can securely tunnel HTTP and HTTPS network connections from the app to servers behind a company’s firewall. A Standalone Sentry is necessary to support tunneling. The VSP administrator handles all tunneling configuration on the VSP. Once the administrator has configured tunneling for the app on the VSP, the AppConnect app wrapper, Mobile@Work, and a Standalone Sentry handle tunneling for the app.

An app accesses its enterprise servers as it normally would using URL requests, using the iOS API NSURLRequest. Apps can also use networking libraries that use NSURLRequest, such as AFNetworking.

Note: An app that accesses sockets directly cannot use the tunneling feature. For example, Apple’s reachability APIs that detect network and host connectivity use sockets directly. Therefore, these APIs can not access a host behind the enterprise’s firewall using the tunneling feature.

Data loss prevention policies

An app can leak data if it uses iOS features such as copying to the iOS pasteboard, document interaction (Open In), and print capabilities. A VSP administrator specifies on the VSP whether each app is allowed to use each of these features. For the Open In policy, the VSP administrator also specifies which apps are in your app’s whitelist. The whitelist is the set of apps that can open documents from your app.

The administrator applies the appropriate policies to a set of devices. Sometimes more than one set of policies exists on the VSP for an app if different users require different policies.

The AppConnect app wrapper enforces the policies in the app. That is, depending on the VSP configuration, the app wrapper:

- disables the app’s printing.
- prohibits the app from copying content to the iOS pasteboard.
- Limits the Open In feature to the apps in your app’s whitelist.

  Note: The wrapper disables the Email action in your app when the VSP configuration specifies an Open In whitelist, unless the iOS email client is included in the whitelist.

Handling app-specific configuration from the VSP

Handling app-specific configuration from the VSP requires some application development before wrapping the app. If you do not use this feature, the app continues to set up its configuration as it always has.

Typically, wrapped apps do not use this feature. However, if you have application developer resources, you can take advantage of this feature.

You determine the app-specific configuration that your app requires from the VSP. Examples are:

- the address of a server that the app interacts with
- whether particular features of the app are enabled for the user
- user-related information from LDAP, such as the user’s ID and password
- certificates for authenticating the user to the server that the app interacts with

Each configurable item is a key-value pair. Each key and value is a string. A VSP administrator specifies the key-value pairs for each app on the VSP. The administrator applies the appropriate set of key-value pairs to a set of devices. Sometimes more than one set of key-value pairs exists on the VSP for an app if different users require different configurations. For example, the administrator can assign a different server address to users in Europe than to users in the United States.

Note: When the value is a certificate, the value contains the base64-encoded contents of the certificate, which is a SCEP or PKCS-12 certificate. If the certificate is password encoded, starting with VSP 5.6, the VSP automatically sends another key-value pair. The key’s name is the string *<name of key for certificate>*_MI_CERT_PW. The value is the certificate’s password.

To use app-specific configuration from the VSP in an app, a developer implements the following method in the class that implements the UIApplicationDelegate protocol:

```
-(NSString *)appConnectConfigChangedTo:(NSDictionary *)config
```

When the app first launches, or when a change has occurred to the app-specific configuration on the VSP, the AppConnect app wrapper calls this method.

The `config` parameter is an `NSDictionary` object which contains the current key-value pairs for the app-specific configuration.

The developer implements the `-appConnectConfigChangedTo:` method as follows:

1. Apply the configuration in the `config` parameter according to the application's requirements and logic.
2. Return the value `nil` if the configuration was successfully applied. Otherwise, return a string that describes the error that occurred in applying the configuration.

Log messages based on log levels

In a wrapped app, the AppConnect app wrapper supports logging messages according to the log level that the VSP administrator specifies for the app. The wrapper logs these messages to the device's console. The log data provides information to help troubleshoot issues with the apps.

Chapter 4

How to wrap an iOS app

The high-level steps for wrapping an iOS app into an AppConnect app are:

1. You submit the app to the MobileIron web portal.
2. MobileIron wraps the app and returns it to you.
3. You re-sign the returned app using a script that MobileIron provides to you.

Important: Do not submit an app for wrapping if either of the following are true:

- You got the app (IPA file) from the Apple App Store.
- The app is a 64-bit app. For 64-bit apps, the Architectures build setting in Xcode is "Standard Architectures (including 64-bit").

Submitting an app to the MobileIron web portal

Important: Before you submit an app for wrapping, sign the app according to Apple's requirements.

To submit a signed app for wrapping:

1. Login to the MobileIron support portal at http://support.mobileiron.com/appconnect.
2. Select Submit Your App Here.
3. Enter the requested information, including your email address, into the submission form.
4. Upload the signed app's IPA file.

   Upload only one IPA file with each submission.

   Note: If the IPA file is greater than 20 megabytes, you cannot upload it using the submission form. Instead, upload it to a secure file-sharing site. Note the URL in the submission form in the Brief Description field.

MobileIron wraps the app, and emails a URL to the email address you specified. The URL is the address from where you download:

- a new IPA file that is the wrapped app

  The name of the file is *<original app name>*-wrapped.ipa.
- a script called sign_wrapped_app.sh for re-signing the wrapped app

Re-signing the wrapped app

When you receive your wrapped app from MobileIron, re-sign the app using the script that MobileIron returns with the app. The script is called sign_wrapped_app.sh.

To run the script, use a Mac computer running OS X.

Before you run the script:

- Make sure the Xcode command-line tools are installed on the Mac. See developer.apple.com/xcode.
- Make sure the signing certificate that you created for the app is in the Mac's login keychain.
- Put the IPA file of the wrapped app and the sign_wrapped_app.sh script in the same directory for convenient access.

Running the sign_wrapped_app.sh script:

1. Open the Terminal application on the Mac.
2. Change to the directory containing the IPA file of the wrapped app and the sign_wrapped_app.sh script. For example:
   ```
   $cd ~/wrapping
   ```
3. Make sure that the sign_wrapped_app.sh script is executable. For example:
   ```
   $chmod 755 sign_wrapped_app.sh
   ```
4. Run the script, specifying two parameters: the original app's signing certificate and the IPA file of the wrapped app. For example:
   ```
   $./sign_wrapped_app.sh -i "iPhone Distribution: myCompanyName" myApp-wrapped.ipa
   ```
   Note: Specify the name of the signing certificate of the original, unwrapped app in double quotes. This name is also the original app's signing identity. The name has the format `"iPhone Distribution: <certificate name>"` where `<certificate name>` is typically the name of your company.
5. When prompted, enter the password to unlock your keychain.

   The script continues to run, displaying the following output when sucessful:
   ```
   /var/folders/6g/z1_193_x0lj6jkzmysxl5wz80000gq/T//resign-QJ4wZrPR/Payload/
   myApp.app/MISandbox.framework/Versions/A: replacing existing signature
   /var/folders/6g/z1_193_x0lj6jkzmysxl5wz80000gq/T//resign-QJ4wZrPR/Payload/
   myApp.app: replacing invalid existing signature
   $
   ```
   The script replaces the IPA file with a signed IPA file. The signed IPA file is the file you distribute to device users.

Optionally, you can specify a different output file for the signed IPA file. Use the -o option as follows:

```
$./sign_wrapped_app.sh -i "iPhone Distribution: myCompanyName"
                -o mySignedWrappedApp.ipa myApp-wrapped.ipa
```

Specifying custom entitlements:

By default, the sign_wrapped_app.sh script takes the Apple entitlements (for example, enabling iCloud, push notifications, and App Sandbox) from the app's embedded.mobileprovision provisioning profile. You can override this behavior by specifying an optional parameter when running the sign_wrapped_app.sh script. The parameter names an entitlements plist file.

```
-e <entitlements plist file name>
```

For example:

```
$./sign_wrapped_app.sh -i "iPhone Distribution: myCompanyName" -e entitle-
ments.plist myApp-wrapped.ipa
```

Troubleshooting the signed wrapped app

If your signed wrapped app exits unexpectedly when you launch it, the issue sometimes involves the original app's bundle ID. In some cases, the sign_wrapped_app.sh script cannot infer the original app's bundle ID, which results in problems running the app.

To correct this issue, run the script using the -b option to specify the original app's bundle ID. For example:

```
$./sign_wrapped_app.sh -i "iPhone Distribution: myCompanyName"
                -b com.myCompanyName.myApp myApp-wrapped.ipa
```

Chapter 5

iOS AppConnect Wrapper revision history

iOS AppConnect Wrapper version 1.9

New features

Log messages based on log levels

The AppConnect app wrapper in a wrapped app now supports logging messages according to the log level that the VSP administrator specifies for the app. The wrapper logs these messages to the device's console. The log data provides information to help troubleshoot issues with the apps. For more information on setting the log level, see the *VSP Administration Guide* or the *Connected Cloud Administration Guide.*

Signing script parameter for customizing entitlements

When re-signing a wrapped app, you can now specify an entitlements plist to apply a customized set of entitlements to the app. For more information, see "Re-signing the wrapped app" on page 15.

Resolved issues

The following issues were resolved for AppConnect for iOS Wrapper version 1.9:

- AP-1557: When using AppTunnel, if a tunneled URL was redirected to the same URL, the resulting connection was not tunneled. Now the resulting connection is tunneled.
- AP-1176: The VSP Admin Portal page Apps > App Tunnels now correctly displays the tunnel information when you have retired a device and when the device then re-registers with a different user.
- AP-1175: On some iOS 5.x devices, AppConnect apps that did not support fast app switching experienced an issue with the AppConnect library. The library causes the app to exit unexpectedly the second time the library attempted to check in with Mobile@Work. The issue has been fixed.

Known issues

The following are known issues in AppConnect for iOS SDK version 1.7:

- AP-1484: The auto-lock time (formerly called inactivity timeout) in the AppConnect global policy specifies the period of inactivity in AppConnect apps after which the device user must reenter the AppConnect passcode. In some cases, the user must reenter the passcode before a continuous inactivity period of the specified length has been reached.

iOS AppConnect Wrapper version 1.8.2

Resolved issues

The following issues were resolved for AppConnect for iOS Wrapper version 1.8.2:

- AP-1500: The AppConnect for iOS wrapper did not correctly handle updates to the app’s AppConnect policies and configuration in some cases. These cases have been fixed. The wrapper has also fixed some cases in which it did not correctly report the updated status of the app’s AppConnect policies and configuration to Mobile@Work.
- ISB-694: On iOS 7 devices, if a wrapped app is retired and then reinstalled, it now correctly initializes its AppConnect states.
- AP-1609: AppConnect apps wrapped with iOS AppConnect Wrapper 1.8.1 crashed on devices running Mobile@Work 5.6.2 or earlier. The issue has been fixed.

iOS AppConnect Wrapper version 1.8.1

Resolved issues

The following issues were resolved for AppConnect for iOS Wrapper version 1.8.1:

- AP-1487: An app that uses key-value coding with a UIApplicationDelegate class object now works in all cases.
- AP-1486: AppTunnel now supports synchronous AJAX calls made from JavaScript code in WebKit-based browsers.

iOS AppConnect Wrapper version 1.8

New features

iOS 7 support

AppConnect for iOS Wrapper version 1.8 adds support for iOS 7 devices. Running wrapped AppConnect apps on iOS 7 devices requires the following:

- The app is wrapped with AppConnect for iOS Wrapper version 1.8.
- Mobile@Work 5.7.4 for iOS is running on the device.

Apps wrapped with AppConnect for iOS Wrapper version 1.8 can also run on iOS 5 or iOS 6 devices, running Mobile@Work 5.5 or later.

For more information, see "iOS 5.0 or later" on page 6.

Re-signing procedure

The procedure for re-signing a wrapped app has changed. The procedure has been reverted to use the sign_wrapped_app.sh script.

For more information, see "Re-signing the wrapped app" on page 15.

Known issues

The following are known issues for AppConnect for iOS Wrapper version 1.8:

- AP-856: An AppConnect for iOS app does not work properly if the app's bundle ID is greater than 46 characters. This limitation applies to apps built with the SDK as well as wrapped apps.

iOS AppConnect Wrapper version 1.7

New features

This version of the iOS AppConnect Wrapper adds these features:

- support for devices running iOS 6.0
- including the Wrapper version number of an app in the device log and the VSP's mifs.log

Resolved issues

This version of the iOS AppConnect Wrapper resolves a number of stability issues.