MobileIron AppConnect Guide for MobileIron Cloud

October 28, 2020

For complete product documentation see:
MobileIron Cloud Product Documentation Home Page

Contents

AppConnect Overview

AppConnect is a MobileIron feature that containerizes apps to protect data on iOS and Android devices. Each AppConnect-enabled app becomes a secure container whose data is encrypted, protected from unauthorized access, and removable. Because each user has multiple business apps, each app container is also connected to other secure app containers. This connection allows the AppConnect-enabled apps to share data, like documents. Polices and configurations set up in a MobileIron unified endpoint management (UEM) platform are used to manage AppConnect-enabled apps.

The MobileIron UEM are MobileIron Cloud and MobileIron Core.

While AppConnect protects data on a device -- data-at-rest, another MobileIron feature, AppTunnel, protects data as it moves between a device and enterprise data sources -- data-in-motion. MobileIron AppTunnel is a MobileIron feature that provides secure tunneling and access control to enterprise data sources. App-by-app session security protects the connection between each app container and the corporate network. AppTunnel is particularly useful when an organization does not want to open up VPN access to all apps on the device. This feature requires a Standalone Sentry configured to support app tunneling.

Related topics

- What are AppConnect-enabled apps?
- AppTunnel overview
- The AppConnect passcode
- AppConnect apps and authentication to enterprise app servers
- App-specific configuration for AppConnect apps
- AppConnect for Android overview
- AppConnect for iOS overview

What are AppConnect-enabled apps?

AppConnect-enabled apps, also known as AppConnect apps, are apps that have been containerized using one of the following methods:

- wrapping (iOS and Android)
- AppConnect SDK (iOS)
- AppConnect Cordova Plugin (iOS)

You configure and distribute AppConnect apps to devices on the MobileIron unified endpoint management (UEM) platform. The MobileIron UEM are MobileIron Cloud and MobileIron Core. From the device user perspective, AppConnect apps are called secure apps. Secure apps can share data only with other secure apps. Unsecured apps cannot access the data.

Related topics
AppConnect Overview

AppConnect apps from MobileIron

MobileIron provides a number of AppConnect apps, including Email+, Web@Work, and Docs@Work.

Third-party and in-house AppConnect apps

Your organization and third-party providers can create secure apps by either:

- wrapping the apps (Android and iOS)
- developing iOS apps by using the AppConnect for iOS SDK or AppConnect for iOS Cordova Plugin

NOTE: You cannot wrap an app that you get from Google Play or the Apple App Store.

See the following for details about how to wrap or develop an AppConnect app :

- AppConnect for Android Product Documentation Home Page
  - *MobileIron AppConnect for Android App Developers Guide*
- AppConnect for iOS Product Documentation Hope Page
  - *MobileIron AppConnect for iOS App Wrapping Developers Guide*
  - *MobileIron AppConnect for iOS SDK App Developers Guide*
  - *MobileIron AppConnect for iOS Cordova Plugin Developers Guide*

AppTunnel overview

MobileIron AppTunnel provides per-app secure tunneling and access control to protect app data as it moves between the device and corporate backend resources. You configure MobileIron Cloud and Standalone Sentry to support AppTunnel for an app. AppTunnel provides:

- HTTP/S tunneling
- TCP tunneling (also known as Advanced AppTunnel)

HTTP/S tunneling

AppTunnel can tunnel HTTP/S traffic between an iOS or Android AppConnect app and a corporate backend resource. The apps must use specific APIs to make their HTTP/S connections. Contact the app vendor or developer to find out if the app works with AppTunnel for HTTP/S tunneling.

TCP tunneling (also known as Advanced AppTunnel)

AppTunnel can tunnel TCP traffic between an AppConnect app and a corporate backend resource. A TCP tunnel supports HTTP/S connections and TCP connections. TCP tunneling for AppConnect apps is set up differently for iOS and Android:

- Android AppConnect apps

  AppConnect apps for Android that are wrapped with the Generation 2 wrapper support TCP tunneling using AppTunnel.
- iOS AppConnect apps

  AppConnect apps for iOS support TCP tunneling using the MobileIron Tunnel app. Therefore, to use TCP tunneling for iOS AppConnect apps, in addition to Standalone Sentry, also deploy and install MobileIron Tunnel on iOS devices and apply the Tunnel VPN profile to the iOS AppConnect app.

NOTE: AppTunnel does not support UDP tunneling. Therefore, if an AppConnect app requires UDP, such as for streaming video, it cannot use AppTunnel to tunnel its data.

Related topics

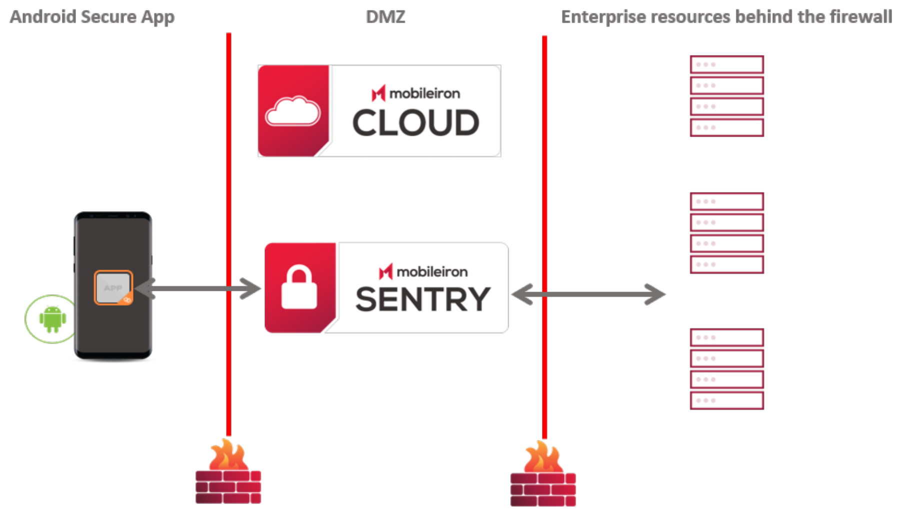
- AppTunnel with TCP tunneling support for Android AppConnect apps
- See "AppTunnel with Standalone Sentry" in the *MobileIron Sentry Guide for MobileIron Cloud* on the MobileIron Sentry Product Documentation Home Page.
- For information about deploying MobileIron Tunnel for iOS, see *MobileIron Tunnel for iOS Guide* on the MobileIron Tunnel for iOS Product Documentation Home Page.

AppTunnel with TCP tunneling support for Android AppConnect apps

AppTunnel can tunnel HTTP/S requests from an AppConnect app to an enterprise server that is behind the enterprise firewall. AppTunnel with HTTP/S tunneling is supported with wrapped Java apps that use a specific set of Java HTTP/S APIs. If a wrapped Java app uses APIs outside of this set, or uses TCP for its network connections, it can use AppTunnel with TCP tunneling to secure data-in-motion to enterprise servers. AppTunnel with TCP tunneling therefore expands the set of AppConnect apps that can tunnel data to an enterprise server.

When an AppConnect app uses AppTunnel with TCP tunneling, the traffic between the device and the Standalone Sentry is secured using an Secure Sockets Layer (SSL) session, as shown in the following diagram:

FIGURE 1. APPTUNNEL WITH TCP TUNNELING FOR ANDROID DEVICES



Types of apps that can use AppTunnel with TCP tunneling

The following types of apps can use AppTunnel with TCP tunneling:

- Hybrid web apps, including PhoneGap apps.
  Hybrid web apps use Android WebView and WebKit technologies to access and display web content. WebView does not use one of the Java HTTP/S APIs that Android AppConnect wrapping supports with AppTunnel with HTTP/S tunneling. Therefore, AppTunnel with TCP tunneling is required.
- Java apps
  Java apps that use APIs outside of the set of Java HTTP/S APIs that AppTunnel with HTTP/S tunneling supports can tunnel the data using AppTunnel with TCP tunneling.
- Java apps which use C or C++ code to access an enterprise server
  C or C++ code does not use the set of Java HTTP/S APIs that AppTunnel with HTTP/S tunneling supports. These apps can tunnel the data using AppTunnel with TCP tunneling.
- React Native apps
- Xamarin apps that use APIs outside the set of APIs that AppTunnel with HTTP/S tunneling supports.

Note The Following:

- AppTunnel does not support UDP tunneling. For example, apps that require UDP for streaming video are not supported.
- AppTunnel with TCP tunneling does not support Kerberos authentication to the enterprise server. It supports only pass through authentication. With pass through authentication, the Standalone Sentry passes the authentication credentials, such as the user ID and password (basic authentication) or NTLM, to the enterprise server.
  Therefore, apps that must use AppTunnel with TCP tunneling, such as hybrid apps, cannot use Kerberos authentication to the enterprise server. However, these apps can use "Certificate authentication using AppConnect with TCP tunneling for Android AppConnect apps.

When to use AppTunnel with HTTP/S tunneling versus TCP tunneling

The following table shows whether to use AppTunnel with HTTP/S tunneling or AppTunnel with TCP tunneling for an Android AppConnect app. It also shows which generation of the wrapper to use.

TABLE 1. APPTUNNEL SUPPORT FOR HTTP/S VERSUS TCP TUNNELING ON ANDROID APPCONNECT APPS

	AppTunnel with HTTP/S tunneling	**AppTunnel with TCP tunneling**
Java code using supported HTTP/S APIs [*]	Supported with Generation 1 or 2 wrapper	Supported Requires Generation 2 wrapper
Java code using unsupported HTTP/S APIs [*]	Not supported	Supported Requires Generation 2 wrapper
C or C++ code	Not supported	Supported Requires Generation 2 wrapper
Hybrid web app, including Phonegap apps	Not supported	Supported Requires Generation 2 wrapper
Xamarin apps	Supported with Generation 1 or 2 wrapper if using supported HTTP/S APIs	Supported Requires Generation 2 wrapper
React Native	Not supported	Supported Requires Generation 2 wrapper

[*] The supported HTTP/S Java APIs are listed in the *MobileIron AppConnect for Android App Developers Guide*.

Contact the application vendor or developer to find out whether to configure AppTunnel with HTTP/S tunneling or AppTunnel with TCP tunneling.

The AppConnect passcode

You can require an AppConnect passcode, also known as the secure apps passcode. With a single login using the AppConnect passcode, the device user can access all the secure apps. You configure the rules for the AppConnect passcode on MobileIron Cloud. The AppConnect passcode is not the same as the passcode used to unlock the device.

For the highest possible security when using AppConnect, MobileIron recommends that devices use both of the following:

- a device passcode
- an AppConnect passcode

In some environments, however, using both passcodes is not feasible due to usability and other requirements. For these reasons, you have the option to not require an AppConnect passcode. Also, you can allow Touch ID or Face ID (iOS) or fingerprint (Android) instead of an AppConnect passcode for accessing AppConnect apps for a simpler user experience.

When the AppConnect passcode is not required, users enter only a device passcode, if one is required, to unlock the device. Users are not encumbered with entering a second authentication to access secure apps. Only *access* to the secure apps changes. The apps are AppConnect-enabled, therefore, secured with AppConnect features such as data loss prevention policies. Also, the secure apps' data is still protected with encryption. However, no AppConnect passcode means that data encryption does not use the AppConnect passcode in creating the encryption key.

Your organization's security requirements determine whether accessing secure apps without an AppConnect passcode is an acceptable trade-off for an improved user experience.

Related topics

- Data encryption for secure apps for Android
- Data encryption for secure apps for iOS
- Touch ID or Face ID for accessing secure apps.
- Security versus convenience of passcode and Touch ID/Face ID options
- Fingerprint login for AppConnect apps for Android

AppConnect apps and authentication to enterprise app servers

You can set up AppConnect apps to provide device users a seamless authentication experience to your enterprise applications. In such a setup, users do not have to enter any credentials when accessing enterprise applications from an AppConnect app from a device managed by MobileIron. When users launch an AppConnect app, MobileIron Go on the managed device authenticates the user. After the user is authenticated the user can access the enterprise application without having to enter any credentials.

The following methods are available to support this capability:

- Authentication using Kerberos Constrained Delegation
- Certificate authentication for Android AppConnect apps
- Certificate authentication for iOS AppConnect apps
- Authentication through MobileIron Access

Authentication using Kerberos Constrained Delegation

You can use Kerberos Constrained Delegation (KCD) for authenticating a user to an enterprise server.

To use this feature, the app must do the following:

- Use the AppTunnel feature, configured for authenticating the user to the enterprise server using Kerberos Constrained Delegation (KCD).
- Interact with an enterprise server that supports authentication using KCD.

NOTE: AppConnect-enabled ActiveSync email apps such as , Email+ for Android, and Email+ for iOS do not use AppTunnel. You configure the Standalone Sentry for authenticating the user to the ActiveSync server using KCD.

All AppConnect apps can use this feature, including:

- Android third-party AppConnect apps
- iOS third-party AppConnect apps built with the AppConnect for iOS SDK or the AppConnect for iOS Cordova Plugin
- Web@Work
- Docs@Work

NOTE: MobileIron does not support KCD with CIFS-based content servers.

Certificate authentication for Android AppConnect apps

An Android AppConnect app can send a certificate to identify and authenticate the app user to an enterprise server when using AppTunnel with TCP tunneling.

Related topics

Certificate authentication using AppConnect with TCP tunneling for Android AppConnect apps.

Certificate authentication for iOS AppConnect apps

An iOS AppConnect app can send a certificate to identify and authenticate the app user to an enterprise service.

Related topics

Certificate authentication from AppConnect apps to enterprise services.

Authentication through MobileIron Access

For an AppConnect app in a MobileIron Access deployment, if an enterprise cloud service is set up in Access,

- Authentication to the cloud service goes through Access.
- If AppTunnel rules are configured in the AppConnect app configuration, data traffic goes through AppTunnel, however authentication traffic goes through Tunnel to Access.
- In addition, with zero sign-on, device users can get passwordless access to cloud services on their managed devices.

Related topics

- For information about MobileIron Access and how to set up Access, see the *MobileIron Access Guide* on the MobileIron Access Product Documentation Home Page.

App-specific configuration for AppConnect apps

On MobileIron Cloud, you can configure settings that are specific to an AppConnect app. Because MobileIron Cloud provides these settings to the app, device users do not have to manually enter configuration details that an AppConnect app requires. By automating the configuration for the device users, each user has a better experience when installing and setting up apps. Also, the enterprise has fewer support calls, and the app is secured from misuse due to misconfiguration. This feature is also useful for apps which do not want to allow the device users to provide certain configuration settings for security reasons.

Each AppConnect-enabled app's product documentation should specify the necessary configuration for the app.

Compliance actions

Compliance actions determine what happens if a device does not comply with policy requirements.

When MobileIron Go detects that the device is non-compliant it takes action depending on the compliance action set in the policy.

Some compliance actions impact AppConnect apps as follows:

- Immediately block access to the web sites configured to use the AppTunnel feature.
- Unauthorize AppConnect apps.
- Delete (wipe) the secure data of AppConnect apps.

Related topics

- Situations that wipe Android AppConnect app data
- Situations that wipe AppConnect for iOS app data
- For information about policies and their associated compliance action, see "Policies" in the *MobileIron Cloud Administrator Guide*.

AppConnect for Android overview

MobileIron supports AppConnect for Android by wrapping Android apps. The following sections provide an overview.

Wrapping modes

Two modes of wrapping are available:

- Generation 2
- Generation 1

Generation 2 wrapping is the default mode, and is required for a number of Android features. Generation 1 wrapping should only be used for features not supported by Generation 2. For information about the features supported by Generation 2 and Generation 1 wrapping modes, see "Wrapping support of commonly used app capabilities" in the *MobileIron AppConnect for Android App Developers Guide* available on the MobileIron AppConnect for Android Product Documentation Home Page.

NOTE: AppConnect apps are supported only in multiple-app kiosk mode. They are not supported in single-app kiosk mode. For Kiosk mode information, see the *MobileIron Cloud Administrator Guide*on the MobileIron Cloud Product Documentation Home Page.

The MobileIron client app, the Secure Apps Manager, and the AppConnect wrapper

Two MobileIron apps work together on the Android device to support AppConnect. Together, they provide the security and management of all the AppConnect apps.

These MobileIron apps are:

- MobileIron Go
- Secure Apps Manager

Each AppConnect app is wrapped with the AppConnect wrapper, which enforces security along with the MobileIron client app and the Secure Apps Manager. On the device, the AppConnect apps are called *secure apps*.

The Secure Apps Manager performs the following tasks to support AppConnect apps on Android devices:

- manages the data encryption key.
- handles the AppConnect passcode login for all AppConnect apps.
- provides a list of all the AppConnect apps on the device.

When a new Secure Apps Manager becomes available, you do not need to re-wrap all your apps. Secure Apps Manager is backward compatible. A wrapped app requires the corresponding or newer version of Secure Apps Manager. For example, an app wrapped with Wrapper 8.5.0.0 requires Secure Apps Manager 8.5.0.0 or later version that supports apps wrapped with Wrapper 8.5.0.0.

For MobileIron Cloud deployments, the Secure Apps Manager is bundled with MobileIron Go. The Secure Apps Manager is automatically installed on a device when you distribute an AppConnect app for Android to a device. The Secure Apps Manager is automatically updated to the latest version of Secure Apps Manager that MobileIron Cloud supports.

For the AppConnect app compatibility with the latest version of Secure Apps Manager, see the AppConnect for Android release notes available in the MobileIron AppConnect for Android Product Documentation Home Page.

NOTE: Support for various AppConnect for Android features sometimes require minimum versions of the MobileIron client app, Secure Apps Manager, and the wrapper, as specified in each feature's description.

Supported Android device processors

AppConnect on Android is supported on devices with:

- 32-bit ARM processors
- 64-bit ARM processors

Supported Android operating systems

For Android versions that AppConnect for Android supports, see the *AppConnect Secure Apps for Android Release Notes and Upgrade Guide*.

For Android versions that the MobileIron Cloud supports, see the release note for MobileIron Cloud.

However, some AppConnect for Android features require one of the more recent Android versions. These exceptions are noted in specific feature descriptions.

Samsung Knox container (Knox Workspace) and AppConnect apps

The Samsung Knox container, known as the Knox Workspace, is not supported with AppConnect apps. Specifically:

- The Samsung Knox container does not support any AppConnect apps running inside the Knox container.
- MobileIron does not support using both a Knox container and AppConnect container on the same device.

AppConnect for Android component support and compatibility

For the supported versions of the various components in an AppConnect deployment, including the Secure Apps Manager, MobileIron Go, and MobileIron Cloud, see the *MobileIron AppConnect for Android Release Notes and Upgrade Guide* in the MobileIron AppConnect for Android Product Documentation Landing Page.

Data loss prevention for secure apps for Android

Data loss prevention policies for secure apps allow you to secure the sensitive data in AppConnect apps. With data loss prevention policies, you determine whether:

- device users can take screen captures of protected data.
- AppConnect apps can access camera photos or gallery images.
- AppConnect apps can stream media to media players.
- AppConnect apps have copy/paste restrictions.
- tapping a web link in an AppConnect app can open the web page in an unsecured browser.
- tapping a web link in a non-AppConnect app can open the web page in Web@Work.

NOTE: Document interaction (Open In) is always restricted to all AppConnect apps for Android.

Data encryption for secure apps for Android

App data for AppConnect apps on the device is encrypted. AES-256 encryption (which uses a key size of 256 bits) is used.

The encryption key is not stored on the device. It is programmatically derived. If an AppConnect passcode is required, it is used in the encryption key's derivation, making the application data secure even on a device that becomes compromised. When a device is compromised, it is rooted.

Special badging for secure apps for Android

An Android device user recognizes that an app is a secure app because its icon is overlaid with a special badge.

AppConnect for Android apps

The following provides an overview of the types of AppConnect apps:

- Types of AppConnect Apps
- AppConnect apps that MobileIron provides for Android
- Other documentation about MobileIron-provided AppConnect apps

Types of AppConnect Apps

AppConnect for Android supports apps developed by:

- third-party developers
- in-house developers
- MobileIron

The apps can be:

- Java apps
- Hybrid web apps, including Cordova and PhoneGap apps

  Hybrid web apps use Android WebView and WebKit technologies to access and display web content.
- Java apps which use C or C++ code

  C and C++ code are native code languages on Android devices. These apps are built with the Android Native Development Kit (NDK)
- Apps built with the Xamarin development platform
- Apps built with the React Native mobile development framework

All apps are wrapped with the AppConnect for Android wrapper. All apps are distributed by uploading them to the App Catalog on MobileIron Cloud as in-house apps.

Wrapping does not support all Java APIs and features or all NDK features. Details are listed in the *MobileIron AppConnect for Android App Developers Guide*.

AppConnect apps that MobileIron provides for Android

MobileIron provides the following AppConnect apps. These apps are available on the **Software > Downloads** page at:

https://help.mobileiron.com/s/software

Docs@Work

The Docs@Work for Android app provides users with an easy way to access, annotate, share, and view documents across a variety of on-premise and cloud storage repositories (for example, SharePoint, CIFS, WebDAV, O365, Box, and Dropbox).

Email+

Email+ provides secure email, calendar, and contacts on corporate-owned and BYOD Android devices by communicating with an ActiveSync server in your enterprise.

Web@Work

Web@Work is a secure browser that allows your device users to easily and securely access your organization's web content.

File Manager

This secure File Manager allows a user to save, browse, and manage files in the secure container. For example, the user can browse saved email attachments. The user can also save documents from any other AppConnect app.

Other documentation about MobileIron-provided AppConnect apps

For more information about the AppConnect apps that MobileIron provides for Android, see:

- *MobileIron Docs@Work for Android Guide*
- *MobileIron Email+ for Android Guide*
- *MobileIron Web@Work for Android Guide*

When an Android device user can use AppConnect for Android

An Android device user can use an AppConnect app only if:

- The device user has been authenticated through a MobileIron Cloud via MobileIron Go.
- You have authorized the app to run on the device.
    - If the app is not authorized, the app does not allow the device user to access any secure data or functionality. If a device user launches an unauthorized wrapped app, the app displays a message and exits.
- No situation has caused an authorized AppConnect app to become unauthorized for a device.
    - These situations include, for example, when the device has been out of contact with Cloud for a period of time that you configure.
- The device user has entered the AppConnect passcode, if you have required one.

AppConnect for iOS overview

AppConnect for iOS apps are either:

- built using the AppConnect for iOS SDK
- wrapped

AppConnect functionality on iOS devices is provided by the AppConnect app and MobileIron Go for iOS.

Component support and compatibility

For the supported versions of the various components in an AppConnect deployment, including MobileIron Cloud and MobileIron Go, see “Product versions required” in either

- the *MobileIron AppConnect for iOS SDK App Developers Guide*
- the *MobileIron AppConnect for iOS App Wrapping Developers Guide*

See the guide that corresponds to the version of AppConnect with which the app is built or wrapped.

Wrapping support for mobile development platforms

Many iOS apps are created using mobile development platforms, rather than using the Apple environment that targets only iOS devices. You can wrap iOS apps that were created using these mobile development platforms:

- PhoneGap
- IBM Worklight
- Xamarin

Data loss prevention for secure apps for iOS

You determine whether an app can use the iOS pasteboard, the document interaction feature (Open In), copy-paste, or print. AppConnect for iOS uses this information to limit the app's functionality to prevent data loss through these features.

Data encryption for secure apps for iOS

The following describe the data encryption for secure apps for iOS:

- AppConnect-related data
- App-specific data

AppConnect-related data

AppConnect-related data, such as app configurations and certificates, is encrypted on the device. The encryption key is not stored on the device. It is either:

- Protected by the device user's AppConnect passcode.
- Protected by the device passcode if the administrator does not require an AppConnect passcode.

If no AppConnect passcode or device passcode exists, the data is encrypted, but the encryption key is not protected by either passcode.

App-specific data

Data that the app saves on the device is also protected with encryption. Specifically:

- For a wrapped app, if the device has a device passcode, then iOS encrypts the app's data.
  If no device passcode exists, iOS encrypts the data, but the encryption key is not protected.

- For an app built with the SDK or Cordova Plugin, if the app enables iOS data protection on its files, and the device has a device passcode, then iOS encrypts the app's data. Most apps enable iOS data protection, which is default app behavior.
  If no device passcode exists, iOS encrypts the data, but the encryption key is not protected.
- Some SDK apps use SDK-provided secure services. For these apps, the app's data is encrypted if the device has a device passcode or an AppConnect passcode.
  If no device passcode or AppConnect passcode exists, iOS encrypts the data, but the encryption key is not protected.

  NOTE: SDK apps that use SDK-provided secure services can also share encrypted data with other SDK apps. To do this, the app's documentation provides an encryption group ID key for you to include in the app's app-specific configuration. If you include the same value for an encryption group ID key for another AppConnect app, the apps can share the encrypted data.

Contact the app developer or vendor to determine whether the app enables iOS data protection, and whether SDK apps use the SDK-provided secure file I/O. This information contributes to your decisions to require an AppConnect passcode and device passcode.

The following table summarizes the protection of the data that AppConnect apps save on the device. Note that if a device user uses Touch ID or Face ID to access AppConnect apps, a device passcode is available.

TABLE 2. ENCRYPTION OF APPCONNECT APP DATA ON THE DEVICE

	Device passcode but no AppConnect passcode	**AppConnect passcode but no device passcode**	**Device passcode and AppConnect passcode**	**Neither a device passcode or AppConnect passcode**
Wrapped apps	App data encrypted	iOS encrypts the data, but the encryption key is not protected.	App data encrypted	iOS encrypts the data, but the encryption key is not protected.
SDK and Cordova apps that enable iOS data protection (typical behavior)	App data encrypted	iOS encrypts the data, but the encryption key is not protected.	App data encrypted	iOS encrypts the data, but the encryption key is not protected.
SDK apps that use SDK-provided secure services	App data encrypted	App data encrypted	App data encrypted	iOS encrypts the data, but the encryption key is not protected.

MobileIron Go for iOS and AppConnect apps

The MobileIron Go for iOS supports AppConnect apps, including the following:

- Periodically does an app check-in with the MobileIron Cloud to get management and security-related information and passes the information to the AppConnect app.
- Enforces the AppConnect passcode and Touch ID / Face ID for accessing AppConnect apps.

App check-in and MobileIron Go

On each app check-in, MobileIron Go gets AppConnect policy updates for all the AppConnect apps that have already run on the device. These updates include changes to:

- the AppConnect Device configuration for the device
- AppConnect app configurations for each of the AppConnect apps that have run on the device.
- the current authorization status for each of the AppConnect apps that have run on the device.

MobileIron Go does an app check-in in the following situations:

- The device user launches an AppConnect app for the first time.
  - In this situation, MobileIron Go finds out about the app for the first time, and adds it to the set of AppConnect apps for which it gets updates.
- The app check-in interval expires while an AppConnect app is running.
- The app check-in interval expired while no AppConnect apps were running and then the device user launches an AppConnect app.

On iOS devices, when MobileIron Go does an app check-in, it comes to the foreground and the AppConnect app goes to the background momentarily. Once MobileIron Go has completed the app check-in, the AppConnect app returns to the foreground.

NOTE: The Force Device Check-in feature on MobileIron Cloud does not sync the policies and settings related to AppConnect for iOS. The app check-in interval in the **AppConnect Device configuration** on MobileIron Cloud controls these updates. However, in MobileIron Go for iOS on the device, the **Check for Updates** option *does* sync the policies and settings related to AppConnect.

The AppConnect passcode auto-lock time and MobileIron Go

The MobileIron Go launches to prompt the device user for the AppConnect passcode or Touch ID / Face ID in the following situations:

- The device user launched or switched to an AppConnect app after the auto-lock time expired. You configure the auto-lock time in the AppConnect global policy.
- The AppConnect passcode auto-lock time expires while the device is running an AppConnect app.

  NOTE: If the device user is interacting with the app, the auto-lock time does not expire. This case occurs only when the device user has not touched the device for the duration of the timeout interval.
- After the device is powered on and the device user first launches an AppConnect app.

- The device user used MobileIron Go to log out of AppConnect apps, and then launches an AppConnect app.
- You have changed the complexity rules of the AppConnect passcode, and an app check-in occurs.

In each of these situations, the MobileIron Go launches, and presents the device user with a screen for entering his AppConnect passcode or Touch ID / Face ID. After the device user enters the passcode or Touch ID / Face ID, the device user automatically returns to the AppConnect app.

Related topics

Touch ID or Face ID for accessing secure apps

Dual-mode apps

Some apps that are built with the AppConnect for iOS SDK can behave as either an AppConnect-enabled app, or a regular, unsecured, standalone app. These apps are called dual-mode apps. For example, Email+ for iOS is a dual-mode app. As a dual-mode app, the same app can behave as a secure enterprise app for enterprise users, or as a regular app for general consumers.

A dual-mode app behaves as an AppConnect-enabled app on a device when:

- The device is registered to MobileIron Cloud and MobileIron Go is installed on the device.
- You have configured MobileIron Cloud to support AppConnect with the relevant AppConnect configurations.

Otherwise, the app behaves as a regular, unsecured, standalone app.

Regarding the decision to run as an AppConnect-enabled app versus a regular app:

- Some dual-mode apps allow the device user to change the app into an AppConnect-enabled app or regular app after having already run it the other way.
- Some dual-mode apps require the user to uninstall and reinstall the app to make this change.
- Some apps delay their decision to run as an AppConnect-enabled app or regular app until after MobileIron Go is installed on the device.

AppConnect apps that MobileIron provides for iOS

MobileIron provides the following AppConnect apps for iOS. These apps are available in the Apple App Store.

- Docs@Work
  Docs@Work provides device users an intuitive way to access, store, view, edit, and annotate documents from content repositories, such as Microsoft SharePoint, and cloud services like Box and Dropbox.
  For more information about Docs@Work, see the MobileIron Docs@Work Product Documentation Home Page.

- Web@Work

  Web@Work allows your users to easily and securely access your organization's web content.

  For more information about Web@Work, see the MobileIron Web@Work Product Documentation Home Page.

- Email+

  Email+ for iOS provides secure email, calendar, contacts, and tasks on iOS devices.

  For more information about Email+, see the MobileIron Email+ Product Documentation Home Page.

When an iOS device user can use AppConnect for iOS

An iOS device user can use an AppConnect app only if:

- The device user has been authenticated through MobileIron Cloud.

  The user must use the MobileIron Go for iOS app to register the device with MobileIron Cloud. Registration authenticates the device user.

- You have authorized the app to run on the device.

  If the app is not authorized, the app does not allow the device user to access any secure data or functionality. If a device user launches an unauthorized wrapped app, the app displays a message and exits. An SDK app (an app built with AppConnect for iOS SDK or Cordova Plugin) should have the same behavior if the app handles only secure data and functionality. Otherwise, an SDK app runs but restricts the user to only unsecured functionality and data.

  To authorize an AppConnect app for a device, you apply the appropriate labels to the app’s AppConnect container policy.

- No situation has caused an authorized AppConnect app to become unauthorized for a device.

  These situations include, for example, when the device OS is compromised. MobileIron Go reports device information to MobileIron Cloud. MobileIron Cloud then determines whether to change the AppConnect apps on the device to unauthorized based on security policies and associated compliance actions that you configure.

- The device user has entered the AppConnect passcode or Touch ID / Face ID.

  You configure whether the AppConnect passcode is required, and also configure rules about its complexity. You also configure whether the device user can use Touch ID or Face ID to access secure apps.

Quick start configuration AppConnect for Android

MobileIron provides the default configurations needed to quickly set up and distribute AppConnect apps to devices. MobileIron provides the following default configurations for AppConnect for Android:

- A default **AppConnect Device** configuration, **Default Android AppConnect Configuration**, which includes the default AppConnect passcode and data loss prevention (DLP) settings that are automatically applied to all devices.
- A default **Android AppConnect Passcode Service** configuration. The passcode service communicates the AppConnect passcocde service URL to Android devices. The configuration cannot be edited.

When you add an AppConnect app to MobileIron Cloud, the app settings provided by the AppConnect app are automatically available. No additional app configurations are needed. Therefore, to quickly distribute an AppConnect app, simply add and distribute the AppConnect app as you would any other in-house app.

See the following:

- Adding AppConnect apps to MobileIron Cloud

To customize the configuration for the AppConnect app, see the following:

- Adding an AppConnect Custom Configuration
- Adding an AppConnect Devices configuration
- Android AppConnect Devices field description

Adding AppConnect apps to MobileIron Cloud

You add Android AppConnect apps to MobileIron Cloud in the same manner you add any Android app. Apps are distributed to devices in the distribution list you selected when adding the app. To add AppConnect apps provided by MobileIron, in MobileIron Cloud, go to **Apps > App Catalog > +Add** and select **In-House**. The MobileIron AppConnect apps for Android are identified as **For Android AppConnect** in the app's tile.

Before you begin

1. Obtain the AppConnect apps for Android.
   - Check with the app developer for the location of in-house and third-party AppConnect apps.

   NOTE: The AppConnect apps for Android provided by MobileIron are available in the App

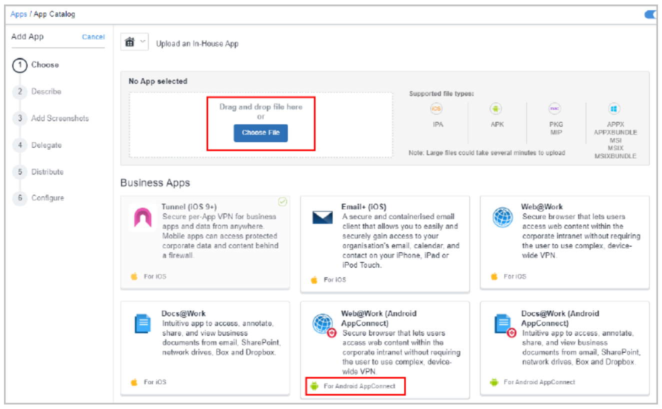
Catalog for upload to MobileIron Cloud.

2. Put the APK files where they are available for upload to MobileIron Cloud.

Procedure

1. In MobileIron Cloud, go to **Apps > App Catalog > Add+ > In-House**.

FIGURE 2. ADD APP THE APP CATALOG IN MOBILEIRON CLOUD



Note: Large files could take several minutes to upload

Tunnel (iOS 9+)
Secure per-App VPN for business apps and data from anywhere. Mobile apps can access protected corporate data and content behind a firewall.

Email+ (iOS)
A secure and containerised email client that allows you to easily and securely gain access to your organisation's email, calendar, and contact on your iPhone, iPad or iPod Touch.

Web@Work
Secure browser that lets users access web content within the corporate intranet without requiring the user to use complex, device-wide VPN.

Docs@Work
Intuitive app to access, annotate, share, and view business documents from email, SharePoint, network drives, Box and Dropbox.

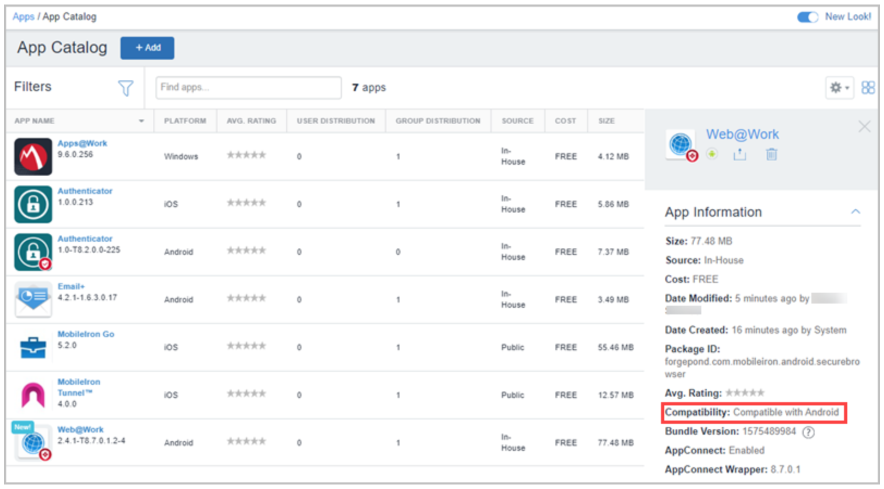
Web@Work (Android AppConnect)
Secure browser that lets users access web content within the corporate intranet without requiring the user to use complex, device-wide VPN.

Docs@Work (Android AppConnect)
Intuitive app to access, annotate, share, and view business documents from email, SharePoint, network drives, Box and Dropbox.

2. Choose the app to add.
   - To add an in-house app or third-party app, click **Choose File** to navigate to the APK file or drag and drop the APK file.
   - To add MobileIron apps, select the tile for the Android AppConnect app from apps available in **Business Apps**.
3. Click **Next**.
4. You can choose to keep the defaults or optionally, make selections for **App Information, Screenshots, App Delegation, and Distribution** by clicking **Next**.
5. Click **Done** to add the app to the App Catalog..
6. Select the app in the App Catalog to verify the app's compatibility.

   If the app uses an older version of AppConnect that is incompatible with MobileIron Cloud, a warning message is visible in the app's details. If the AppConnect version is not compatible, the app is not installed on Android devices.

FIGURE 3. VERIFY APPCONNECT APP COMPATIBILITY



Apps / App Catalog

New Look!

App Catalog + Add

Filters

Find apps...

7 apps

APP NAME	PLATFORM	AVG. RATING	USER DISTRIBUTION	GROUP DISTRIBUTION	SOURCE	COST	SIZE
Apps@Work 9.6.0.256	Windows	★★★★★	0	1	In-House	FREE	4.12 MB
Authenticator 1.0.0.213	iOS	★★★★★	0	1	In-House	FREE	5.86 MB
Authenticator 1.0-T8.2.0.0-225	Android	★★★★★	0	0	In-House	FREE	7.37 MB
Email+ 4.2.1-1.6.3.0.17	Android	★★★★★	0	1	In-House	FREE	3.49 MB
MobileIron Go 5.2.0	iOS	★★★★★	0	1	Public	FREE	55.46 MB
MobileIron Tunnel™ 4.0.0	iOS	★★★★★	0	1	Public	FREE	12.57 MB
New! Web@Work 2.4.1-T8.7.0.1.2-4	Android	★★★★★	0	1	In-House	FREE	77.48 MB

Web@Work

App Information

Size: 77.48 MB
Source: In-House
Cost: FREE
Date Modified: 5 minutes ago by [redacted]
Date Created: 16 minutes ago by System
Package ID: forgepond.com.mobileiron.android.securebrowser
Avg. Rating: ★★★★★
Compatibility: Compatible with Android
Bundle Version: 1575489984
AppConnect: Enabled
AppConnect Wrapper: 8.7.0.1

Next steps

- If your app requires any custom configurations, create an **AppConnect Custom Configuration**.
  See Adding an AppConnect Custom Configuration.
- To customize the passcode and data loss prevention (DLP) settings, edit the default **AppConnect Device** configuration or create a new **AppConnect Device** configuration. See Adding an AppConnect Devices configuration.

Related topics

- See the *MobileIron Cloud Administrator Guide* or Help for more information on adding apps to the MobileIron Cloud app catalog.
- See Android AppConnect Devices field description for the field descriptions and defaults for settings in the Android **AppConnect Device** configuration.
- To add an AppTunnel configuration, see Adding an AppTunnel configuration

Adding an AppConnect Custom Configuration

The settings in an Android AppConnect app are automatically available. To customize the app behavior, add certificates, or allow screen capture in the app, create an **AppConnect Custom Configuration**. You configure key-value pairs to customize the app behavior and add certificates for distribution.

Before you begin

- Check your app documentation for the key-value pairs supported by your app.
- For the supported key-value pairs see AppConnect Key-value Pairs.

Procedure

1. In MobileIron Cloud, go to **App > App Catalog**.
2. Click the app listing to edit the settings.
3. Go to **App Configurations > AppConnect Custom Configuration**.
4. Click **Add** to add a new **AppConnect Custom Configuration**.
5. Enter a name for the configuration.
6. Enter the key-value pairs for the desired configuration and certificates.
7. Optionally, select **Allow screen capture**.
8. Select a distribution option.
9. Click **Save**.

Related topics

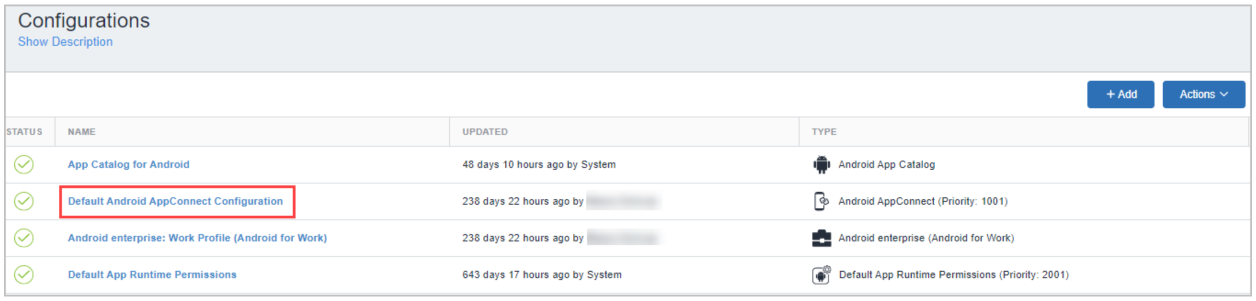
AppConnect for Android key-value pairs

Adding an AppConnect Devices configuration

Using AppConnect for Android requires that an AppConnect Devices configuration is set up. This configuration specifies settings that are not specific to a particular AppConnect app such as the AppConnect passcode requirements and data loss protection(DLP) requirements.

MobileIron provides a default AppConnect Devices configuration, **Default Android AppConnect Configuration**, which is by default applied to all devices. You can either edit the default configuration for you specific deployment or create a new configuration.

FIGURE 4. DEFAULT APPCONNECT ANDROID DEVICES CONFIGURATION

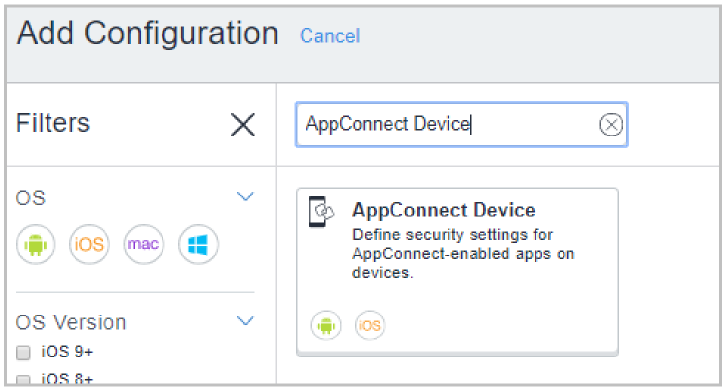


STATUS	NAME	UPDATED	TYPE
✓	App Catalog for Android	48 days 10 hours ago by System	Android App Catalog
✓	Default Android AppConnect Configuration	238 days 22 hours ago by [redacted]	Android AppConnect (Priority: 1001)
✓	Android enterprise: Work Profile (Android for Work)	238 days 22 hours ago by [redacted]	Android enterprise (Android for Work)
✓	Default App Runtime Permissions	643 days 17 hours ago by System	Default App Runtime Permissions (Priority: 2001)

Procedure

1. In your instance of MobileIron Cloud, go to **Configurations > +Add**.
2. Enter AppConnect Devices in the **Search Configuration** text box to filter quickly to the configuration.

FIGURE 5. ADD APPCONNECT DEVICE CONFIGURATION



3. Click the tile for **AppConnect Device**.
4. Select **Android** to diplay the settings for the **AppConnect Device** configuraiton for Android.
5. Update the settings as needed and click **Next**.
6. Select a distribution option and click **Done**.

Next steps

To add an AppTunnel configuration, see Configuring AppTunnel for AppConnect apps

Related topics

Android AppConnect Devices field description

Android AppConnect Devices field description

The following table describes the settings in the **AppConnect Devices** configuration.

TABLE 3. ANDROID APPCONNECT DEVICES SETTINGS DESCRIPTION

Setting	What To Do
Name	Enter a name that identifies this configuration.
Description	Enter a description that clarifies the purpose of this configuration.
AppConnect Passcode	
Enable Secure Apps Passcode	Select to require users to enter their secure apps passcode before accessing AppConnect apps.
Numeric	Select to allow the passcode to have only digits in it. However, the user can choose to create an alphanumeric passcode.
Alphanumeric	Select to require the passcode to contain at least one digit and one letter.
Don't specify	Select to allow the passcode to have characters of any type.

TABLE 3. ANDROID APPCONNECT DEVICES SETTINGS DESCRIPTION (CONT.)

Setting	What To Do
Minimum passcode length	Select the minimum number of characters required.
Minimum number of complex characters	For alphanumeric passcodes, select the minimum number of complex characters required. NOTE: A complex character is any character which is not 0-9, a-z, or A-Z. For example, $, \, and ä are special characters.
Maximum Password Age	Select an age from the list or select Custom to enter a specific number of days after which the user must change the secure apps passcode.
Auto-Lock	Select the amount of time that passes before the AppConnect auto-lock feature requires the user to re-enter the secure apps passcode.
Passcode history	Enter the number of unique secure apps passcodes that the user must enter before repeating a passcode. For example, if you set this option to 3, then the user must use 3 different passcodes when resetting the secure apps passcode before being able to reuse the first passcode.
Maximum number of failed attempts	Select a value between 1 and 10. Select “None” if you do not want to limit failed attempts. If the device user fails to correctly enter the AppConnect passcode after a certain number of attempts, the user cannot access AppConnect apps.
Allow user to recover passcode	Select to allow the user to recover passcode.
Enable Fingerprint Authentication	Drag the slider to the right to switch **ON** the option to allow the user to use fingerprint for authentication. By default, the option is **OFF**.
App Authorization	
Unauthorized message	Enter the default message that is displayed to the user if the app is not authorized on the device. If you do not enter a default message, the system provides one.
Data Loss Prevention Settings	
Copy/Paste • No restrictions • Among AppConnect apps • Within an AppConnect app	Select if you want the device user to be able to copy content from AppConnect apps to other apps. You can override this option in each app’s individual AppConnect container policy. When you select this option, then select either: • **No restrictions** Select if you want the device user to be able to copy content from the AppConnect app and paste it into any other app. • **Among AppConnect apps** Select AppConnect apps if you want the device user to be able to copy content from the AppConnect app and paste it only into other AppConnect apps.

TABLE 3. ANDROID APPCONNECT DEVICES SETTINGS DESCRIPTION (CONT.)

Setting	What To Do
	• **Within an AppConnect app** Select if you want the device user to be able to copy content from the AppConnect app and paste it only into the same AppConnect app.
Allow Camera	Select to allow camera photo access for all the AppConnect apps on an Android device.
Allow Gallery	Select to allow all the AppConnect apps on an Android device to access images from the gallery.
Allow Media Player	Select to allow all the AppConnect apps to stream media to media players.
Allow Screen Capture	Select to allow AppConnect apps to do screen capture.
Allow web	Select to allow an unsecured browser to attempt to display a web page when a device user taps the page's URL in a secure app. If you do not select Allow web, only Web@Work can display the page.
Allow Non-AppConnect apps to open URLs in Web@Work	Select to allow device users to choose to view a web page in Web@Work or other AppConnect-enabled browser when they tap a link (URL) in an app that is not AppConnect-enabled.

Quick start configuration AppConnect for iOS

MobileIron provides the default configurations needed to quickly set up and distribute AppConnect apps to devices. MobileIron provides the following default configurations for AppConnect for iOS:

- A default **AppConnect Device** configuration, **Default iOS AppConnect Configuration**, which includes the default AppConnect passcode and data loss prevention (DLP) settings that are automatically applied to all devices.

When you upload an AppConnect app to MobileIron Cloud, the app settings provided by the AppConnect app are automatically available. No additional app configurations are needed. Therefore, to quickly distribute an AppConnect app, simply add and distribute the AppConnect app as you would any other in-house app.

Adding AppConnect apps to MobileIron Cloud

The MobileIron AppConnect apps and many third-party AppConnect apps for iOS are available in the Apple App Store. By adding them to the MobileIron Cloud App Catalog, you can distribute them to devices using Apps@Work.

You add iOS AppConnect apps to MobileIron Cloud in the same manner you add any iOS apps. Apps are distributed to devices in the distribution list you selected when adding the app. To add AppConnect apps from the Apple App Store, go to **Apps > App Catalog > +Add**.

Before you begin

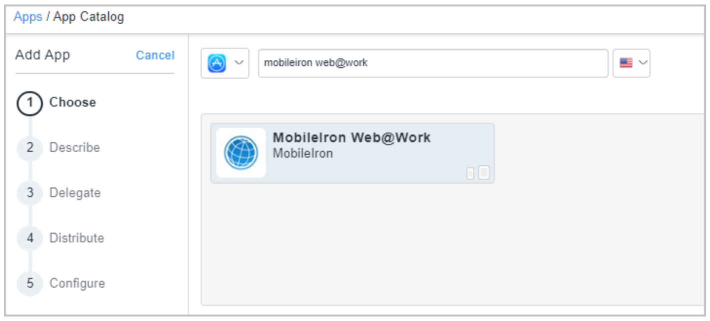
If you are adding in-house AppConnect apps, obtain the apps. Check with the app developer for the location of in-house and third-party AppConnect apps. Place the IPA files where they are available for upload to MobileIron Cloud.

NOTE: The AppConnect apps for iOS provided by MobileIron are available in the Apple App Store.

Procedure

1. In MobileIron Cloud, go to **Apps > App Catalog > iOS Store**.
   Or, to add an in-house app, go to **Apps > App Catalog > Add+ > In-House**.
2. In the search box enter the name of the app to add.
   Or, to add an in-house app, click **Choose File** to navigate to the IPA file or drag and drop the IPA file.

FIGURE 6. FIND IOS APPCONNECT APP IN THE APPLE APP STORE



3. Select the app to add, and click **Next**
4. You can choose to keep the defaults or optionally, make selections for **App Information, Screenshots, App Delegation, and Distribution** by clicking **Next**.
5. Click **Done** to add the app to the App Catalog.

Next steps

- If your app requires any custom configurations, create an **AppConnect Custom Configuration**.

  See Adding an AppConnect Custom Configuration.
- To customize the passcode and data loss prevention (DLP) settings, edit the **Default iOS AppConnect Configuration** configuration or create a new **AppConnect Device** configuration. See Editing AppConnect Devices configuration.

Related topics

- See the *MobileIron Cloud Administrator Guide* or help for more information on adding apps to the MobileIron Cloud app catalog.
- See Quick start configuration AppConnect for iOS for the field descriptions and defaults for settings in the iOS AppConnect Device configuration.
- To add an AppTunnel configuration, see Adding an AppTunnel configuration

Adding an AppConnect Custom Configuration

The settings in an iOS AppConnect app are automatically available. To customize the app behavior, add certificates, or allow screen capture in the app, create an **AppConnect Custom Configuration**. You configure key-value pairs to cusomize the app behavior and add certificates for distribution.

Before you begin

- Check your app documentation for the key-value pairs supported by your app.
- For the supported key-value pairs see AppConnect Key-value Pairs.

Procedure

1. In MobileIron Cloud, go to **App > App Catalog**.
2. Click the app listing to edit the settings.
3. Go to **App Configurations > AppConnect Custom Configuration**.
4. Click **Add** to add a new **AppConnect Custom Configuration**.
5. Enter a name for the configuration.
6. Enter the key-value pairs for the desired configuration and certificates.
7. Optionally, select **Allow screen capture**.
8. Select a distribution option.
9. Click **Save**.

Related topics

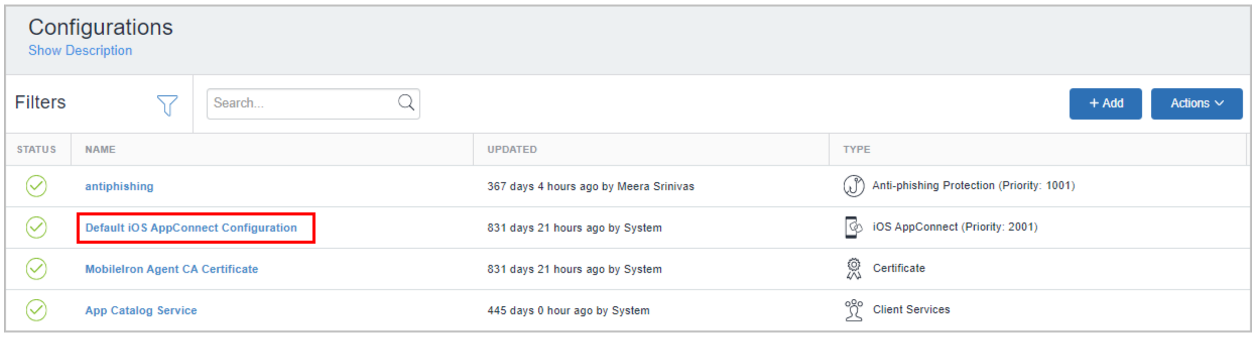
AppConnect for iOS key-value pairs

Editing AppConnect Devices configuration

Using AppConnect for iOS requires that an AppConnect Devices configuration is set up. This configuration specifies settings that are not specific to a particular AppConnect app such as the AppConnect passcode requirements and data loss protection(DLP) requirements.

MobileIron provides a default AppConnect Devices configuration, **Default iOS AppConnect Configuration**, which is by default applied to all devices. You can either edit the default configuration for you specific deployment or create a new configuration.

FIGURE 7. DEFAULT APPCONNECT IOS DEVICES CONFIGURATION

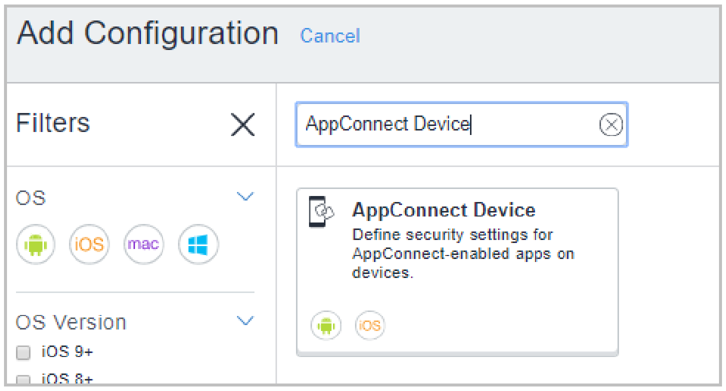


STATUS	NAME	UPDATED	TYPE
✓	antiphishing	367 days 4 hours ago by Meera Srinivas	Anti-phishing Protection (Priority: 1001)
✓	Default iOS AppConnect Configuration	831 days 21 hours ago by System	iOS AppConnect (Priority: 2001)
✓	MobileIron Agent CA Certificate	831 days 21 hours ago by System	Certificate
✓	App Catalog Service	445 days 0 hour ago by System	Client Services

Procedure

1. In your instance of MobileIron Cloud, go to **Configurations > +Add**.
2. Enter AppConnect Devices in the **Search Configuration** text box to filter quickly to the configuration.

FIGURE 8. ADD APPCONNECT DEVICE CONFIGURATION



3. Click the tile for **AppConnect Device**.
4. Select **iOS** to diplay the settings for the **AppConnect Device** configuraiton for iOS.
5. Update the settings as needed and click **Next**.
6. Select a distribution option and click **Done**.

Next steps

To add an AppTunnel configuration, see Configuring AppTunnel for AppConnect apps

Related topics

iOS AppConnect Devices field description

iOS AppConnect Devices field description

The following table describes the settings in the **AppConnect Devices** configuration.

TABLE 4. IOS APPCONNECT DEVICES FIELD DESCRIPTION

Setting	What To Do
Name	Enter a name that identifies this configuration.
Description	Enter a description that clarifies the purpose of this configuration.
AppConnect Passcode	
Enable Secure Apps Passcode	Select to require users to enter their secure apps passcode before accessing AppConnect apps.
4-digit numeric	Select to allow the passcode to have only 4 digits in it.
Alphanumeric	Select to require the passcode to contain at least one digit and one letter.
Maximum Password	Select an age from the list or select Custom to enter a specific number of days after which the user must change the secure apps passcode.

TABLE 4. IOS APPCONNECT DEVICES FIELD DESCRIPTION (CONT.)

Setting	What To Do
Age	
Auto-Lock	Select the maximum amount of time to allow as an inactivity timeout. After this period of inactivity in AppConnect apps, the device user is locked out of the apps if an AppConnect passcode is required. The device user must reenter the AppConnect passcode to access AppConnect apps.
Passcode history (1-50 passcodes)	Select a value from 1 to 50. This value specifies the number of most recently used secure apps passcodes that the device user cannot use when changing his passcode. By default, no value is set. In this case, the user can reuse any previous passcode, including the current passcode.
Maximum number of failed attempts	Select a value between 4 and 10. Select "--" if you do not want to limit failed attempts. If the device user fails to correctly enter the AppConnect passcode after a certain number of attempts, the user cannot access AppConnect apps.
Enable Touch ID	Slide the toggle to ON to allow device users to enter their Touch ID (fingerprint) or Face ID, if available, to access secure apps.
App Authorization	
App check-in interval	Enter the number of minutes the app should wait before checking in with MobileIron Cloud to receive AppConnect-related configuration updates. Note that app authorization is an automatic result of adding an app to the app catalog.
Unauthorized message	Enter the default message that is displayed to the user if the app is not authorized on the device. If you do not enter a default message, the system provides one.
Device Out of Contact	
Wipe AppConnect device after	Enter the number days (1-90) that the device can remain out of contact before having its AppConnect data wiped. Enter 0 to disable this option. Once the configuration is applied to the device, wiping the AppConnect apps occurs on the device after the specified time without reconnecting to MobileIron Cloud.
Block AppConnect data after	Enter the number days (1-90) that the device can remain out of contact before having its AppConnect data blocked. Enter 0 to disable this option. Once the configuration is applied to the device, blocking the AppConnect apps occurs on the device after the specified time without reconnecting to MobileIron Cloud.
Data Loss Prevention Settings	

TABLE 4. IOS APPCONNECT DEVICES FIELD DESCRIPTION (CONT.)

Setting	What To Do
Allow copy/paste to	Select to if you want the device user to be able to copy content from AppConnect apps to other apps. You can override this option in each app's individual AppConnect container policy.
Allow printing	Select if you want AppConnect apps to be allowed to use print capabilities by default. You can override this option in each app's individual AppConnect container policy.
Allow open-in	Select if you want AppConnect apps to be allowed to use the Open In (document interaction) feature by default. When you select this option, then select either: • **All apps** Select if you want the app to be able to send documents to any other app. • **Whitelist Apps only** Select if you want the app to be able to send documents only to the apps that you specify. Enter the name of each app in your App catalog to Whitelist, one per line, or in a semi-colon delimited list. **Example** com.myAppCo.myApp1 com.myAppCo.myApp2;com.myAppCo.myApp3

AppConnect for Android

The following provides information about AppConnect features and how to enable the features:

Hybrid web app support

Android Secure Apps supports full containerization for AppConnect-enabled hybrid web apps on devices. A *hybrid web app* is an Android app (APK file) that the device user installs on the device, unlike a *pure web app* that the user accesses through a web browser. A hybrid web app includes at least one screen that displays a web page. Phonegap apps are a type of hybrid web app.

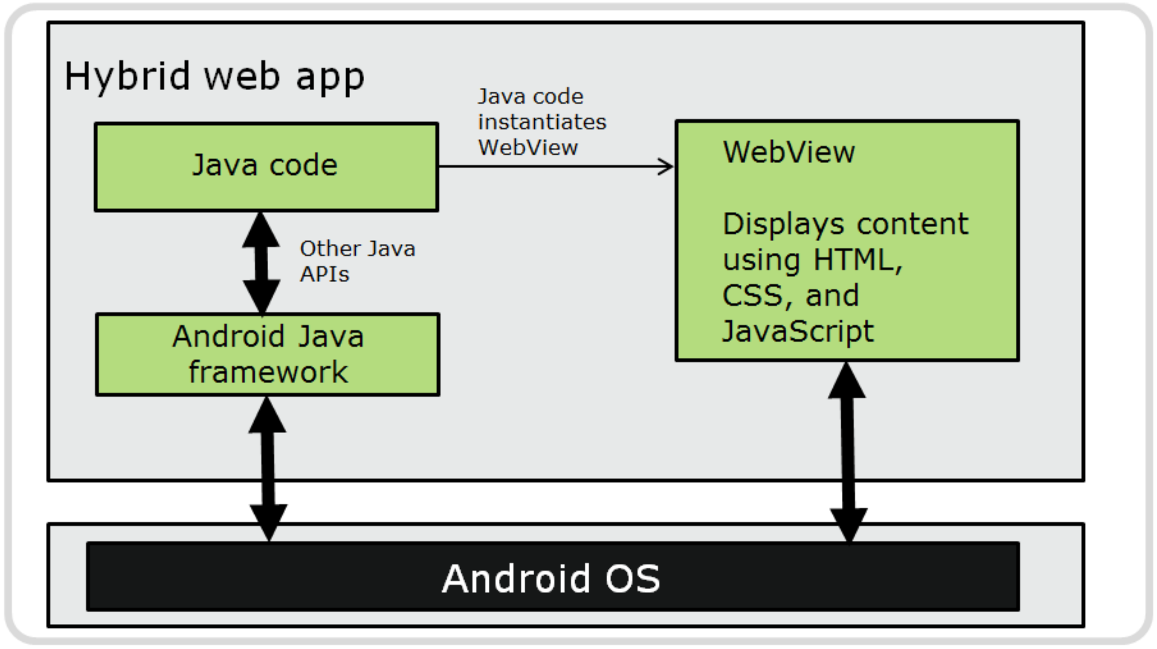
NOTE: Web@Work for Android, the secure browser that MobileIron provides, allows you to run pure web apps in the AppConnect secure container.

In a hybrid web app, business logic and content presentation occurs using Android WebView and WebKit technologies, specifically within an object of the Java class android.Webkit.WebView. The WebView object locally renders content using web technologies such as HTML, CSS, and JavaScript. The WebView object can access the web content from a network resource or from embedded web content.

Like other app data, data related to the android.webkit.WebView class is encrypted. This web-related data can include cookies, the web cache, and web databases.

The following diagram illustrates a hybrid web app on an Android device.

FIGURE 9. A HYBRID WEB APP ON AN ANDROID DEVICE


Hybrid web app
Java code
Java code instantiates WebView
WebView
Displays content using HTML, CSS, and JavaScript
Other Java APIs
Android Java framework
Android OS

Fingerprint login for AppConnect apps for Android

Fingerprint login for AppConnect apps gives the device user the convenience of using a fingerprint instead of an AppConnect passcode to access AppConnect apps. When using fingerprint, a user still creates an AppConnect passcode. If entering the fingerprint fails, the user enters the AppConnect passcode to access AppConnect apps.

The Secure Apps Manager gives the device user the choice to use fingerprint or an AppConnect passcode. This choice is useful when a device is shared among multiple users, such as co-workers or even a family, each of whom uses a fingerprint to access the device. Although all the users can access the device with fingerprint, sometimes only one of those users should be allowed to access AppConnect apps. That user can choose to use the AppConnect passcode instead of fingerprint for accessing AppConnect apps. Having a choice therefore ensures that only an appropriate device user accesses AppConnect apps.

Required product versions for fingerprint login for AppConnect for Android

The following table shows the required product versions for fingerprint login for Android secure apps.

TABLE 5. REQUIRED PRODUCT VERSIONS FOR FINGERPRINT LOGIN FOR SECURE APPS

Product	Version
MobileIron Go for Android	40 through the most recently released version as supported by MobileIron.
Secure Apps Manager	7.6.0 through the most recently released version as supported by MobileIron.
Android	6.0 through the most recently released version as supported by MobileIron

Requirements for fingerprint login for AppConnect for Android

Device users can use a fingerprint to access AppConnect apps for Android if the following are true:

- The product versions meet the requirements in Required product versions for fingerprint login for AppConnect for Android.
- The device has a fingerprint reader.
- The fingerprint option is set as follows in the MobileIron Cloud:
  - The fingerprint option is enabled in the AppConnect Device configuration for Android.
  - The fingerprint unlock option is enabled in the Passcode Config.

    NOTE: If fingerprint unlock option is disabled, enabling the fingerprint option in the AppConnect Device configuration has no impact.

If all of the above are true, Secure Apps Manager gives device users the choice whether to use fingerprint or use an AppConnect passcode to access AppConnect apps.

NOTE: In addition to choosing fingerprint, device users also create an AppConnect passcode. The AppConnect passcode is necessary if fingerprint login fails.

Configuring fingerprint login for AppConnect for Android (Cloud)

Configure fingerprint login for AppConnect apps on the MobileIron Cloud.

Procedure

1. On MobileIron Cloud, go to **Configurations**.
2. Select the appropriate AppConnect Device configuration for Android.
3. Click **Edit**.
4. For **Enable Fingerprint Authentication**, drag the slider to the right to **ON** .
5. Click **Next** and **Done**.
6. Go to **Configurations > Passcode Config**.
7. Click **Edit**.

8. Scroll down to the **Android only** section.
9. Verify that **Fingerprint Unlock** is **ON**.

Device User impact of fingerprint login for AppConnect for Android

If the requirements to use fingerprint login for AppConnect apps are fulfilled, the Secure Apps Manager gives device users the choice to use fingerprint or to use the AppConnect passcode for logging into AppConnect apps.

For more information about device user requirements, see Requirements for fingerprint login for AppConnect for Android

NOTE: The AppConnect passcode is called the secure apps passcode in the Secure Apps Manager.

The followig describe the device user experience:

- Device user experience at registration
- Device user experience if already registered
- Device user options for enabling or disabling fingerprint login

Device user experience at registration

The overall device user experience at registration is:

1. The Secure Apps Manager prompts the device user to create a secure apps passcode.
2. After creating the secure apps passcode, the Secure Apps Manager gives the user the option to use fingerprint to log into secure apps.
   If no fingerprint is available, the Secure Apps Manager prompts the user to add a fingerprint in the device's settings. The device user can then return to the Secure Apps Manager to enable fingerprint login.
3. If the user chooses the fingerprint option, he can use any fingerprint on the device for subsequent logins to secure apps.
4. If the user does not choose the fingerprint option, he will use the secure apps passcode for subsequent logins to secure apps.
5. The device user can at any time use a menu option in the Secure Apps Manager to change the choice about using fingerprint.

Device user experience if already registered

If you enable fingerprint login on the MobileIron Cloud after a device user is registered and has already created a secure apps passcode:

1. The next time the user logs into secure apps, the Secure Apps Manager prompts the device user to change the secure apps passcode.
2. After changing the secure apps passcode, the Secure Apps Manager gives the user the option to use fingerprint to log into secure apps.

If no fingerprint is available, the Secure Apps Manager prompts the user to add a fingerprint in the device's settings. The device user can then return to the Secure Apps Manager to enable fingerprint login.

3. If the user chooses the fingerprint option, he can use any fingerprint on the device for subsequent logins to secure apps.
4. If the user does not choose the fingerprint option, he will use the secure apps passcode for subsequent logins to secure apps.
5. The device user can at any time use a menu option in the Secure Apps Manager to change the choice about using fingerprint.

Device user options for enabling or disabling fingerprint login

When the Secure Apps Manager gives the user the option to use fingerprint to log into secure apps:

- If a fingerprint is available on the device, the user chooses one of the following:
    - to enable fingerprint login to secure apps immediately
    - to be reminded to enable it later
    - to never be reminded again
- If no fingerprint exists on the device, the user can choose to go to the device's settings to add a fingerprint. After adding the fingerprint, the user can return to the Secure Apps Manager to enable fingerprint login.

The device user can:

- At any time, use the options menu in Secure Apps Manager to disable or enable fingerprint login to secure apps.
- When fingerprint login is disabled, tap on **Enable Fingerprint Login** on the screen for entering the secure apps password.

In both of the above cases, the Secure Apps Manager prompts the device user to enter the secure apps passcode before changing the fingerprint login status.

Less common device user scenarios for fingerprint login for AppConnect for Android

These scenarios describe the device user experience in less common scenarios relating to fingerprint login to Android secure apps.

TABLE 6. LESS COMMON DEVICE USER SCENARIOS RELATING TO FINGERPRINT LOGIN

Scenario	Behavior on the device
Device has more than one fingerprint.	Any fingerprint can log into secure apps when fingerprint login is enabled.
Fingerprint login to secure apps fails due to too many attempts.	The Secure Apps Manager prompts the user for the secure apps passcode. NOTE: The Android OS controls the number of fingerprint login attempts.
The device user taps **Cancel** on the **Fingerprint Login** dialog for logging into secure apps.	The Secure Apps Manager prompts the user for the secure apps passcode.
A device user adds a fingerprint and a device passcode to the device, but does not enable fingerprint login for the device. NOTE: This scenario is possible only on some device models, such as some Samsung devices.	Fingerprint login is available for secure apps although it is not available for device login.
A device user adds a fingerprint to the device, but does not add a device passcode. NOTE: This scenario is possible only on some device models, such as some Samsung devices.	If you have configured fingerprint login for secure apps, the Secure Apps Manager prompts the user to go to settings. In the settings, the user must add a device passcode.
A device user adds a fingerprint to the device without enabling fingerprint login for the device. NOTE: This scenario is not possible on some device models.	Fingerprint login is available for secure apps although it is not available for device login.
The device user changes the secure apps passcode while fingerprint login is enabled for secure apps.	Fingerprint login remains enabled for secure apps.
The device user changes the secure apps passcode while fingerprint login is available, but disabled, for secure apps.	The Secure Apps Manager gives the device user the option to enable fingerprint login.

TABLE 6. LESS COMMON DEVICE USER SCENARIOS RELATING TO FINGERPRINT LOGIN (CONT.)

Scenario	Behavior on the device
1. Fingerprint login is available for secure apps. 2. A device user creates a new secure apps passcode because the user forgot the passcode.	The device user must again choose whether to enable fingerprint login. NOTE: This case applies when the device user initiates the "forgot passcode" scenario or the administrator unlocks the AppConnect container.
The device user restarts the device.	The device user must enter the secure apps passcode on the next secure apps login, even if fingerprint login had been enabled. The device user can use fingerprint login on subsequent logins to secure apps.
The device user terminates the Secure Apps Manager.	The device user must enter the secure apps passcode on the next secure apps login, even if fingerprint login had been enabled. The device user can use fingerprint login on subsequent logins to secure apps.
You enable or disable the **Use fingerprint authentication when supported** option on the AppConnect global policy.	The Secure Apps Manager prompts the device user to change the secure apps passcode after the user next logs in. This behavior is similar to changing any of these secure apps passcode characteristics on the AppConnect global policy: - passcode type - minimum passcode length - minimum number of complex characters - passcode strength usage or level changes NOTE: The device user can use a fingerprint to log in one last time when you disable the **Use fingerprint authentication when supported** option. After logging in, the Secure Apps Manager notifies the device user that the administrator disabled fingerprint login.
You change the **Block Fingerprint** option on the security policy.	The Secure Apps Manager prompts the device user to change the secure apps passcode after the user next logs in. NOTE: If your change is to block fingerprint, when the device user next logs into secure apps, the user cannot use a fingerprint to login. The Secure Apps Manager notifies the device user that the administrator disabled fingerprint login.

Security versus convenience of passcode and fingerprint for AppConnect for Android

AppConnect for Android security involves:

- access to AppConnect apps.
- encrypting AppConnect-related data such as app configurations, certificates, and data that the app saves on the device.

The following table lists possible passcode and fingerprint choices **from most secure to least secure**, and discusses the level of device user convenience. It compares the choices you can make on MobileIron Cloud involving:

- Whether you require a device passcode .
- Whether you require an AppConnect passcode.
- When requiring an AppConnect passcode, whether you allow fingerprint login to AppConnect apps.

The security level is impacted by the following:

- An AppConnect passcode ensures that AppConnect app data is encrypted and secure if the device is compromised (rooted). Without an AppConnect passcode, AppConnect app data is encrypted, but **not** secure if the device is compromised.
- A device passcode adds a layer of security.
- Fingerprint login allows all users of the same device who have added fingerprints to access the device and AppConnect apps. This access is a possible security risk.

NOTE: In all cases, stronger passcodes are more secure than weaker passcodes (such as a 4-digit number).

TABLE 7. SECURITY VERSUS DEVICE USER CONVENIENCE OF PASSCODE AND FINGERPRINT OPTIONS

Passcode and fingerprint configuration on MobileIron Cloud	**Security of AppConnect apps**	**Convenience for device user**
Device passcode: Required AppConnect passcode: Required Fingerprint: Not allowed	Highest	Least convenient for accessing both the device and AppConnect apps.
Device passcode: Not required AppConnect passcode: Required Fingerprint: Not allowed	Very High	Convenient for accessing the device but inconvenient for accessing AppConnect apps.
Device passcode:	High	Convenient for accessing both the

TABLE 7. SECURITY VERSUS DEVICE USER CONVENIENCE OF PASSCODE AND FINGERPRINT OPTIONS (CONT.)

Passcode and fingerprint configuration on MobileIron Cloud	Security of AppConnect apps	Convenience for device user
Required AppConnect passcode: Required Fingerprint: Allowed		device and AppConnect apps.
Device passcode: Not required AppConnect passcode: Required Fingerprint: Allowed	Lower	Very convenient for accessing the device, and convenient for accessing AppConnect apps.
Device passcode: Required AppConnect passcode: Not required Fingerprint: Not allowed	Low	Convenient for accessing AppConnect apps, but inconvenient for accessing the device.
No passcodes required	Lowest	Most convenient for accessing both the device and AppConnect apps. However, unauthorized users also have access.

Related topics

- The AppConnect passcode
- Data encryption for secure apps for Android

Lock, unlock, and retire impact on AppConnect for Android

Locking or retiring an Android device impacts access to AppConnect apps and their associated data. Also, unlocking the AppConnect container impacts access to AppConnect apps.

Lock impact

When you lock a device from MobileIron Cloud, the device user is also locked out of AppConnect apps. The user must reenter the secure apps passcode (or fingerprint) to access AppConnect apps. The Secure Apps Manager prompts the user to reenter the passcode when the user launches:

- the Secure Apps Manager
- any AppConnect app

If the device also uses a device passcode, the user must first reenter the device passcode (or other identification, such as a fingerprint).

Related topics

- For MobileIron Cloud deployments: Locking a device in the *MobileIron Cloud Administrator Guide*

Unlock the AppConnect container impact

When you unlock a device from MobileIron Cloud, the device passcode is removed. However, the AppConnect passcode is not impacted. Unlocking the AppConnect container using the **AppConnect Unlock**(Cloud) command removes the secure apps passcode. The Secure Apps Manager notifies the device user to create a new secure apps passcode when the user launches:

- MobileIron Go
- Secure Apps Manager
- any AppConnect app

No data relating to AppConnect apps is removed when the AppConnect container is unlocked. Once the device user creates a new secure apps passcode, the data becomes accessible again.

Issuing the AppConnect unlock command is useful in the following scenarios:

- You do not allow self-service AppConnect passcode recovery, and the device user has forgotten their secure apps passcode.
- The device user has exceeded the maximum number of failed attempts for the secure apps passcode.

Related topics

- MobileIron Cloud deployments: "Unlocking a Device," "Unlocking Android devices," Unlocking AppConnect for Android app," in the *MobileIron Cloud Administrator Guide*

Retire impact

Retiring a device unregisters the device from MobileIron Cloud.

Retiring a device impacts AppConnect apps as follows:

- The device user cannot open any AppConnect app or the Secure Apps Manager.
- Data that the AppConnect apps saved to device storage is deleted.

- In MobileIron Cloud deployments, the AppConnect container is removed from the device.

However, device users must manually uninstall the AppConnect apps and the Secure Apps Manager. On Samsung KNOX devices, Secure Apps Manager is uninstalled automatically.

Retiring a device, therefore, *retires* the AppConnect apps on the device.

Related topics

- MobileIron Cloud deployments: "Retiring a device" in the *MobileIron Cloud Administrator Guide*

Copy/Paste for AppConnect for Android

You configure the copy/paste DLP policy for AppConnect for Android in the AppConnect Device configuration on MobileIron. You can choose no restrictions for copy/paste, copy/paste only among AppConnect apps, or copy/paste only within each AppConnect app.

Each row of the following table summarizes whether copy/paste is allowed for a set of apps depending on the copy/paste setting:

TABLE 8. APPCONNECT GLOBAL POLICY COPY/PASTE DLP SETTING FOR ANDROID

	Copy/Paste setting in AppConnect global policy		
	No restrictions	**Among AppConnect apps**	**Within an AppConnect app**
Between an AppConnect app and an unsecured app	Allowed	Not allowed	Not allowed
Between different AppConnect apps	Allowed	Allowed	Not allowed
Within each AppConnect app	Allowed	Allowed	Allowed
Between different unsecured apps	Allowed	Allowed	Allowed
Within each unsecured app	Allowed	Allowed	Allowed

Comparison with AppConnect for iOS copy/paste policy

The copy/paste policy behavior differs between AppConnect for Android and iOS. The following table highlights some differences.

TABLE 9. COMPARISON WITH APPCONNECT FOR IOS COPY/PASTE POLICY

	AppConnect for Android	**AppConnect for iOS**
Symmetrical versus one-way	Copy/paste restrictions are symmetrical. For example, if you restrict copy/paste to among AppConnect apps, you cannot copy out of an AppConnect app into a unsecured app, and you cannot copy out of an unsecured app into an AppConnect app.	Copy/paste restrictions are one-way. The iOS Copy/Paste To DLP setting prohibits copying out of an AppConnect app, or prohibits copying out of an AppConnect app into an unsecured app. However, you can copy *from* an unsecured app *into* an AppConnect app.
Restriction levels	The copy/paste policy provides these restriction levels: • No copy/paste restrictions • Allow copy/paste only among AppConnect apps. • Allow copy/paste only within an AppConnect app.	The iOS Copy/Paste To DLP setting provides these restriction levels: • Do not allow copying from an AppConnect app. • Allow copying from an AppConnect app to any other app. • Allow copying from an AppConnect app only to other AppConnect apps.
Default setting in AppConnect global policy	The default copy/paste option is no restrictions. This behavior is consistent with the behavior of your AppConnect for Android installed base.	The default option is to not allow the user to copy data from AppConnect apps.

Copying from non-AppConnect apps to AppConnect apps

When the **Copy/Paste** DLP setting is either **Among AppConnect Apps** or **Within an AppConnect app**, you can also allow device users to copy data from a non-AppConnect app to an AppConnect app. That is, the device user can copy data **into** the AppConnect container, but cannot copy data **out** of the container.

To allow users to copy data from a non-AppConnect app to an AppConnect app, add the following key-value pair

- Key: **MI_ALLOW_SECURE_COPY_INBOUND**
- Value: **true**

You add the key-value pair in the **AppConnect Custom Configuration** for the app.

Interaction with Exchange setting

The Exchange setting for a device has a copy/paste option for Email+ for Android. This option allows or disables the use of copy/paste commands in these apps. The option applies to both the AppConnect-enabled version and the unsecured version of these apps.

If the Exchange setting disables copy/paste commands, then no copy/paste use is possible in these apps. In this case, the copy/paste DLP setting in the AppConnect global policy has no impact on these apps.

If the Exchange setting allows copy/paste commands, the copy/paste DLP setting in the AppConnect global policy determines the extent of copy/paste use in these apps, just as it does with other apps.

The following table summarizes the copy/paste behavior Email+, depending on the Exchange setting and the AppConnect global policy setting:

TABLE 10. EXCHANGE POLICY AND APPCONNECT GLOBAL POLICY COPY/PASTE SETTING INTERACTION

	Copy/Paste DLP setting on AppConnect global policy		
	No restrictions	**Among AppConnect apps**	**Within an AppConnect app**
Exchange setting disables copy/paste	Not allowed for Email+.	Not allowed for Email+.	Not allowed for sEmail+.
Exchange setting allows copy/paste	AllowedEmail+.	Allowed among AppConnect apps and Email+ Allowed among unsecured apps and Email+	Allowed within Email+ Allowed among unsecured appsand Email+

Web-related DLP policies

The following describes the web-related DLP policies:

- Web DLP policy for browser launching
- DLP allowing links from non-AppConnect apps to open in Web@Work
- Web DLP versus Non-AppConnect apps can open URLs in Web@Work DLP

Web DLP policy for browser launching

You configure the Web DLP policy for browser launching in the AppConnect global policy. This Web DLP policy specifies whether an unsecured browser can attempt to display a web page when a device user taps the page's URL in a secure app.

For example, consider a device user who is viewing an email in a secure email app, and the email body contains a URL. The user taps on the URL to view the web page in a browser. The following table describes the behavior for opening browsers from secure apps:

TABLE 11. WEB DLP POLICY BEHAVIOR WITH AND WITHOUT WEB@WORK

	Web@Work installed	**Web@Work not installed**
Web DLP policy: allowed	The user is prompted to choose between Web@Work and available unsecured browsers to attempt to display the web page.	Unsecured browser attempts to display the web page.
Web DLP policy: not allowed	Web@Work displays the web page.	Web page does not display. An error message is displayed that indicates that a secure browser is required but not installed.

NOTE: If the URL points to a server behind the enterprise's firewall, an unsecured browser's attempt to display the web page fails.

DLP allowing links from non-AppConnect apps to open in Web@Work

AppConnect supports a data loss prevention policy (DLP) that determines whether device users can choose to view a web page in Web@Work when they tap a link (URL) in an app that is not AppConnect-enabled. You specify whether to give device users that choice on the AppConnect Device configuration for Android.

NOTE: This DLP also determines whether device users can choose AppConnect-enabled browsers besides Web@Work.

Allowing links from non-AppConnect apps to open in Web@Work benefits device users who use:

- Apps that are not AppConnect-enabled, especially email apps.
- Web@Work for viewing enterprise web pages.

Without this feature, links to enterprise web pages in email apps that are not AppConnect-enabled do not give Web@Work as a choice for viewing the web page. To view the web page, device users have to copy the link's URL from the email into Web@Work. Now, if you allow it, the user can tap on the link and choose to view the resulting web page in Web@Work, which results in a simpler user experience.

Web DLP versus Non-AppConnect apps can open URLs in Web@Work DLP

The AppConnect global policy has two similar sounding data loss prevention policies for Android devices:

- **Web**
- **Non-AppConnect apps can open URLs in Web@Work**

The following table compares them:

TABLE 12. WEB DLP VERSUS NON-APPCONNECT APPS CAN OPEN URLS IN WEB@WORK DLP

If you allow **Web**...	You can tap on a link in an AppConnect-enabled app...	and open the web page in an unsecured browser.	Therefore, this option is about data leaving the AppConnect container.
If you allow **Non-AppConnect apps can open URLs in Web@Work**....	You can tap on a link in an app that is **not** AppConnect-enabled....	and open the web page in Web@Work.	Therefore, this option is about data coming into the AppConnect container.

You can allow or not allow these two options in any combination.

DLP policy for media player access

You configure the DLP policy for media player access on the MobileIron Cloud. You can choose whether to allow AppConnect apps to stream media to media players on the device.

You configure the setting in the AppConnect Device configuration for Android on MobileIron Cloud.

Consider these scenarios:

- An AppConnect email app has an email with a voice recording attached. The email app can play the recording by using a media player on the device.
- An AppConnect app contains video assets for executive communication and training.

If you allow this capability, AppConnect apps can stream the following file types to media players:

- MP3 audio files
- WAV audio files
- MP4 video files

Media file requirements

AppConnect apps optimize media file downloading, encryption, and decryption, while still keeping the data secure in the AppConnect container. No encrypted copy of the media file is temporarily stored on the device’s SD card.

This optimization supports streaming larger files, where the supported file size depends on:

- the device specifications
- the Android version on the device
- the apps running concurrently on the device

IMPORTANT: MobileIron recommends that secure apps developers perform tests to profile app performance based on the device, Android version, concurrently running apps, and media file size.

Device-initiated security controls for AppConnect for Android

You can protect corporate data on devices even when the devices are off-line. If the device is compromised (rooted) or USB debugging is enabled, MobileIron Go can retire all secure apps on the device. Retiring secure apps means that they become unauthorized (blocked), and their data is deleted (wiped).

The detection of these two security violations occurs on the device. Furthermore, the decision to retire secure apps because of these violations also occurs on the device. Connectivity with MobileIron Cloud is not required for these security controls.

You configure these actions in the **Compromised Devices**, **Data Protection/Encryption Disabled**, or **Custom Policy** policies on MobileIron Cloud.

For information about configuring actions in policies, see "Policies" in the *MobileIron Cloud Administrator Guide*.

Interaction with the Exchange setting

These compliance actions retire all secure apps, which can include email clients. However, the device user can still use lower priority email clients, such as the native Samsung email client, if the device's Exchange setting allows them.

Therefore, if you do not want to allow any email access when the device is compromised or USB debugging is enabled, modify the Exchange setting:

1. In MobileIron Cloud, go to **Configurations**.
2. Edit the Exchange setting that is applied to the devices of interest.
3. In the Android section, modify the **Exchange App Priority** so that only AppConnect-enabled email clients are selected.
4. Click **Save**

Secure File Manager features

The Secure File Manager allows a user to save, browse, and manage files in the secure container. For example, the user can browse saved email attachments or SharePoint documents. The user can also save documents from any other AppConnect app.

The secure File Manager app also supports the following:

- Unzipping files from a secure app
  When the device user taps a ZIP file in a secure app, such as when a ZIP file is an email attachment, the File Manager app opens. The files in the ZIP file are stored in the folder sdcard/UnzippedFiles. If the device

user subsequently unzips a ZIP file containing files with the same name as previously stored files, the files are overwritten.

- File download using the Android DownloadManager API

  Some secure apps use the Android DownloadManager API to download files securely to the device. For such downloads to be successful, the FileManager that MobileIron provides must also be installed on the device. The FileManager ensures downloaded files remain in the secure container. Only secure apps in the container can access the files.
- Opening HTML files
- Opening image files (file types supported by Android)

Secure folder access

AppConnect apps have read-only access to the device's system folder. The system folder contains, for example, ringtone files and font files. System folder access means that:

- An AppConnect app can allow a device user to select one of the system folder's ringtones.
- An AppConnect app can access the system folder's font files.
- The secure File Manager can display the system folder.

About allowing a secure app to ignore the auto-lock time

You can specify that a particular secure app is allowed to ignore the auto-lock time.

The auto-lock time specifies the length of a period of inactivity. After this period of inactivity, the device user is prompted to reenter his secure apps passcode to continue accessing secure apps.You configure the auto-lock time:

- MobileIron Cloud deployments: on the AppConnect Device configuration.

For some apps, staying on a screen is critical. For example, in a navigation app, the device user taps the screen only infrequently, but the screen must continue displaying. Therefore, the app is designed to ignore the Android screen timeout setting, which turns off the screen after a period of time.

Such apps also require that when the auto-lock time expires, the app's screen continues displaying. The normal behavior of having the Secure Apps Manager prompt for the secure apps passcode is not compatible with the app's functionality.

By allowing an app to ignore the auto-lock time for these critical screens, you improve the app's user experience. The app's critical screens are not interrupted by prompting the user to reenter his secure apps passcode.

You specify that a secure app is allowed to ignore the configured auto-lock time by adding the following key-value pair in the app's AppConnect app configuration:

- Key: AC_IGNORE_AUTO_LOCK_ALLOWED
- Value: true.

App requirements to ignore the auto-lock time

Only apps that use particular Android APIs to keep a screen active can ignore the auto-lock time. The app developer or app vendor will inform you if this feature is possible and important for the app.

NOTE: Most apps do not need to, and should not, ignore the auto-lock time. Even if an app developer requests that you allow the app to ignore the auto-lock time, the choice to do so is yours. Your choice depends on whether your requirements for forcing the user to reenter the secure apps passcode outweigh your requirements for the app to have an uninterrupted screen.

What the device user sees when an app ignores the auto-lock time

Critical screens of the app are not interrupted by prompting the user to reenter his secure apps passcode.

Although the critical screen is not interrupted, note that the secure apps container is still locked when the auto-lock time expires.

For example, consider these scenarios:

- The device user leaves the app by selecting the Home button.
  If the auto-lock time had expired while the app was displayed, the device user is prompted for the secure apps passcode when he relaunches the app or any other secure app.
- The device user changes from an app screen that requires continuous display to another app screen that does not require it.
  If the auto-lock time had expired while the first screen was displayed, the device user is prompted for the secure apps passcode when he changes screens.

Situations that wipe Android AppConnect app data

When an AppConnect app is retired, it becomes unauthorized (blocked), and its data is deleted (wiped). The following situations retire an AppConnect app:

- The device user uninstalls the MobileIron Go or the Secure Apps Manager on the device
- You retire the device.
- You quarantine the device due to a compliance action.

Accessible Android apps to preserve the user experience

AppConnect apps can share data only with other AppConnect apps.

However, some exceptions exist to this rule to:

- Preserve the device user experience.
- Enable the use of system services, such as making voice calls.

The exceptions are:

- Maps
  Tapping a meeting location in an AppConnect email app launches a maps app.
- Phone calls
  Tapping a phone number in any AppConnect app will make a phone call.
- SMS
  An AppConnect app can allow the device user to send an SMS to a corporate contact.
- Browsers
  Tapping a link in an AppConnect app launches a browser. However, you can limit the behavior to opening the link in Web@Work by using a data loss prevention policy.

Secure Apps Manager Android permissions

When the device user installs a version of the Secure Apps Manager prior to version 8.0, the device user is presented a list of permissions that the Secure Apps Manager requires. The device user then chooses whether or not to continue the installation.

Secure Apps Manager 8.0 through the most recently released version as supported by MobileIron behaves differently. Specifically, on devices running Android 6.0 through the most recently released version as supported by MobileIron, the device user is **not** presented a list of permissions when installing the Secure Apps Manager. Instead, the device user is asked to grant certain permissions when the Secure Apps Manager runs. The permissions are for accessing:

- SD card storage
- the camera
- the phone
- contacts

Note The Following:

- On Android versions prior to Android 6.0, regardless of the Secure Apps Manager version, the device user is presented the list of permissions during installation.The device user then chooses whether or not to continue the installation.

- On Samsung devices, regardless of the Secure Apps Manager version, the device user is not presented with a list of permissions at any time (installation time or run-time). The Secure Apps Manager can access the capabilities without asking the device user for permission.

The following table provides more information about each permission request:

TABLE 13. SECURE APPS MANAGER PERMISSION REQUESTS

Permission	Reason needed	Requested at these times	Behavior if not granted
Storage	To securely store AppConnect-related data on the SD card.	When the device user first launches the Secure Apps Manager	The device user cannot login to secure apps.
Camera	A secure app wants to access the camera.	When both of the following are true: • A secure app requests access to the camera • The device user has not yet granted access to the camera for secure apps.	The secure app cannot access the camera.
Phone	A secure app wants to access the phone.	When both of the following are true: • A secure app requests access to the phone. • The device user has not yet granted access to the phone for secure apps.	The secure app cannot access the phone.
Contacts	A secure app wants to access the device’s contacts.	When both of the following are true: • A secure app requests access to contacts. • The device user has not yet granted access to contacts for secure apps.	The secure app cannot access contacts.

Disable analytics data collection for AppConnect for Android

MobileIron collects data to analyze the use of AppConnect for Android apps to help provide customer support, perform bug fixes, improve product functionality and reliability, and fulfill obligations to our customers. For AppConnect apps, MobileIron collects the AppConnect key-value data for the pairs set in MobileIron Cloud.

For a complete list of these key-value pairs, see AppConnect for Android key-value pairs.

You can disable this analytics data collection by adding the following key-value pair:

- Key: MI_AC_DISABLE_ANALYTICS
- Value: true

You add the key-value pair in the **AppConnect Custom Configuration** for the app. See Adding an AppConnect Custom Configuration.

Analytics are collected only if the device is using Secure Apps Manager 8.3 through the most recently released version as supported by MobileIron.

AppConnect for iOS

The following describe features for AppConnect for iOS:

- Open-In data loss prevention policy details
- Custom keyboard control
- Screen blurring
- Dictation with the native keyboard is not allowed for wrapped apps
- Heightened security for AppConnect apps using the Secure Enclave
- Situations that wipe AppConnect for iOS app data
- Device-initiated (local) compliance for iOS jailbreak detection
- Touch ID or Face ID for accessing secure apps
- Certificate authentication from AppConnect apps to enterprise services

Open-In data loss prevention policy details

The following provides details about the Open-In feature:

- Open In behavior in wrapped apps versus SDK apps
- iOS native email use and the Open In DLP policy
- AirDrop use and the Open In DLP policy
- App extension use and the Open In DLP policy
- Whitelisting services integrated into iOS in the Open In DLP policy
- Overriding the Open In policy for an app

Open In behavior in wrapped apps versus SDK apps

You select an app's Open In data loss prevention policy:

- MobileIron Cloud: on the AppConnect Device configuration.

When Open In is allowed, an AppConnect app's Open In behavior is the same as the behavior of a regular, non-AppConnect app. However, if Open In is not allowed to some or all apps, the AppConnect app's behavior depends on the following:

- whether the AppConnect app requesting Open In is a wrapped app or an SDK app
- the iOS version on which a wrapped app is running

- the AppConnect for iOS SDK version of an SDK app

By default, Open In is allowed to all apps.

TABLE 14. OPEN IN BEHAVIOR IN WRAPPED APPS VERSUS SDK APPS

	Open In Not allowed	Open In is allowed to all apps or only to apps in the whitelist
Wrapped apps		
Wrapped apps running on iOS versions prior to iOS 11	**Open In policy enforced by:** AppConnect wrapper in the app. **What if the app initiates Open In?** No target apps are displayed.	**Open In policy enforced by:** AppConnect wrapper in the app. **What if the app initiates Open In?** All apps or apps in the whitelist are displayed as possible target apps.
Wrapped apps running iOS 11 through the most recently released version as supported by MobileIron	**Open In policy enforced by:** AppConnect wrapper in the app. **What if the app initiates Open In?** iOS displays all apps that support the document type as possible target apps. Because of the AppConnect library's enforcement, if the user taps on any of the apps, nothing happens.	**Open In policy enforced by:** AppConnect wrapper in the app. **What if the app initiates Open In?** iOS displays all apps that support the document type as possible target apps. Because of the AppConnect library's enforcement, if the user taps on an app that is not allowed, nothing happens

TABLE 14. OPEN IN BEHAVIOR IN WRAPPED APPS VERSUS SDK APPS (CONT.)

	Open In Not allowed	**Open In is allowed to all apps or only to apps in the whitelist**
SDK apps		
Apps built with AppConnect 3.5 for iOS through the most recently released version as supported by MobileIron	**Open In policy enforced by:** The AppConnect library, contained in the AppConnect app. **App is responsible for:** Disabling user interfaces, such a menu items, that provide the Open In capability. **What if the app initiates Open In anyway?** iOS displays all apps that support the document type as possible target apps. Because of the AppConnect library's enforcement, if the user taps on any of the apps, that target app cannot open the file. In this case: • Target app error handling varies. For example, some target apps display an error pop-up. • Error handling also varies for the SDK app that initiated the Open In. Some apps display an error message.	**Open In policy enforced by:** The AppConnect library, contained in the AppConnect app. **App is responsible for:** Enabling user interfaces, such as menu items, that provide the Open In capability. **What happens when the app initiates Open In?** iOS displays all apps that support the document type, including the apps that are not allowed by the Open In policy. Because of the AppConnect library's enforcement, if the user taps on an app that is not allowed, that target app cannot open the file. In this case: • Target app error handling varies. For example, some target apps display an error pop-up. • Error handling also varies for the SDK app that initiated the Open In. Some apps display an error message. **Special case for iOS native email app:** For apps using AppConnect 4.0 for iOS through the most recently released version as supported by MobileIron, if the user taps to launch the native email app, but it is not in the whitelist, Email+ for iOS is launched if it is installed on the device.

iOS native email use and the Open In DLP policy

Apps have various ways to launch the iOS native email app, including:

- using the iOS Open In menu in which the native email app is an option
- specifically launching the iOS native email app
- displaying a standard native email interface inside the app
- requesting to launch any app that handles email

The first way, using the iOS Open In menu, is part of the handling described in Open In behavior in wrapped apps versus SDK apps.

The other ways of invoking the iOS native email app are also impacted by the Open In Data Loss Prevention (DLP) policy. The impact depends on whether the AppConnect app uses:

- Open In and native email with an AppConnect version prior to AppConnect 4.0 for iOS
- Open In and native email with AppConnect 4.0 for iOS through most recently released version

If you want to include iOS native email in the Open In whitelist, seePutting iOS native email into the Open In Whitelist .

Open In and native email with an AppConnect version prior to AppConnect 4.0 for iOS

For apps using AppConnect versions prior to AppConnect 4.0 for iOS, the Open In DLP policy does not impact launching the iOS native email app from an AppConnect app. That is, launching the iOS native email app is always allowed. However, one exception exists to this rule. Launching the native email app is **not** allowed when:

- the Open In policy specifies a whitelist, and
- the iOS native email app is not in the whitelist

Therefore, even when you set the Open In policy to, for example, not allowed, launching the iOS native email app is allowed when the device user taps the app to:

- specifically launch the iOS native email app
- display a standard native email interface inside the app
- launch any app that handles email

Open In and native email with AppConnect 4.0 for iOS through most recently released version

For apps using AppConnect 4.0 for iOS through the most recently released version as supported by MobileIron, the Open In DLP policy impacts launching the iOS native email app from an AppConnect app. If Open In is allowed for all apps, then iOS native email can be launched. However, the behavior for the other Open In policy settings is described in the following table:

TABLE 15. OPEN IN POLICY AND IOS NATIVE EMAIL

Device user action	Open In Not Allowed	Open In is allowed only to whitelisted apps, and iOS native email is NOT in the whitelist	Open In is allowed only to whitelisted apps, and iOS native email is in the whitelist
Taps to specifically launch the iOS native email app	iOS native email is **not** launched.	iOS native email is **not** launched.	iOS native email **is** launched.
Taps to display a standard native email interface inside the app	iOS native email is **not** launched.	iOS native email is **not** launched.	iOS native email **is** launched.
Taps to launch any app that handles email	iOS native email is **not** launched.	iOS native email is **not** launched. Email+ for iOS is launched if it is installed on the device.	iOS native email **is** launched.

For apps built or wrapped with AppConnect 4.1 through the most recently released version supported by MobileIron, you can override the behavior for the scenario when the user taps to launch any app that handles email. Specifically, if the Open In policy blocks launching the iOS native email app in this scenario, you can allow iOS native email to launch. To allow iOS native email to launch, add the key MI_AC_DISABLE_SCHEME_BLOCKING with the value `true` to the app's AppConnect app configuration.

AppConnect apps can also override the Open In policy for this scenario, allowing the iOS native email app to launch. Contact the application vendor or developer to find out if the app overrides the policy.

Putting iOS native email into the Open In Whitelist

To put the native iOS mail app is in the whitelist, put both of these bundle IDs in the whitelist:

- com.apple.UIKit.activity.Mail
- com.apple.mobilemail

IMPORTANT: However, include iOS native email in a whitelist for an AppConnect app only if you understand the potential impact of leaking secure data.

AirDrop use and the Open In DLP policy

The Open In DLP policy impacts the use of AirDrop as follows:

- A wrapped AppConnect app's use of AirDrop behaves according to the Open In DLP policy.
- An AppConnect app built with the AppConnect 3.1 for iOS SDK through the most recently supported version behaves according to the Open In policy.

App extension use and the Open In DLP policy

For apps using AppConnect 4.0 for iOS through the most recently released version as supported by MobileIron, the Open in data loss protection policy includes restricting access to the iOS extensions that apps provide. Specifically:

Open In DLP for host app (the app using the extension)	Extension behavior
All apps allowed	The host app can use any app's extension for Open In.
Whitelist	The host app can use only extensions of apps in the whitelist for Open In.

Whitelisting services integrated into iOS in the Open In DLP policy

When you whitelist apps for the Open In DLP setting, you provide the bundle ID of each whitelisted app in the AppConnect Device configuration . Although the bundle ID of apps in the Apple App Store or your own in-house apps are readily available, the bundle IDs for services integrated into iOS are not well known.

The following list gives the bundle IDs of some common services integrated into iOS. ***However, include them in a whitelist for an AppConnect app only if you understand the potential impact of leaking secure data***.

- com.apple.UIKit.activity.PostToFacebook
- com.apple.UIKit.activity.PostToTwitter
- com.apple.UIKit.activity.PostToWeibo
- com.apple.UIKit.activity.AssignToContact
- com.apple.UIKit.activity.AddToReadingList
- com.apple.UIKit.activity.Quicklook
- com.apple.UIKit.activity.Message

Overriding the Open In policy for an app

You specify the Open In behavior for an AppConnect app in the AppConnect Device configuration.

To override the Open In setting specified in the AppConnect Device configuration for a specific app, create or modify an AppConnect app configuration for the app, and add this key-value pair to its **App-specific Configurations** section:
Key: MI_AC_DISABLE_OPEN_IN_ENFORCEMENT
Value: YES
This key-value pair disables Open In enforcement in the AppConnect library of an AppConnect app, which means that Open In is allowed from the app to all apps.

NOTE: Disabling Open In behavior, has the potential impact of leaking secure data.

Custom keyboard control

Custom keyboard extensions sometimes send data to servers when a device user enters data into an app. They send this data for assistance with word-prediction, for example. This behavior has potential for harmful data loss.

To allow users to use custom keyboards, add the following key-value pair

- Key: MI_AC_IOS_ALLOW_CUSTOM_KEYBOARDS
- Value: true

You add the key-value pair in the **AppConnect Custom Configuration** for the app.

When the key's value is true, the AppConnect app is allowed to use custom keyboards. If the value is false, the app is not allowed to use custom keyboards.

If you do not include the key-value pair for the app, the AppConnect app is allowed to use custom keyboards. However, when the key is not present, an AppConnect app can override this behavior and not allow some or all custom keyboards. Check the app's documentation for its behavior regarding custom keyboards.

The following table summarizes whether an AppConnect app is allowed to use custom keyboards. The behavior depends on:

- whether the app is a wrapped app or SDK app
- whether the release of AppConnect that the app uses supports the key MI_AC_IOS_ALLOW_CUSTOM_KEYBOARDS.
- whether you provide the key-value pair MI_AC_IOS_ALLOW_CUSTOM_KEYBOARDS

TABLE 16. WHETHER AN APPCONNECT APP CAN USE CUSTOM KEYBOARDS

	Key not provided	**Key provided and value is true**	**Key provided and value is false**
Wrapped apps			
Wrapped with AppConnect 4.0	Custom keyboard use is allowed, but the app can override the behavior and not allow some or all custom keyboards.	Custom keyboard use is allowed	Custom keyboard use is not allowed
Wrapped with versions of AppConnect prior to AppConnect 4.0	Custom keyboard use is not allowed	Key is not applicable. Custom keyboard use is	Key is not applicable. Custom keyboard use is

TABLE 16. WHETHER AN APPCONNECT APP CAN USE CUSTOM KEYBOARDS (CONT.)

	Key not provided	**Key provided and value is true**	**Key provided and value is false**
		not allowed	not allowed
SDK apps			
Built with AppConnect 4.0	Custom keyboard use is allowed, but the app can override the behavior and not allow some or all custom keyboards.	Custom keyboard use is allowed.	Custom keyboard use isnot allowed
Built with versions of AppConnect prior to AppConnect 4.0	The app determines whether custom keyboard use is allowed.	Key is not applicable. The app determines whether custom keyboard use is allowed.	Key is not applicable. The app determines whether custom keyboard use is allowed.

Screen blurring

For added security, an AppConnect app's screen should be blurred whenever the app becomes inactive. This behavior hides sensitive data.

Wrapped apps always blur the screen when the app becomes inactive. However, for SDK apps, the behavior changes with AppConnect 4.0 for iOS. Prior to AppConnect 4.0 for iOS, the app itself chose whether to blur the screen and implemented the behavior. For AppConnect 4.0 for iOS through the most recently released version as supported by MobileIron, the AppConnect library controls blurring screens. However, the app has to give the AppConnect library this control. Check your AppConnect app's documentation to see whether it gives screen blurring control to the AppConnect library.

If the app has given screen blurring control to the AppConnect library, you can disable screen blurring by setting a key-value pair in your app's AppConnect app configuration. The key is MI_AC_ENABLE_SCREEN_BLURRING with the value false.

Dictation with the native keyboard is not allowed for wrapped apps

Apps wrapped with AppConnect 4.0 for iOS block the use of dictation when using the native iOS keyboard. You can override this behavior by setting a key-value pair on the app's configuration. The key is called MI_AC_WR_

ALLOW_KEYBOARD_DICTATION. If you do not include the key, dictation is not allowed. If you set the value to true, then wrapped AppConnect apps can use dictation with the native keyboard.

NOTE: Apps built with the AppConnect for iOS SDK, and apps wrapped with wrapper versions prior to AppConnect 4.0 for iOS allow the use of dictation when using the native iOS keyboard. The key MI_AC_WR_ALLOW_KEYBOARD_DICTATION has no impact on those apps.

Heightened security for AppConnect apps using the Secure Enclave

For heightened security of especially sensitive data, such as encryption keys and passwords, you can configure AppConnect apps to use the Apple hardware known as the Secure Enclave. By using the Secure Enclave, the app reduces the sensitive data's attack surface, because the sensitive data is stored in the Secure Enclave rather than in plain-text in memory. When sensitive data is stored in memory, it can be captured in a memory dump.

For an AppConnect app to use the Secure Enclave, the device must:

- have Apple's Secure Enclave hardware.

  NOTE: Devices that have biometric security have Secure Enclave hardware
- be running iOS 11 through the most recently released version as supported by MobileIron

To configure an AppConnect app to use the Apple Secure Enclave, you use the key named MI_AC_CONTAINER_ TYPE in the app's AppConnect app configuration.

The possible values for MI_AC_CONTAINER_TYPE are:

Value	Description
ENCLAVE	The Secure Enclave is used to store: • Sensitive data as defined by the app. Check the app's documentation to see if the app uses the Secure Enclave. • encryption keys used by the AppConnect library
LOCAL	No data is stored in the Secure Enclave. This value is this default if you do not include the key.

Related topics

Adding an AppConnect Custom Configuration

Situations that wipe AppConnect for iOS app data

When an iOS AppConnect app is retired, it becomes unauthorized (blocked), and its data is deleted (wiped). The following situations retire an AppConnect app:

- The device has not completed an AppConnect check-in in the number of days specified in **Wipe AppConnect Data After** in the **AppConnect Device** configuration for iOS in Cloud.
- You retire the device.
- You quarantine the device due to a compliance action.
- MobileIron Go is not present on the device, or present but not registered to MobileIron Cloud.
- The app has retired itself. This action can occur in some apps that behave as either AppConnect apps or regular, unsecured apps.

Device-initiated (local) compliance for iOS jailbreak detection

MobileIron Cloud checks a device for compliance with its security policy each time the device checks in. MobileIron Cloud also checks all devices for compliance at regular intervals to detect out-of-compliance devices that have not checked in. When a device is out of compliance, MobileIron Cloud initiates the specified compliance actions.

When an iOS device is jailbroken (compromised), some compliance actions can be device-initiated. Device-initiated compliance (local compliance) for jailbreak detection means:

- The MobileIron client detects the violation.
- The MobileIron client performs one or more of the following: alerts the user, blocks AppConnect apps, or retires AppConnect apps (blocks access to the apps and wipes their data).

These actions do not depend on connectivity to MobileIron Cloud.

Compliance actions for jailbroken devices is configured in the Compromised Devices policy on MobileIron Cloud. For more information, see "Policies" in the *MobileIron Cloud Administrator Guide*.

Touch ID or Face ID for accessing secure apps

The option to enable Touch ID or Face ID is available in the **AppConnect Device** configuration or the **Default iOS AppConnect** configuration.

See Quick start configuration AppConnect for iOS for information on editing an AppConnect device configuration and the enabling the Touch ID option. Enabling the option automatically also enables Face ID. Secure apps passcode must be enabled to enable the Touch ID option.

Enabling the Touch ID option gives the device user the convenience of using Touch ID or Face ID rather than an AppConnect passcode to access secure apps. If entering the Touch ID or Face ID fails, the user enters (falls back to) the device passcode to access secure apps.

Device user experience with Touch ID or Face ID

The following is the device user experience for a newly registered user:

1. After device users register with MobileIron Go, they are prompted to create an AppConnect passcode.
2. After creating the AppConnect passcode, MobileIron Go gives users the option to use Touch ID/Face ID to access secure apps.
3. If users choose the Touch ID/Face ID option, they can use Touch ID or Face ID when accessing secure apps after the auto-lock time has expired.
4. Device users can later change their choice about using Touch ID/Face ID in MobileIron Go in **Settings > Secure Apps > Authentication**.

Security versus convenience of passcode and Touch ID/Face ID options

AppConnect security involves:

- access to AppConnect apps.
- encrypting AppConnect-related data such as app configurations and certificates.
- encrypting data that the app saves on the device.

The following table lists possible passcode and Touch ID/Face ID choices **from most secure to least secure**, and discusses the level of device user convenience.

NOTE: In all cases, stronger passcodes are more secure than weaker passcodes (such as a 4-digit number).

TABLE 17. SECURITY VS DEVICE USER CONVENIENCE OF PASSCODE AND TOUCH ID/FACE ID OPTIONS

Passcode and Touch ID/Face ID configuration on MobileIron Cloud	**Security of AppConnect apps**	**Convenience for device user**
Require both: • a device passcode • an AppConnect passcode	Highest	Least convenient for accessing both the device and AppConnect apps.
Require only a device passcode	Very high NOTE: Once the device is unlocked, unauthorized users can access AppConnect apps.	Convenient for accessing AppConnect apps, but inconvenient for accessing the device. However, the device user can make accessing the device more convenient by setting up Touch ID or Face ID for unlocking the device.
Require only an AppConnect passcode	High NOTE: Data that the app saves to	Convenient for accessing the device but inconvenient for accessing AppConnect apps.

TABLE 17. SECURITY VS DEVICE USER CONVENIENCE OF PASSCODE AND TOUCH ID/FACE ID OPTIONS (CONT.)

Passcode and Touch ID/Face ID configuration on MobileIron Cloud	Security of AppConnect apps	Convenience for device user
	the device is not encrypted unless the app uses the secure file I/O provided in the AppConnect for iOS SDK.	
Require both: • a device passcode • Touch ID or Face ID for AppConnect apps	High NOTE: Other device users who have added fingerprints or Face IDs, such as family members, can also access AppConnect apps.	Very convenient for accessing both the device and AppConnect apps.
No passcodes required	Lowest Note The Following: • Unauthorized users can access the device and AppConnect apps. • AppConnect-related data, such as app configurations and certificates, is encrypted but the encryption key is not protected by a passcode. • Data that the app saves on the device is encrypted but the encryption key is not protected by a passcode.	Most convenient for accessing both the device and AppConnect apps.

Certificate authentication from AppConnect apps to enterprise services

An AppConnect app can send a certificate to identify and authenticate the app user to an enterprise service. The AppConnect library, which is part of every AppConnect app, makes sure the connection uses the certificate. No additional development is required for the app.

When an AppConnect app uses a certificate to authenticate the app user to an enterprise service using HTTPS:

- The authentication occurs without interaction from the app user.

  The app user does not need to enter a user name and password to log into enterprise services, resulting in a better user experience.
- Access to the enterprise service is more secure.

The AppConnect app must use networking methods that this feature supports. Contact the application vendor or developer to find out if the app supports certificate authentication to enterprise services.

Certificate authentication from AppConnect apps to enterprise services:

- is for HTTPS connections only.
- does not work if you are using AppTunnel with HTTP/S tunneling for the connection.
- does work if you are using AppTunnel with TCP tunneling for the connection.

NOTE: Authenticating users to enterprise servers using Kerberos Constrained Delegation (KCD) with AppTunnel has been available for some time. This feature does not use KCD or AppTunnel.

Impact on AppTunnel use

If an AppConnect app uses a certificate to authenticate the app user to an enterprise service:

- The app **can** use AppTunnel with TCP tunneling (provided with the Tunnel app) to access an enterprise service.
- The app **cannot** use AppTunnel with HTTP/S tunneling for accessing the same URL to which the certificate authenticates.

  The app can use AppTunnel for HTTP/S tunneling for accessing a different URL than the one that the certificate authenticates to.

Set up certificate authentication from an AppConnect app

Setting up certificate authentication from an AppConnect app requires the following two main steps:

1. Configure MobileIron Cloud with the certificate that the app will use to authenticate to the enterprise service.

   See Configuring a certificate on MobileIron Cloud.
2. Add two sets of key-value pairs to the AppConnect Custom Configuration.

   See Configuring a certificate on MobileIron Cloud.

Configuring a certificate on MobileIron Cloud

Configure the certificate that the app will use to authenticate to the enterprise service on MobileIron Cloud.

Procedure

1. Add a certificate authority in **Admin > Certificate Management > Certificate Authority**.
   A Connector installation is required if you are using an external certificate authority.
2. Create an Identity Certificate setting, in **Configurations > Add > Identity Certificate**.
   For Certificate Distribution, select **Dynamically Generated** and for Source, select the certificate you configured in **Admin > Certificate Management > Certificate Authority**. You will reference the Identity Certificate configuration in the key-value pair in the app’s Custom AppConnect configuration.

Related topics

- "Certificate Management" in the *MobileIron Cloud Administrator Guide*.
- "Identity Certificate" in the *MobileIron Cloud Administrator Guide*.

Configuring the key-value pairs for the certificate and URL matching rule

To set up certificate authentication from an AppConnect app to an enterprise service, add two sets of key-value pairs to the app's **AppConnect Custom Configuration**. The two key-value pairs in each set specify:

- a certificate
- a URL matching rule

When the app makes a web request to a URL that matches a URL matching rule, the connection uses the certificate.

Procedure

1. In MobileIron Cloud, **App Catalog**, edit the setting for the AppConnect app.
2. For the AppConnect app, go to **App Configurations > AppConnect Custom Configuration**.
3. Edit an existing custom configuration or add a new **AppConnect Custom Configuration**.

4. In the **AppConnect Certificate Configuration** section add the following key value pairs.

Key	Value
MI_AC_CLIENT_CERT_# substituting digits of your choice for # The key is case-sensitive. **Example** MI_AC_CLIENT_CERT_1 MI_AC_CLIENT_CERT_2 MI_AC_CLIENT_CERT_15 NOTE: You can have many MI_AC_CLIENT_CERT_# keys, each with a different digit substitution.	Select the Certificate Enrollment setting from the dropdown list.
MI_AC_CLIENT_CERT_#_RULE Substituting the same digits for # that you used in MI_AC_ CLIENT_CERT_#. The key is case-sensitive. **Example** MI_AC_CLIENT_CERT_15_RULE	Enter the URL that will use the certificate for authentication. **Example** *.mycompany.com/sales myserver.mycompany.com/hr/benefits

5. Click **Save**.

Related topics

- Adding an AppConnect Custom Configuration
- Details about MI_AC_CLIENT_CERT_#_RULE

Details about MI_AC_CLIENT_CERT_#_RULE

Rule format

The value for the key MI_AC_CLIENT_CERT_#_RULE is a URL matching rule. The rule has one of these formats:

- <*host* >
- <*host* >/<*path*>

The <*host* > component specifies a host within a domain. It can include one or more wildcard characters *.

Example

- myserver.mycompany.com
- anotherserver.mycompany.com

- *.mycompany.com

The <*path* > component specifies a path within the host. It cannot include the wildcard character *. It can contain multiple <*subpath*> components.

Example

- sales
- sales/west
- sales/west/california
  The <*subpath*> components are sales, west, and california.

Matching logic

When the app makes a URL request, the AppConnect library within the app compares the URL request with the values of the MI_AC_CLIENT_CERT_#_RULE keys. If the rule matches the URL request, the connection uses the certificate.

A match occurs if all of the following are true:

- the rule's <*host*> matches the URL request's <*host*>.
- Any <*subpath*> in the rule matches the URL request's <*subpath*>, in the same order.
  More <*subpath*> components can be in the URL request than are in the rule.

NOTE: The matching logic ignores any port number or query parameters in the URL request.

Example

TABLE 18. MATCHING RULE EXAMPLE: MY.SERVER.MYCOMPANY.COM

URL Request	Match?	Comments
https://myserver.mycompany.com	Yes	Exact match
https://myserver.mycompany.com/sales	Yes	Matches <*host*>
https://myserver.mycompany.com:8080	Yes	Matches -- port is ignored
https://myserver.mycompany.com:sales?range=quarter	Yes	Matches -- query parameters are ignored
https://anotherserver.mycompany.com	No	<*host*> does not match
http://anotherserver.mycompany.com	No	HTTP, not HTTPS

TABLE 19. MATCHING RULE EXAMPLE: *.MYCOMPANY.COM/SALES

URL Request	Match?	Comments
https://myserver.mycompany.com/sales	Yes	<*host*> and <*subpath*> match.
https://myserver.mycompany.com/sales/west	Yes	<*host*> and <*subpath*> match
https://anotherserver.mycompany.com/sales	Yes	<*host*> and <*subpath*> match
https://myserver.mycompany.com/salesmeeting	No	Entire <*subpath*> must match.
https://myserver.mycompany.com	No	Missing required <*subpath*>
https://myserver.mycompany.com/s	No	Entire <*subpath*> must match.

If more than one MI_AC_CLIENT_CERT_#_RULE value matches the URL request, the rule with the most number of non-wildcard characters is chosen.

For example, consider four MI_AC_CLIENT_CERT_#_RULE keys with the following values:

TABLE 20. EXAMPLE OF SIMILAR MI_AC_CLIENT_CERT_#_RULE VALUES

Key	Value
MI_AC_CLIENT_CERT_1_RULE	*.mycompany.com
MI_AC_CLIENT_CERT_2_RULE	*.mycompany.com/sales/west
MI_AC_CLIENT_CERT_3_RULE	myserver.mycompany.com/sales
MI_AC_CLIENT_CERT_4_RULE	myserver.mycompany.com/sales/west

If the app requests URL https://myserver.mycompany.com/sales/west, the request matches all the values, but only one match is chosen. The chosen match is myserver.mycompany.com/sales/west, and the connection uses the corresponding certificate in MI_AC_CLIENT_CERT_4.

Configuring AppTunnel for AppConnect apps

Adding an AppTunnel configuration

AppTunnel secures the data that moves between your secure AppConnect apps and your corporate data sources. A Standalone Sentry deployment is required to set up AppTunnel. Setting up AppTunnel for an AppConnect app is a two-step process.

1. Configure a **Custom HTTP** or **Custom TCP** AppTunnel service on a Standalone Sentry configured for AppTunnel.
2. Add an AppTunnel configuration for the app.

Before you begin

- Ensure that you have a Standalone Sentry configured to support AppTunnel. The required steps include:
  - Setting up the Standalone Sentry connectivity settings, which include the Sentry host name or IP address, and the port number MobileIron Cloud uses to access the Sentry.
  - Enabling the Standalone Sentry for AppTunnel.
  - Configuring the Standalone Sentry for device authentication, which is how the device authenticates to the Standalone Sentry. This authentication includes setting up certificates if you require them.

  See "Standalone Sentry for AppTunnel," in the *MobileIron Sentry Guide for MobileIron Cloud* for information on how to set up MobileIron Sentry for AppTunnel.
- Configure a **Custom HTTP** or **Custom TCP** AppTunnel service on Standalone Sentry.

  You create the AppTunnel service in MobileIron Cloud, in **Admin > Infrastructure > Sentry**, in a Sentry profile.

  For information about creating a AppTunnel service, see "Configuring Standalone Sentry for AppTunnel" in the *MobileIron Sentry Guide for MobileIron Cloud*.

- Add the AppConnect app to the MobileIron Cloud.
  - For Android, see Quick start configuration AppConnect for Android.
  - For iOS, see Quick start configuration AppConnect for iOS

Procedure

1. In MobileIron Cloud, go to **Apps > App Catalog**.
2. Click the name of the AppConnect app to edit.
3. Click **App Configurations**.
4. For AppTunnel, click **+** to add a new AppTunnel configuration.
5. Use the guidelines in the following table to enter the configuration.

Item	Description
Name	Enter a descriptive name for the configuration.
AppTunnel	
Sentry Profile	Select a Sentry profile configured for AppTunnel from the drop-down list.
Enable Split Tunneling using MobileIron Tunnel	iOS only. Requires MobileIron Go 5.4.0 for iOS and MobileIron Tunnel 4.1.0 for iOS. Before enabling the option, ensure that MobileIron Tunnel is deployed and the Tunnel VPN configuration is applied to the AppConnect app. For information about deploying Tunnel and applying the Tunnel VPN configuration to a managed app, see "Main tasks for configuration MobileIron Tunnel for iOS (Cloud)" in the *MobileIron Tunnel for iOS Guide*. Select the option if the AppConnect app will transition to using WKWebView or the app currently uses WKWebView and any of the following is also true: • AppTunnel rules are configured to tunnel app data. • Enable MobileIron Access is selected. Enabling the option allows the configured AppTunnel rules to be managed through MobileIron Tunnel rather than through AppTunnel. For information about the UIWebView API deprecation, see UIWebView Deprecation and AppConnect Compatibility. NOTE: Rules configured in the Tunnel VPN configuration impact whether app data to the enterprise resource is tunneled. Consider the following case: • You have an AppTunnel rule set up to tunnel app data to an enterprise resource. • Tunnel VPN is configured to disconnect if the enterprise Wi-Fi is available. In the above case, data from the app to the enterprise resource will not be tunneled if the device switches to the enterprise Wi-Fi network.

Item	Description
AppTunnel Rules	
Choose service	Select a service name from the drop-down list. This is the Custom HTTP or Custom TCP service you created in the Sentry profile.
URL Wildcard	Enter one of the following: • an app server's hostname Example: finance.yourcompany.com • a hostname with wildcards. The wildcard character is *. Example: *.yourcompanyname.com If the AppConnect app requests access to this hostname, Sentry tunnels the app data. The Sentry profile and service fields that you specify determine the target app server.
Port	(Optional) Enter a port numbe for the backend enterprise resource to which the traffic is tunneled. The app data is tunneled only if the app's request matches the hostname in the URL Wildcard field and this port number.
+	Click to configure additional service and URL wildcards. Multiple wildcards are evaluated in the order in which they appear. You can reorder the sequence by dragging a row up or down.

6. Select a distribution option.
7. Click **Save**.

Related topics

- "Configuring Standalone Sentry for AppTunnel" in the*MobileIron Sentry Guide for MobileIron Cloud*

Configuring per-app idle session timeout for AppTunnel with TCP tunneling

For an AppConnect app using AppTunnel with TCP tunneling, you can control the idle session timeout for the TCP connection between the app and the enterprise server. This timeout is useful if the enterprise server takes more than 60 seconds to respond to a request from the app. The default idle session timeout is 60 seconds.

To specify a idle session timeout for an AppConnect app, provide a key-value pair in the app's AppConnect app configuration that specifies the idle session timeout.

TABLE 21. IDLE SESSION TIMEOUT KEY-VALUE PAIR

Key	Value
MI_AC_TCP_IDLE_TIMEOUT_MS	An integer greater than 0. The value is the number of milliseconds in which the enterprise server must respond to a request when using AppTunnel with TCP tunneling. The Standalone Sentry handling the AppTunnel times out if this value is exceeded. Default value: 60000

Certificate authentication using AppConnect with TCP tunneling for Android AppConnect apps

Android AppConnect apps support certificate authentication using AppTunnel with TCP tunneling. An AppConnect app can send a certificate to identify and authenticate the app user to an enterprise server. Depending on the server implementation, this authentication occurs without interaction from the device user beyond entering the AppConnect passcode. That is, the device user does not need to enter a user name and password to log into enterprise services. Therefore, this feature provides a higher level of security and an improved user experience.

App and enterprise server requirements

Apps using certificate authentication with AppTunnel with TCP tunneling must initiate a connection that does not use Secure Socket Layer (SSL) to the enterprise server. For example, the app can initiate the connection with a HTTP request, but not with an HTTPS request.

Contact the application vendor or developer to find out whether the app meets these requirements.

IMPORTANT: The connection that this feature makes to the enterprise server is secure; it uses SSL.

The enterprise server must use client certificate authentication with Secure Sockets Layer (SSL).

Configuring certificate authentication using AppTunnel with TCP tunneling for Android AppConnect apps

Configuring certificate authentication with AppTunnel using TCP tunneling for Android AppConnect apps requires the following:

- AppTunnel TCP services on the Standalone Sentry that require certificate authentication.
- The user certificate for the app to present to the enterprise server.

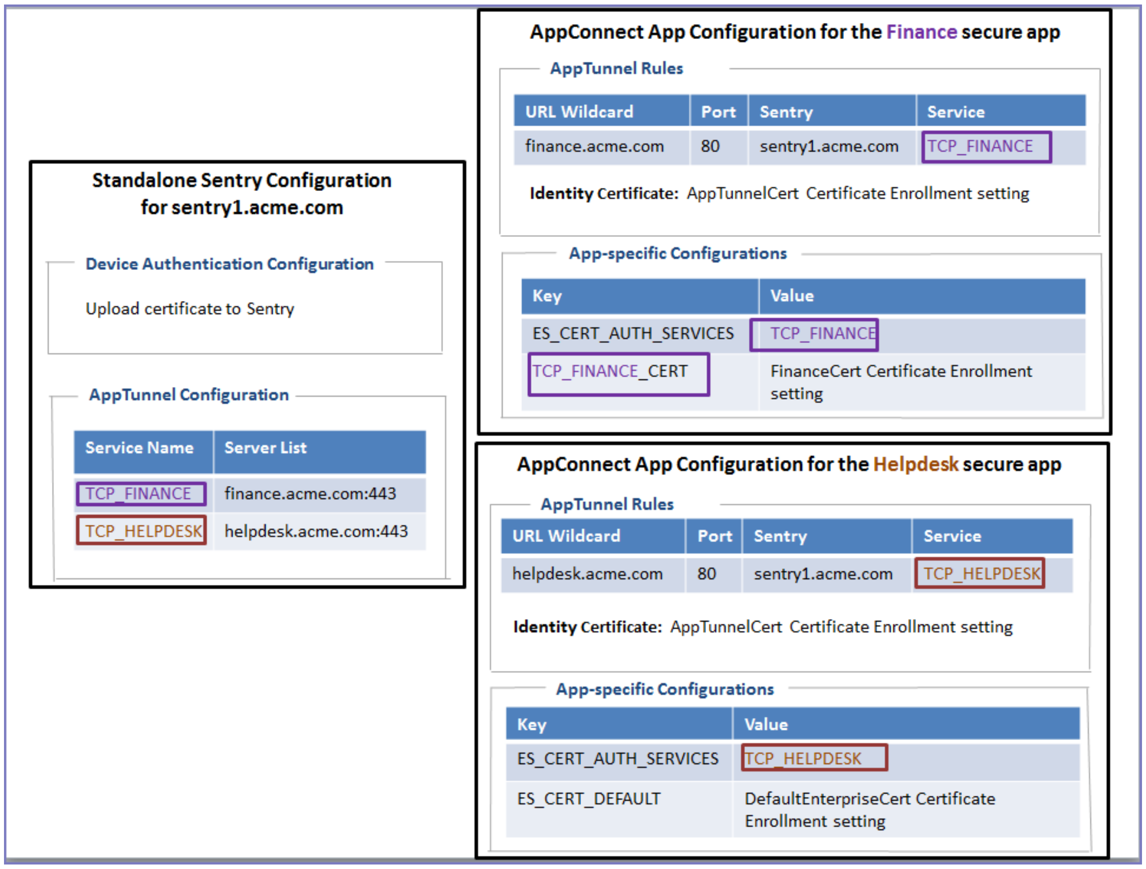
This certificate can be specifically for the enterprise server only, or a default user certificate if you do not require a specific certificate for a service. One other option is to use the same certificate that the app presents to the Standalone Sentry.

The certificate is either an identity certificate or a group certificate.

- Specifying the TCP service and the associated certificate in the app's app configuration on MobileIron Cloud.

The following sample configuration summarizes the configuration in Standalone Sentry and the AppConnect app configuration for a Finance app and Helpdesk app:

FIGURE 10. SAMPLE CONFIGURATION FOR CERTIFICATE AUTHENTICATION WITH TCP TUNNELING



Standalone Sentry Configuration for sentry1.acme.com

Device Authentication Configuration

Upload certificate to Sentry

AppTunnel Configuration

Service Name	Server List
TCP_FINANCE	finance.acme.com:443
TCP_HELPDESK	helpdesk.acme.com:443

AppConnect App Configuration for the Finance secure app

AppTunnel Rules

URL Wildcard	Port	Sentry	Service
finance.acme.com	80	sentry1.acme.com	TCP_FINANCE

Identity Certificate: AppTunnelCert Certificate Enrollment setting

App-specific Configurations

Key	Value
ES_CERT_AUTH_SERVICES	TCP_FINANCE
TCP_FINANCE_CERT	FinanceCert Certificate Enrollment setting

AppConnect App Configuration for the Helpdesk secure app

AppTunnel Rules

URL Wildcard	Port	Sentry	Service
helpdesk.acme.com	80	sentry1.acme.com	TCP_HELPDESK

Identity Certificate: AppTunnelCert Certificate Enrollment setting

App-specific Configurations

Key	Value
ES_CERT_AUTH_SERVICES	TCP_HELPDESK
ES_CERT_DEFAULT	DefaultEnterpriseCert Certificate Enrollment setting

The Finance app and the Helpdesk app:

- Authenticate to Standalone Sentry using the certificate defined by the AppTunnelCert Certificate Enrollment setting.

  This Certificate Enrollment setting is specified as the Identity Certificate in the AppTunnel rules for the AppConnect app configuration for each app.
- Use AppTunnel with TCP tunneling to access the TCP_FINANCE service and TCP_HELPDESK service, respectively.

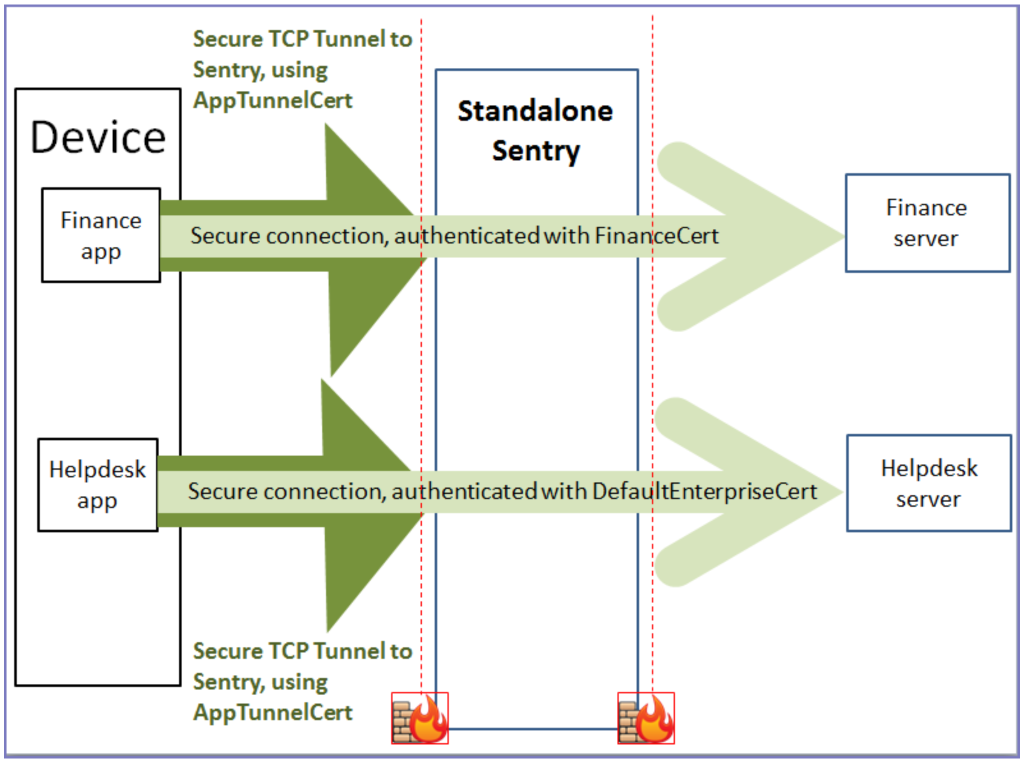
- Use certificate authentication with AppTunnel with TCP tunneling.

  In each app's AppConnect app configuration, the value of ES_CERT_AUTH_SERVICES lists the service that uses certificate authentication.

The two apps use different certificates to authenticate to their respective enterprise servers. The Finance app uses a specific certificate, defined in the FinanceCert Certificate Enrollment setting. The Helpdesk app uses a a default certificate, defined in the DefaultEnterpriseCert Certificate Enrollment setting, to authenticate to its enterprise server. Other apps that access other enterprise services also can use this certificate.

The following diagram illustrates the use of the certificates:

FIGURE 11. CERTIFICATE USAGE IN CERTIFICATE AUTHENTICATION WITH TCP TUNNELING


Secure TCP Tunnel to Sentry, using AppTunnelCert
Device
Finance app
Standalone Sentry
Secure connection, authenticated with FinanceCert
Finance server
Helpdesk app
Secure connection, authenticated with DefaultEnterpriseCert
Helpdesk server
Secure TCP Tunnel to Sentry, using AppTunnelCert

Before you begin

- Configure the TCP AppTunnel service in Standalone Sentry profile.

  For information about creating a AppTunnel service, see "Configuring Standalone Sentry for AppTunnel" and "AppTunnel Service" in the *MobileIron Sentry Guide for MobileIron Cloud*.

- Configure the certificate to authenticate to the enterprise server.

  Add a **Certificate Authority** and create an **Identity Certificate** setting in MobileIron Cloud.

  - Add the **Certificate Authority** in **Admin > Certificate Authority**.
  - Create an **Identity Certificate** setting in **Configuration > Add > Identity Certificate**.

  See the "Identity Certificate" in the *MobileIron Cloud Administrator Guide*.

- Added an AppTunnel configuration app's app configuration.
  See Adding an AppTunnel configuration.

Procedure

1. On MobileIron Cloud, go to the App Catalog.
2. Click the Android AppConnect app to edit its configurations.
3. Go to the App Configurations tab.
4. For AppConnect Custom Configurations, click +.
5. Enter the key-value pairs as described in the following table:

TABLE 22. CERTIFICATE KEY-VALUE PAIRS

Key	**Value**
AppConnect Custom Configuration	
ES_CERT_AUTH_SERVICES	Enter th AppTunnel services that the app uses. Typically, apps use only one services. However, using mutliple services is supported. Separate each service witha semi-colon. The service name must match the service names created in the Sentry profile. **Example** TCP_HELPDESK TCP_HELPDESK;TCP_WIKI;TCP_FINANCE
AppConnect Certificate Configuration	

TABLE 22. CERTIFICATE KEY-VALUE PAIRS (CONT.)

Key	**Value**
<service_name>_CERT where <*service_name*> is one of the AppTunnel services that the app uses, which you listed in Configuring certificate authentication using AppTunnel with TCP tunneling for Android AppConnect apps. Example: TCP_HELPDESK_CERT NOTE: The key is case sensitive. Make sure that the <*service_name*> exactly matches, including case, the AppTunnel service name.	The Certificate Enrollment setting for a certificate used specifically for this AppTunnel service. The configured Certificate Enrollment settings appear in the value field's dropdown list. Note The Following: • If you do not add a **<service_name>_CERT** key for a service that uses certificate authentication, the certificate specified for the key **ES_ DEFAULT_CERT** is used for that service. • If you do not add the **ES_DEFAULT_CERT** key, the certificate that authenticates the user to the Standalone Sentry is used.
ES_DEFAULT_CERT NOTE: The key is case sensitive.	The Certificate Enrollment setting for a default certificate used for services that do not require a specific certificate. The configured Certificate Enrollment appear in the value field's dropdown list.

AppTunnel and TLS protocol versions in Android AppConnect apps

An AppConnect for Android app uses a TLS protocol version to communicate with:

- the Standalone Sentry for network requests using AppTunnel with HTTP/S tunneling or TCP tunneling
- enterprise servers that use certificate authentication using AppTunnel with TCP tunneling

TLSv1.2 is more secure. Therefore, MobileIron recommends that you configure your Standalone Sentry and applicable enterprise servers to accept TLSv1.2.

The following table shows the TLS protocol version the app uses, which depends on:

- the version of the AppConnect wrapper
- whether the app is configured for AppTunnel with HTTP/S tunneling or AppTunnel with TCP tunneling
- whether the app is configured with the applicable key-value pair.

IMPORTANT: In all cases, make sure your Standalone Sentry and applicable enterprise servers accept one of the TLS protocol versions that the AppConnect wrapper requests.

TABLE 23. TLS PROTOCOL VERSIONS USED BY APPCONNECT WRAPPER FOR TCP TUNNELING

Wrapper version	Default TLS protocol	Applicable key-value pair in the app's AppConnect app configuration
8.0 through 8.4 **HTTP/S Tunneling**	TLSv1.2 falling back to TLSv1.0 if required by server	None
8.0 through 8.4 **TCP Tunneling** **(Generation 2 wrapper only)**	TLSv1.0	**MI_AC_USE_TLS1.2** Defaults to false Include this key with the value set to **true** to make the AppConnect wrapper in the app use TLSv1.2 instead of TLSv1.0. Defaults to false Include this key with the value set to **true** to make the AppConnect wrapper in the app use TLSv1.2 instead of TLSv1.0.
8.5 through the most recently released version as supported by MobileIron **HTTP/S Tunneling and TCP Tunneling**	TLSv1.2	**MI_AC_ENABLE_TLS_FALLBACK KVP** Defaults to false Include this key with the value set to **true** if you want the AppConnect wrapper in the app to fallback to TLSv1.0 if the TLSv1.2 request is not accepted by the server.

NOTE: The AppConnect wrapper is the consumer of the key-value pair; the AppConnect app itself ignores it.

Related topics

- To add the key-value pairs, see Adding an AppConnect Custom Configuration
- “Incoming SSL configuration” in the *MobileIron Sentry Guide for MobileIron Cloud*
- AppTunnel with TCP tunneling support for Android AppConnect apps
- Certificate authentication using AppConnect with TCP tunneling for Android AppConnect apps

Impact to tunneling when using a global HTTP proxy

A global HTTP proxy policy ensures that HTTP traffic is redirected to a proxy server that you specify. Configuring a global HTTP proxy policy for devices includes specifying the URL for the proxy auto-configuration (PAC) file. Details are available in “Working with global HTTP proxy policies” in the *MobileIron Core Device Management Guide for iOS and macOS Devices.*

Consider the case in which you have defined an AppTunnel rule in an AppConnect's AppConnect app configuration (or Web@Work setting or Docs@Work setting) that includes the URL to the PAC file. That is, the AppTunnel rule does one of the following:

- Uses a wildcard character in the AppTunnel rule's **URL Wildcard** field such that the PAC file URL matches the rule
- Explicitly names the PAC file URL in the AppTunnel rule's **URL Wildcard** field

The impact of this configuration to tunneling varies depending on the AppConnect app's AppConnect version, as shown in the following table:

TABLE 24. IMPACT TO TUNNELING WHEN USING A GLOBAL HTTP PROXY

AppConnect for iOS SDK or Wrapper version used in the app	Impact to tunneling of defining an AppTunnel rule that includes the URL to the PAC file
3.0 and prior	• The request to the URL for the PAC file is tunneled. • Other URL requests are tunneled according to the AppTunnel rules.
3.1.0, 3.1.1, 3.1.2	Tunneling to the URL for the PAC file is not supported. A tunneling attempt to this URL results in no network access for the app, whether tunneled or not.
3.1.3 through the most recently released version as supported by MobileIron	To support tunneling in these apps, configure a key-value pair in the app's AppConnect app configuration (or Web@Work setting or Docs@Work setting) as follows: • key name: `global_http_proxy_url` • value: the URL of the PAC file, which you also enter in the Proxy PAC URL field of the global HTTP proxy policy. Example: http://pac.myproxy.mycompany.com With this key-value pair: • the URL request to the PAC file is not tunneled. • other URL requests are tunneled as specified by the AppTunnel rules.

AppConnect Key-value Pairs

Some AppConnect features are configured using key-value pairs.

The key-value pairs are summarized in the following tables, along with references to the appropriate documentation with the details.

- AppConnect for Android key-value pairs
- AppConnect for iOS key-value pairs

AppConnect for Android key-value pairs

The following table summarizes the AppConnect for Android key-value pairs used to configure various AppConnect features. The key-value pairs are configured in the AppConnect Custom Configuration for an AppConnect app. For information about add an AppConnect Custom Configuration, see Adding an AppConnect Custom Configuration.

TABLE 25. APPCONNECT FOR ANDROID KEY-VALUE PAIRS

Key name	Description	Reference
`MI_ALLOW_SECURE_COPY_INBOUND`	When **true**, data can be copied from a non-AppConnect app to an AppConnect app even though the Copy/Paste data loss prevention setting is **Among AppConnect Apps** or **Within an AppConnect app**.	Copy/Paste for AppConnect for Android
`AC_IGNORE_AUTO_LOCK_ALLOWED`	When **true**, an AppConnect app ignores the auto-lock time.	About allowing a secure app to ignore the auto-lock time
`MI_AC_TCP_IDLE_TIMEOUT_MS`	Specifies the idle session timeout for the TCP connection between an AppConnect app and an enterprise server when using AppTunnel with TCP tunneling.	Configuring per-app idle session timeout for AppTunnel with TCP tunneling
`MI_AC_DISABLE_ANALYTICS`	When **true**, analytics about AppConnect apps are not collected.	Disable analytics data collection for AppConnect for Android
`MI_AC_USE_TLS1.2`	When **true**, the AppConnect wrapper uses TLSv1.2 for network requests using AppTunnel with TCP tunneling. Applicable only to apps wrapped with AppConnect wrapper versions 8.0 through 8.4	AppTunnel and TLS protocol versions in Android AppConnect apps

TABLE 25. APPCONNECT FOR ANDROID KEY-VALUE PAIRS (CONT.)

Key name	Description	Reference
`MI_AC_ENABLE_TLS_FALLBACK`	When **true**, the AppConnect wrapper falls back to using TLSv1.0 if TLSv1.2 is not accepted for network requests using AppTunnel. Applicable only to apps wrapped with AppConnect wrapper versions 8.5 through the most recently released version as supported by MobileIron.	AppTunnel and TLS protocol versions in Android AppConnect apps
`ES_CERT_AUTH_SERVICES`	Specifies the list of AppTunnel services that use certificate authentication using AppTunnel with TCP tunneling.	Configuring certificate authentication using AppTunnel with TCP tunneling for Android AppConnect apps
`<service_name>_CERT`	Specifies the certificate enrollment setting for the certificate for authenticating to the enterprise server when using AppTunnel with TCP tunneling.	
`ES_DEFAULT_CERT`	Specifies the certificate enrollment setting for the default certificate for authenticating to the enterprise server when using AppTunnel with TCP tunneling.	
`AC_PUBLIC_KEY`	Specifies the certificate setting containing the public certificate that matches the enterprise private key used to sign apps wrapped with the AppConnect wrapping tool.	See "The MobileIron AppConnect for Android Wrapping Tool" in the *MobileIron AppConnect for Android App Developers Guide*

Related topics

AppConnect for iOS key-value pairs

AppConnect for iOS key-value pairs

The following table summarizes the AppConnect for iOS key-value pairs used to configure various AppConnect features.

TABLE 26. APPCONNECT FOR IOS KEY-VALUE PAIRS

Key name	Description	Reference
`MI_AC_DISABLE_OPEN_IN_ENFORCEMENT`	When **Yes**, Open In is allowed to all apps.	Open-In data loss prevention policy details
`MI_AC_DISABLE_SCHEME_BLOCKING`	When **true**, Open In is allowed to the iOS native email app when the user taps the AppConnect app to launch an email app..	Open-In data loss prevention policy details
`MI_AC_LOG_LEVEL`	Specifies the log level for the app: **error**, **info**, **verbose**, or **debug**.	Logging for AppConnect apps for iOS
`MI_AC_LOG_LEVEL_CODE`	Specifies the string that the device user enters to activate the verbose or debug log level.	
`MI_AC_ENABLE_LOGGING_TO_FILE`	When **Yes**, an AppConnect app's logs are logged to files on the device.	
`MI_AC_WR_ENABLE_LOG_CAPTURE`	When **Yes**,when emailing AppConnect-related log files from MobileIron Go, the logs of a wrapped app are emailed along with the logs of the AppConnect wrapper and the AppConnect librar	
`MI_AC_IOS_ALLOW_CUSTOM_KEYBOARDS`	When **true**, the AppConnect app is allowed to use customer keyboards.	Custom keyboard control
`MI_AC_WR_ALLOW_KEYBOARD_DICTATION`	When **true**,a wrapped app can use dictation with the iOS native keyboard.	Dictation with the native keyboard is not allowed for wrapped apps
`MI_AC_ENABLE_SCREEN_BLURRING`	When **false**, screen blurring is disabled if the AppConnect app has given screen blurring control to the AppConnect library.	Screen blurring
`MI_AC_CLIENT_CERT_#`	Used in setting up certificate authentication from an AppConnect app to an enterprise service.	Certificate authentication from AppConnect apps to enterprise services

TABLE 26. APPCONNECT FOR IOS KEY-VALUE PAIRS (CONT.)

Key name	Description	Reference
`MI_AC_CLIENT_CERT_#_RULE`	Used in setting up certificate authentication from an AppConnect app to an enterprise service.	Certificate authentication from AppConnect apps to enterprise services
`MI_AC_CONTAINER_TYPE`	When set to **ENCLAVE**, sensitive data, such as encryption keys, is stored in the Apple Secure Enclave on the device.	Heightened security for AppConnect apps using the Secure Enclave
`MI_AC_USE_ORIGINAL_COOKIES_FOR_DOMAINS`	Some web pages inject custom cookies into web requests. For example, when an end user taps on a link in a web page, the page's JavaScript injects a custom cookie. If a user makes such a request from a web page displayed in an AppConnect app, by default AppConnect does not include the injected cookies in the web request, which can cause the request to fail. AppConnect includes the custom cookies in the request if you include the following key in the app's app-specific configuration: `MI_AC_USE_ORIGINAL_COOKIES_FOR_DOMAINS`. The value of the key is a comma-separated string listing the domains for which the custom cookies should be included. Make sure no spaces are included in the value. For example: `www.somewebsite.com,somename.someotherwebsite.com` Supported with apps built or wrapped with AppConnect 4.2.1 for iOS through the most recently released version as supported by MobileIron.	

Related topics

AppConnect for Android key-value pairs

Troubleshooting AppConnect

Troubleshooting AppConnect setup

You can verify configurations for AppConnect app by checking the configuration applied to the device.

Procedure

1. Go to **Devices > Devices**.
2. Select a device that should be AppConnect ready.
3. Check the **Configurations** tab for the expected device configuration.
4. Check the **AppConnect Apps** tab to ensure that expected apps have been installed as AppConnect apps.
5. Check the app's custom configurations.

Logging for AppConnect apps for iOS

Overview of logging for AppConnect apps for iOS

You can collect detailed log data for AppConnect for iOS apps. You specify the AppConnect apps that should log detailed data. The AppConnect library contained in each specified app also logs detailed data. The log data provides information to help MobileIron Technical Support troubleshoot issues with the apps.

Depending on your configuration, the data is logged to:

- the device’s console.
- the device’s console and files on the device.

Log levels

You choose one of four log levels for an AppConnect app. The two highest levels can log sensitive data. To prohibit unauthorized users from accessing sensitive data, the two highest levels require the device user to enter a debug code that you specify.

Exactly what sensitive data is logged depends on the app, but can include, for example:

- Device user data, including document names and contents, contact lists, notes, and bookmarks
- Encryption keys, passwords, certificates, signing identities, and cookies
- Complete URLs and URL POST data
- Data that reveals the contents of encrypted data

The following table describes the log levels from lowest (least verbose) to highest (most verbose):

TABLE 27. APPCONNECT LOG LEVELS

Log level	Description	Contains sensitive data?	Requires the user to enter the debug code?
Error	Provides error, warning, and status messages. This level is the default. It is always turned on. Error messages are for events that block access to part or all of the app. Example: Corrupt or missing data Warning messages are for events that are suspicious, but not quite failures like errors. Example: Unexpected data that is ignored Status messages indicate major changes in the state of the app. Example: User successfully logged in	No	No
Info	Provides error, warning, and status messages, plus more information. Info messages indicate minor changes in the state of the app. Example: AppConnect app check-in times	No	No
Verbose	Provides error, warning, status, and info messages, plus more, possibly sensitive, information.	Yes	Yes

TABLE 27. APPCONNECT LOG LEVELS (CONT.)

Log level	Description	Contains sensitive data?	Requires the user to enter the debug code?
	Verbose messages provide more extensive information, possibly including sensitive details. Example: Server URLs		
Debug	Provides error, warning, status, info, and verbose messages, plus further information, which is possibly sensitive. Debug messages have the most information, possibly including sensitive details. Example: URL request details	Yes	Yes

How the log level appears in messages

When you set the log level for an app, messages logged by the following components are impacted:

- the AppConnect app
- the MobileIron AppConnect library contained in the AppConnect apps
- the AppConnect wrapper (only applicable for wrapped AppConnect apps)

The messages logged by these components include the log level as shown in the following table:

TABLE 28. HOW THE LOG LEVEL APPEARS IN MESSAGES

Component	App name in log message	How the log level appears in messages

An AppConnect app	The app's name	`[Error]`
		`[Warning]`
		`[Status]`
		`[Info]`
		`[Verbose]`
		`[Debug]`
		NOTE: The value `error` for the **MI_AC_LOG_LEVEL** key in an app's AppConnect app configuration, can result in messages with `[Error]`, `[Warning]`, and `[Status]`.

AppConnect library contained in an AppConnect app	The app's name	`[AppConnect:Error]`
		`[AppConnect:Warning]`
		`[AppConnect:Status]`
		`[AppConnect:Info]`
		`[AppConnect:Verbose]`
		`[AppConnect:Debug]`
		NOTE: The value `error` for the **MI_AC_LOG_LEVEL** key in an app's AppConnect app configuration, can result in messages with `[AppConnect:Error]`, `[AppConnect:Warning]`, and `[AppConnect:Status]`.

The AppConnect wrapper (only applicable for wrapped AppConnect apps)	The app's name	`[AppConnectWrapper:Error]`
		`[AppConnectWrapper:Warning]`
		`[AppConnectWrapper:Status]`
		`[AppConnectWrapper:Info]`
		`[AppConnectWrapper:Verbose]`
		`[AppConnectWrapper:Debug]`
		NOTE: The value `error` for the **MI_AC_LOG_LEVEL** key in an app's AppConnect app configuration, can result in messages with `[AppConnectWrapper:Error]`, `[AppConnectWrapper:Warning]`, and `[AppConnectWrapper:Status]`.

Log file details

The following details are available in the log files for each app:

- The log files for each app are saved to the following directory:
  Apps/<*app name*>/Library/Application Support/AppConnectLogs
- The log file for each app is named appconnect.log.
- The log file is at most 1 MB.
- When appconnect.log exceeds 1 MB:
  1. It is renamed to appconnect.log.<*timestamp*>.
     Example: appconnect.log.2015-05-28 15:13:21
  2. Logging begins in a new file named appconnect.log.
  3. If 20 log files already exist, the oldest file is deleted.

Enable logging for an AppConnect app

To enable the log level and debug code for an app, and to specify that you want to log to files in addition to the device console, add the following key-value pairs in the AppConnect app configuration for the app. The key is case sensitive.

TABLE 29. APPCONNECT FOR IOS KEY-VALUE PAIRS

Key	**Value**
`MI_AC_LOG_LEVEL`	Specifies the log level for the app. Enter one of the following: • error • info • verbose • debug.
`MI_AC_LOG_LEVEL_CODE`	Add this key-value pair if you entered verbose or debug. Specifies the string that the device user enters to activate the verbose or debug log level. Enter a string for the value. The device user will enter this string to activate the verbose or debug log level. You can make up any string. For example, enter 37!8D. For the most security, use a code that is difficult to guess. The string is case-sensitive.
`MI_AC_ENABLE_LOGGING_TO_FILE`	The AppConnect app's logs are logged to files on the device. Enter Yes.

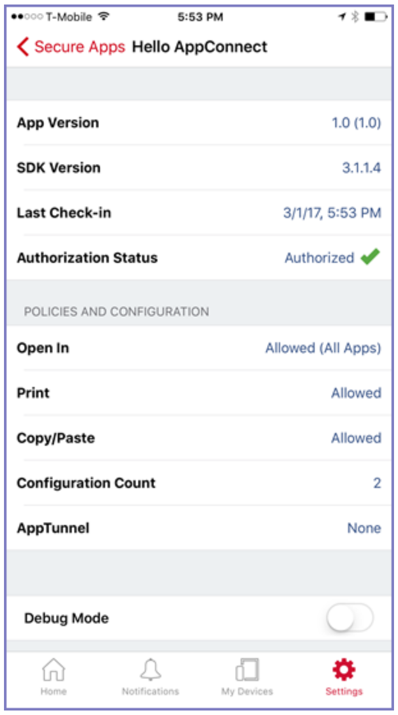
Log level configuration impact on the device

Error level logging is always on, regardless of whether you have configured the MI_AC_LOG_LEVEL key-value pair, and it requires no actions from the device user. Info level logging also does not require device user interaction. However, verbose or debug level logging do not begin until the device user activates debug mode in MobileIron Go.

The status details for an AppConnect app include a Debug Mode switch only when you have configured both of the following in the app's AppConnect app configuration:

- a log level of verbose or debug
- a debug code

In this case, the status details for an AppConnect app shows the Debug Mode switch:



App Version	1.0 (1.0)
SDK Version	3.1.1.4
Last Check-in	3/1/17, 5:53 PM
Authorization Status	Authorized ✓

POLICIES AND CONFIGURATION

Open In	Allowed (All Apps)
Print	Allowed
Copy/Paste	Allowed
Configuration Count	2
AppTunnel	None

Debug Mode

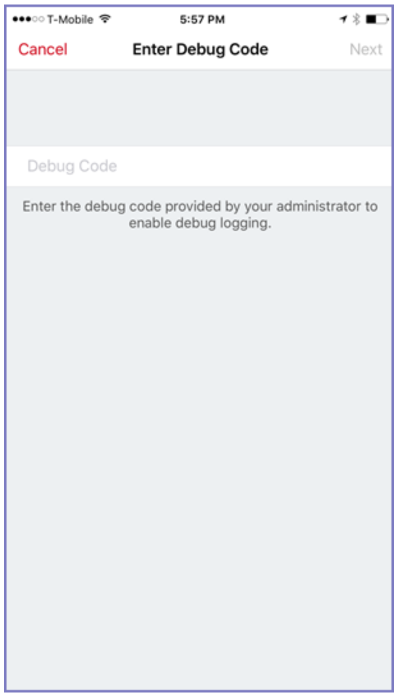
NOTE: The keys MI_AC_LOG_LEVEL and MI_AC_LOG_LEVEL_CODE are not included in the configuration count on an app's detailed status display.

Activating verbose or debug logging on the device

The following describes how to activate verbose or debug logging on the device.

Procedure

1. Open MobileIron Go on the device.
2. Tap **Settings**.
3. Tap **Check For Updates**.
4. Tap **Secure Apps**.
5. Tap the app for which you want verbose or debug level logging.
6. Slide the toggle for **Debug Mode**.



Enter the debug code provided by your administrator to enable debug logging.

7. Enter the debug code.
8. Tap **Next.**

Verbose or debug level logging is activated for 24 hours, after which it is automatically deactivated the next time that the device user launches or switches to the app. However, the device user can deactivate it any time by tapping **Debug Mode** again.

Emailing log files from MobileIron Go

MobileIron Go displays the option to send logs on the app's status details screen, available in MobileIron Go at **Settings > Secure Apps > <*app name*>.** The option is at the bottom of the screen with this text: **Send <*app name*> Logs.** However, the option is disabled if the app's AppConnect authorization status is not authorized.

When the option is displayed and enabled, tapping it brings up the list of apps able to share the log files, such as email apps, if you included the following key-value pair for the app in its AppConnect app configuration:

- **MI_AC_ENABLE_LOGGING_TO_FILE** set to **Yes**

For wrapped apps, you can also include the key **MI_AC_WR_ENABLE_LOG_CAPTURE** set to **Yes.** This key causes the app's logs to be included in the log files along with the logs from the AppConnect wrapper and AppConnect library.

NOTE: Once you have collected the logs from the device user, remove the **MI_AC_LOG_LEVEL, MI_AC_LOG_LEVEL_CODE**, and **MI_AC_ENABLE_LOGGING_TO_FILE** key-value pairs from the AppConnect app configuration. This best practice ensures the app does not continue logging sensitive data unnecessarily.

Secure Apps on Android Devices - User Perspective

From a device user perspective, AppConnect apps are called secure apps. You configure whether a device uses secure apps, and you determine which secure apps are downloaded and installed on the device. From the device user's perspective, a secure app:

- keeps its data secure.
  A secure app can share its data and files only with other secure apps.
- requires the device user to log in with a secure apps passcode, if you require one.
  Logging in one time with the secure apps passcode allows the device user to access all the secure apps.
- overlays its icon with a special badge that indicates it is a secure app.

MobileIron Go works with the Secure Apps Manager app to download, install, and manage the secure apps. The Secure Apps Manager is downloaded and installed along with the secure apps.

The device user does the following tasks relating to secure apps:

1. Downloading and installing the secure apps
2. Creating the secure apps passcode
3. Choosing a more complex AppConnect passcode

Also related to secure apps, the device user sees:

- Secure apps notifications
- Secure apps status bar icons
- Camera, gallery, and media player warning messages

Downloading and installing the secure apps

To download and install the secure apps on Android devices, the device user:

1. Starts MobileIron Go.
   If the device user does not see **Secure Apps** on MobileIron Go, you have not configured the device to use secure apps.
2. Follows the instructions to install secure apps, including the Secure Apps Manager.
3. Continues to Creating the secure apps passcode.

Creating the secure apps passcode

After the device user downloads and installs all his secure apps, he creates a passcode for the secure apps if you require one. Logging in one time provides access to all the secure apps.

NOTE: The secure apps passcode is not the same passcode as the device password, if the device has one. The device user can choose the same values for both the secure apps passcode and the device password, or choose a different value for each of them.

To create the secure apps passcode, the device user:

1. Completes the steps in Downloading and installing the secure apps.
2. Follows the instructions on the **Passcode Setup** screen, entering a new secure apps passcode, and then reentering it.
   The device user must adhere to the passcode requirements that are stated on the screen.

After creating the secure apps passcode, a lock icon appears in the status bar.

Related topics

Device User impact of fingerprint login for AppConnect for Android

Choosing a more complex AppConnect passcode

Secure Apps Manager allows the device user to create a more complex AppConnect passcode than you require. This capability gives device users more flexibility in their passcode choice while still meeting your minimum security requirements.

Specifically, the feature works as follows. In the AppConnect global policy, you specify whether the type of the AppConnect passcode must be numeric or alphanumeric. Secure Apps Manager allows the device user to enter non-numeric characters when you specify the type as numeric.

The following table shows Secure Apps Manager behavior depending on the specified AppConnect passcode type and minimum length specified in the AppConnect global policy:

TABLE 30. SECURE APPS MANAGER BEHAVIOR WHEN PROMPTING FOR APPCONNECT PASSCODE

AppConnect passcode type	AppConnect passcode length	Secure Apps Manager behavior
Numeric	4	Numeric keypad with the option **Create more complex passcode**. When the user taps the option, an alphanumeric keyboard displays.
Numeric	Anything except 4	Alphanumeric keyboard
Alphanumeric	Any	Alphanumeric keyboard

Note The Following:

- Because using a length of 4 with type numeric is the most common use of numeric passcodes, it is the only case when Secure Apps Manager displays a numeric keypad.
- Consider the case when the device user switches from the numeric keypad to the alphanumeric keyboard to create the AppConnect passcode. Even if the created passcode contains only digits, when the device user needs to enter the passcode again, Secure Apps Manager will present the alphanumeric keyboard.

Secure apps notifications

Throughout the steps for setting up secure apps on a device, and after the steps are completed, the device user receives notifications about the status of MobileIron Go and secure apps. For example, a notification indicates whether the device user has logged in with the secure apps passcode.

When the device user powers on the device, a notification indicates that the user has not logged in with his secure apps passcode, and that the user has no email connection. The device user must log in to access secure apps.

To log in, the device user:

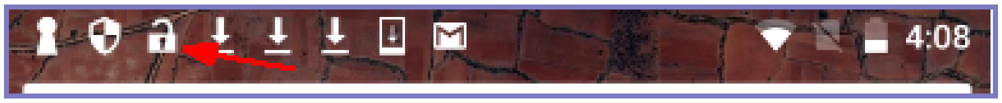
1. Opens any secure app or the Secure Apps Manager.
2. Enters his secure apps passcode.

Some secure apps, such as the email app, are active even when the device user is not using them. For example, the email app syncs email and calendar items. Until the device user logs in with his secure apps passcode, these apps cannot do their jobs.

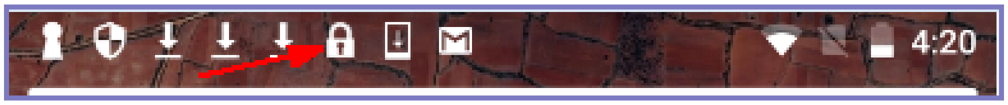
Secure apps status bar icons

A secure apps icon appears in the status bar of the device.

When the device user has entered his secure apps passcode, the icon looks like a lock that is unlocked, because the user has unlocked the AppConnect container and can access AppConnect apps:

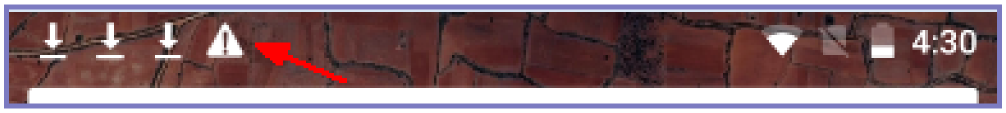


When logged out of secure apps, the icon looks like a lock that is locked, because the user is locked out of the AppConnect container. To unlock the container and access AppConnect apps, the user must enter his secure apps passcode.



For example, the device user is logged out when he has not used a secure app for five minutes.

The secure apps icon turns into a warning icon in some situations:



The warning icon appears when the device user needs to reenter his secure apps passcode, such as after powering on the device.

Camera, gallery, and media player warning messages

You can allow or prohibit secure apps on a device to do the following:

- access camera photos from the app
- access gallery images from the app
- stream media from the app to a media player

If a capability is prohibited, if an app attempts to use the capability, a message displays indicating that the administrator has disabled the capability.

If you allow accessing camera photos from secure apps, when an app accesses the camera, the app displays a warning. The warning indicates that the photo will not be secured, and that a photo from an unsecured camera app may compromise secure data.

If you allow accessing gallery images from secure apps, when an app accesses an image, the app displays a warning. The warning indicates that the image will not be secured and that an image from an unsecured app may compromise secure data.

If you allow media streaming from secure apps, when an app is about to stream media, the app displays a warning. The warning indicates that media will be streamed outside the secure container.

The warnings also provide the option to turn off future warnings.

Secure apps on iOS Devices - User Perspective

From a device user perspective, AppConnect apps are called secure apps. Secure apps on iOS devices allow the device user to securely access sensitive work documents and data on the device. The device user perspective includes the following:

- Secure apps passcode management

  Device users use a secure apps passcode to access secure apps. They use MobileIron Go to manage their secure apps passcode.
- Touch ID or Face ID with fallback to device passcode -- device user perspective

  Device users sometimes use Touch ID or Face ID to access secure apps. Most customers use Touch ID or Face ID with fallback to device passcode.
- Touch ID or Face ID with fallback to AppConnect passcode -- device user perspective

  Some customers with restrictions on requiring device passcodes want to allow device users to access secure apps with Touch ID or Face ID. These customers use Touch ID or Face ID with fallback to AppConnect passcode.

MobileIron Go app also provides displays to help you troubleshoot secure apps and AppTunnel. End users typically do not use these displays.

Secure apps passcode management

Typically, you configure AppConnect to require the device user to use a secure apps passcode to use secure apps. The device user creates and uses a secure apps passcode as follows:

- Creating a secure apps passcode
- Creating a more complex secure apps passcode
- Logging in with the secure apps passcode
- Logging out or resetting passcode for secure apps
- Secure apps passcode management
- Resetting the secure apps passcode - administrator initiated
- Secure apps passcode management

Creating a secure apps passcode

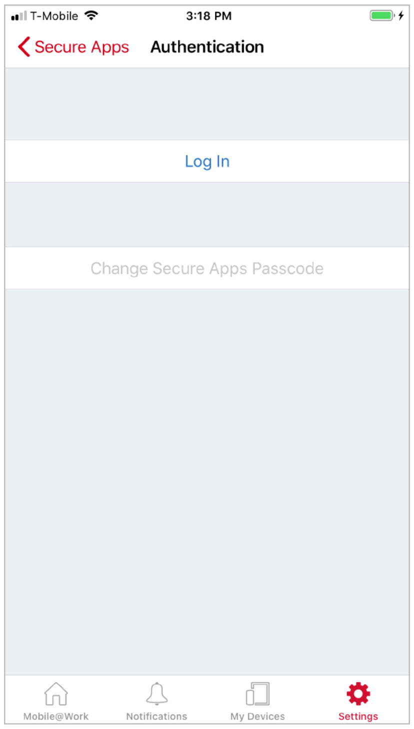
When you have configured a device so that a secure apps passcode is required, MobileIron Go prompts the device user to create a secure apps passcode the first time the user launches any secure app.

Device users can also create a secure apps password in MobileIron Go without first having to launch a secure app.

Procedure

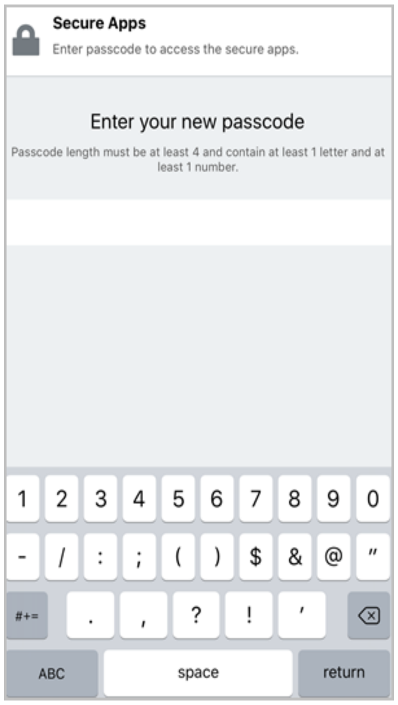
1. Launch MobileIron Go.
2. Go to Settings > Secure Apps > Authentication.

FIGURE 12. LOG IN FOR SECURE APPS PASSCODE



3. Tap Log In.

FIGURE 13. ENTER NEW PASSCODE



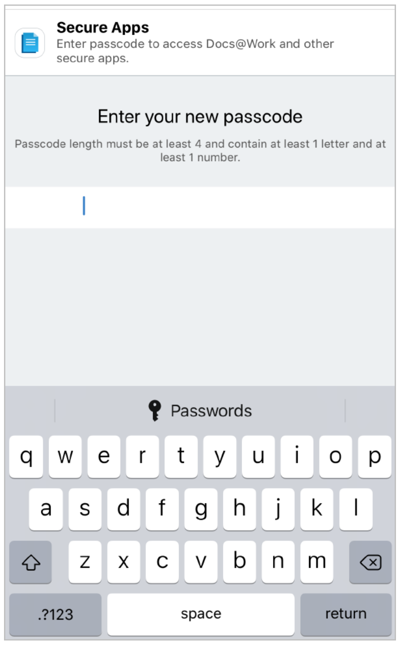
Secure Apps
Enter passcode to access the secure apps.

Enter your new passcode

Passcode length must be at least 4 and contain at least 1 letter and at least 1 number.

4. Enter a passcode according to the specified instructions.
5. Tap **Done.**

FIGURE 14. RE-ENTER THE NEW PASSCODE



Secure Apps
Enter passcode to access Docs@Work and other secure apps.

Enter your new passcode

Passcode length must be at least 4 and contain at least 1 letter and at least 1 number.

6. Tap **Done** and **Done** again.

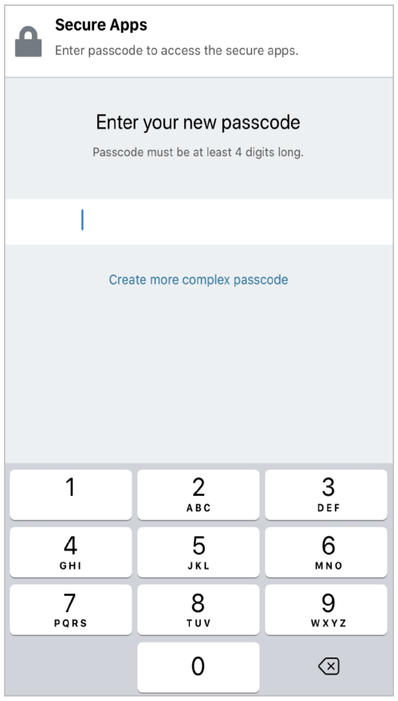
Creating a more complex secure apps passcode

MobileIron Go chooses which keyboard to display for entering a secure apps passcode based on the passcode requirements in the AppConnect global policy. For example, on an iPhone, when the AppConnect global policy requires a numeric passcode, MobileIron Go displays a numeric keypad. However, MobileIron Go gives the device user the option to enter a more complex secure apps passcode. Some users may want to choose to exceed the secure apps passcode requirements because:

- they value stronger security against guessing and brute force attacks
- they do not mind the reduced convenience of entering a more complex passcode.

If the secure apps passcode requirements in the AppConnect global policy are 4 numeric digits, MobileIron Go displays the following:

FIGURE 15. NUMERIC PASSCODE REQUIREMENT


Secure Apps

Enter passcode to access the secure apps.

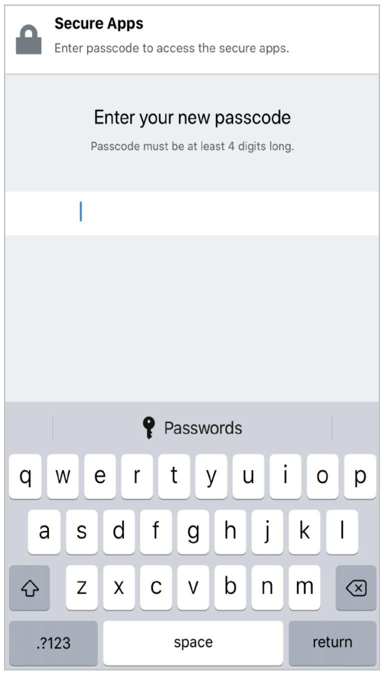
Enter your new passcode

Passcode must be at least 4 digits long.

Create more complex passcode

MobileIron Go presents a QWERTY keyboard when you tap **Create more complex passcode**.

FIGURE 16. ALPHA NUMERIC PASSCODE REQUIREMENT


Secure Apps

Enter passcode to access the secure apps.

Enter your new passcode

Passcode must be at least 4 digits long.

Use this screen to create a secure apps passcode that is more complex than required by the AppConnect global policy.

The device user has the option to create a more complex passcode when:

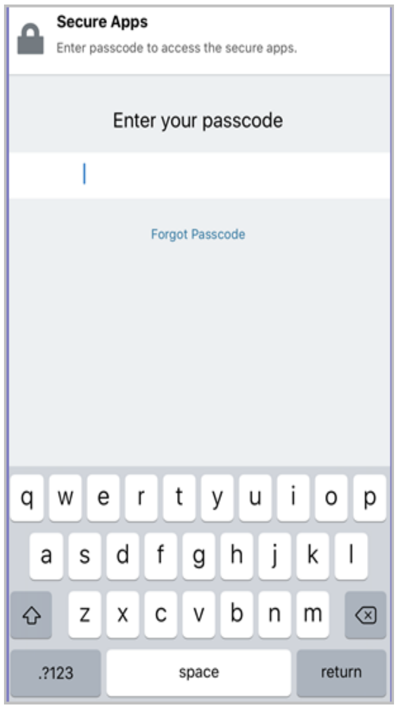
- Creating the secure apps passcode for the first time.
- Changing the secure apps passcode.
- After tapping **Forgot Passcode** and reentering their user name and password for MobileIron Cloud.
- After exceeding the maximum number of failed passcode attempts and reentering their user name and password for MobileIron Cloud.

NOTE: The last two options involve self-service secure apps passcode recovery, which is available only if you select **Allow iOS users to recover their passcode** on the AppConnect global policy.

Logging in with the secure apps passcode

After a period of time in which the device user uses no secure apps, MobileIron Go automatically logs the device user out of secure apps. When the user once again launches a secure app or taps **Log In** in MobileIron Go, MobileIron Go prompts the user to log in with the secure apps passcode:

FIGURE 17. LOGGING WITH SECURE APPS PASSCODE



The device user does the following:

1. Enters the secure apps passcode.
2. Taps **Done**.

The device user can now continue with the secure app.

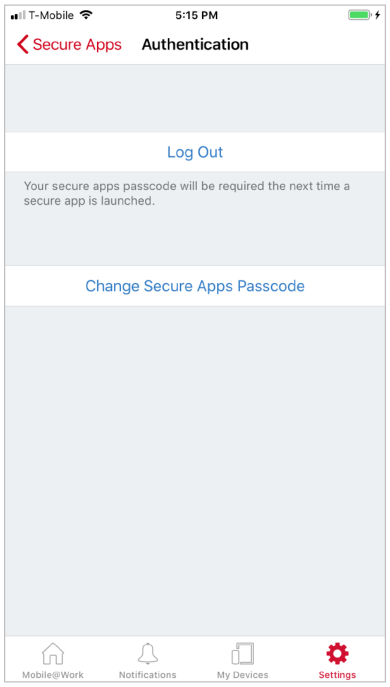
Logging out or resetting passcode for secure apps

The device user can log out of secure apps or reset the secure passcode. Logging out is useful, for example, if the user is lending the mobile device to a family member for a few minutes.

NOTE: The user is automatically logged out after a period of inactivity.

To log out of secure apps or reset the secure apps passcode, in MobileIron Go go to Settings > Secure Apps > Authentication.

FIGURE 18. SECURE APPS LOG OUT OR CHANGE PASSCODE



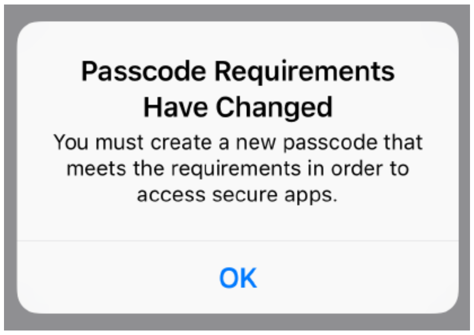
Your secure apps passcode will be required the next time a secure app is launched.

MobileIron Go prompts the device user for the secure apps passcode the next time the user launches a secure app or taps **Log In** in MobileIron Go.

Resetting the secure apps passcode - administrator initiated

You can change the secure apps passcode requirements on MobileIron Cloud by modifying the **iOS AppConnect Configuration**. When MobileIron Go checks in with Cloud, MobileIron Go prompts the device user to create a new password.

FIGURE 19. RESET PASSCODE PROMPT



Passcode Requirements Have Changed

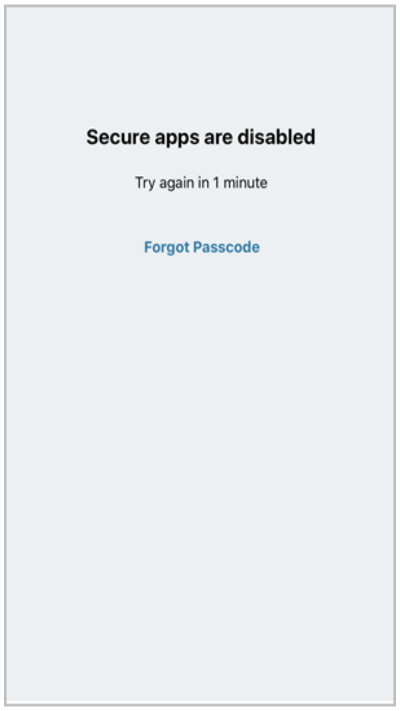
You must create a new passcode that meets the requirements in order to access secure apps.

Tap **OK** and follow the prompts to reset the passcode.

When the device user exceeds the maximum number of attempts

The maximum number of attempts to correctly enter the secure apps passcode is configurable. If it is greater than 5, after the device user makes five attempts to correctly enter the secure apps passcode, MobileIron Go displays the following:

FIGURE 20. SECURE APPS IS DISABLED



Secure apps are disabled

Try again in 1 minute

Forgot Passcode

After the maximum number of failed attempts, the device user must enter their Cloud credentials and then create a new AppConnect passcode. If the maximum is greater than 5, after the 5th attempt, the user can attempt to reenter the secure apps passcode only after waiting progressively longer time periods. Specifically, after the 5th, 6th, 7th, 8th, and 9th attempts, the user must wait 1, 5, 15, 60, and 60 minutes respectively.

Touch ID or Face ID with fallback to device passcode -- device user perspective

You can allow the device user to use Touch ID/Face ID instead of a secure apps passcode to access secure apps. Two options are available:

- Touch ID or Face ID with fallback to device passcode
- Touch ID or Face ID with fallback to AppConnect passcode

Most customers use Touch ID or Face ID with fallback to device passcode. With this option, the device user can do the following tasks using MobileIron Go:

- Choosing Touch ID or Face ID with fallback to device passcode to access secure apps
- Changing from secure apps passcode to Touch ID/Face ID to access secure apps
- Changing from Touch ID/Face ID to secure apps passcode to access secure apps

See also: Touch ID or Face ID for accessing secure apps for the administrative perspective.

NOTE: Screenshots in this chapter show only Touch ID, not Face ID, but Face ID behavior is similar.

Choosing Touch ID or Face ID with fallback to device passcode to access secure apps

The device user is prompted to choose whether to use Touch ID or Face ID to access secure apps when:

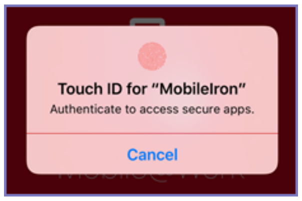
- On the AppConnect global policy, you have selected **Use Touch ID or Face ID when supported** and for **When using Touch ID or Face ID, fall back to** you have selected **Device passcode**.
- The device user has enabled the device passcode and at least one of Touch ID or Face ID.
- The device user has registered a device and then either
  - Accesses secure apps for the first time or
  - Taps **Log In** (to secure apps) on the MobileIron Go home screen

NOTE: MobileIron Go does not present this choice on devices on which the user has not enabled both Touch ID/Face ID and the device passcode, or the device does not support Touch ID/Face ID. For those devices, MobileIron Go prompts the device user to enter a new secure apps passcode.

The device user chooses Touch ID/Face ID

1. MobileIron Go prompts the device user to choose whether to use Touch ID/Face ID to access secure apps.
2. If the device user taps **Yes**, he is prompted for his fingerprint or Face ID.

   FIGURE 21. FINGERPRINT OR FACEID

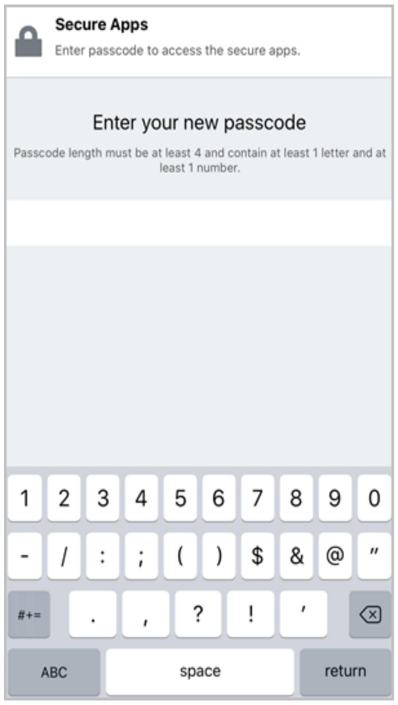
   Touch ID for "MobileIron"
   Authenticate to access secure apps.

3. The device user enters the Touch ID or Face ID and is logged into secure apps.

The device user will use Touch ID for all further authentications to secure apps, unless the device user changes the authentication method using **Settings > Secure Apps > Authentication** in MobileIron Go**.**

The device user chooses passcode

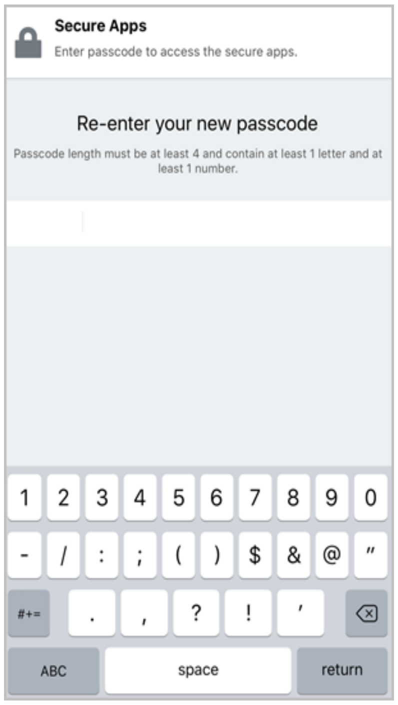
1. MobileIron Go prompts the device user to choose whether to use Touch ID or Face ID to access secure apps.
2. If the device user taps **No**, he is prompted to create a secure apps passcode.


Secure Apps
Enter passcode to access the secure apps.

Enter your new passcode

Passcode length must be at least 4 and contain at least 1 letter and at least 1 number.

3. The device user enters a new secure apps passcode.


Secure Apps
Enter passcode to access the secure apps.

Re-enter your new passcode

Passcode length must be at least 4 and contain at least 1 letter and at least 1 number.

4. The device user reenters the new passcode.

The device user will use the secure apps passcode for all further authentication to secure apps, unless the device user changes the authentication method using **Settings > Secure Apps > Authentication** in MobileIron Go.

Changing from secure apps passcode to Touch ID/Face ID to access secure apps

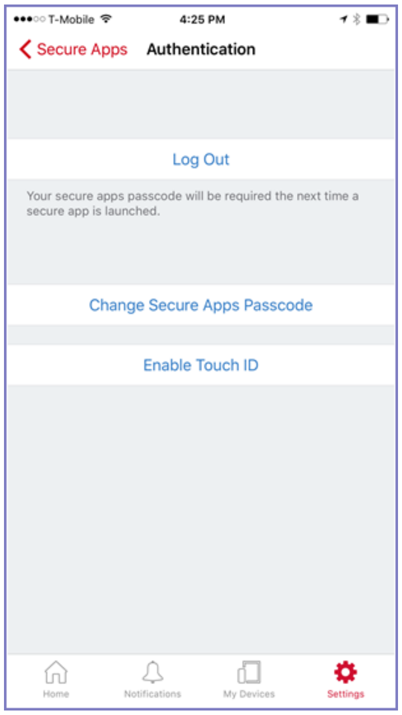
The device user can change the authentication method for accessing secure apps to Touch ID/Face ID when both of the following are true:

- You have selected **Use Touch ID or Face ID when supported** on the AppConnect global policy.
- The device user has enabled the device passcode and at least one of Touch ID or Face ID.

NOTE: Screenshots in this chapter show only Touch ID, not Face ID, but Face ID behavior is similar.

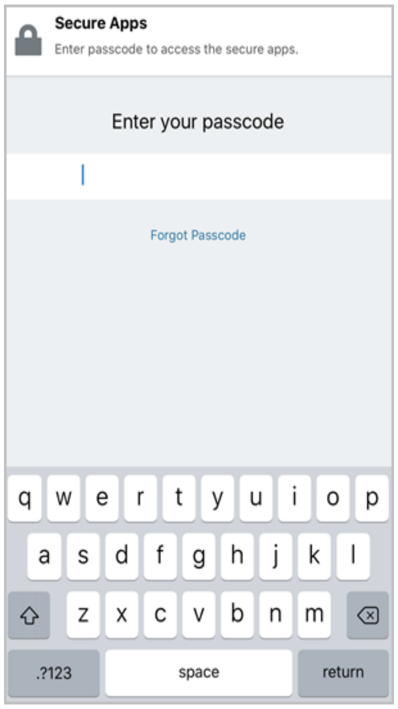
The device user does the following in MobileIron Go:

1. The device user navigates to **Settings > Secure Apps > Authentication.**

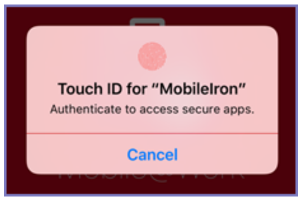


Your secure apps passcode will be required the next time a secure app is launched.

2. The device user taps **Enable Touch ID**.



3. The device user enters the secure apps passcode to confirm the change to using Touch ID/Face ID, and taps **Done**.


Touch ID for "MobileIron"
Authenticate to access secure apps.
Cancel

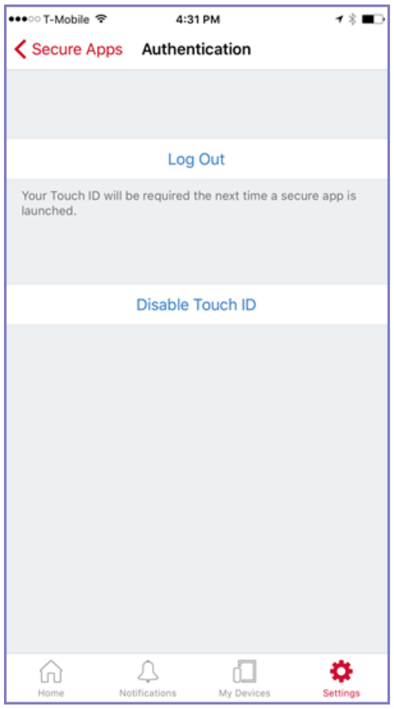
4. The device user enters the Touch ID or Face ID.

The device user will use Touch ID or Face ID for all further authentications to secure apps, unless the device user changes the authentication method using **Settings > Secure Apps > Authentication** in MobileIron Go**.**

Changing from Touch ID/Face ID to secure apps passcode to access secure apps

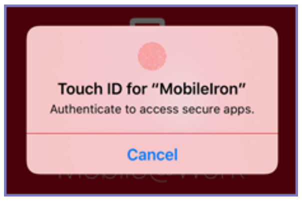
The device user can change the authentication method for accessing secure apps to the secure apps passcode using the following steps in MobileIron Go:

1. The device user navigates to **Settings > Secure Apps > Authentication.**



Your Touch ID will be required the next time a secure app is launched.

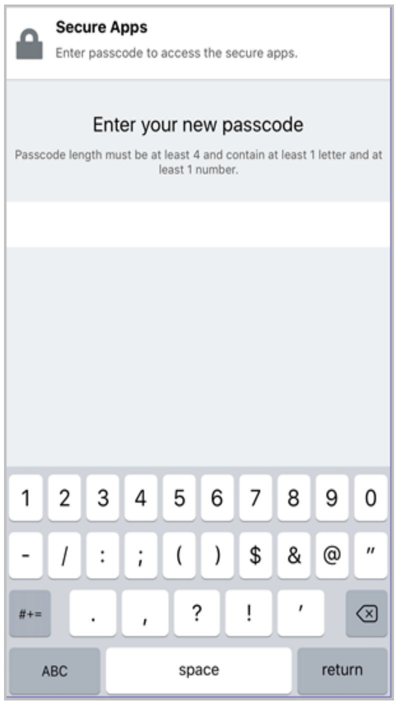
2. The device user taps **Disable Touch ID**.



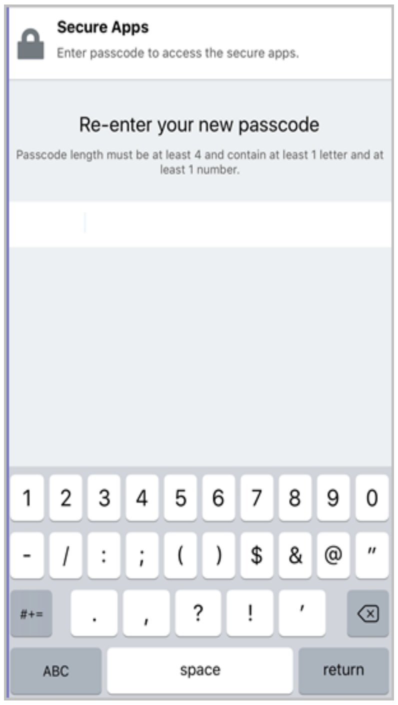
Touch ID for "MobileIron"

Authenticate to access secure apps.

3. The device user enters the Touch ID or Face ID to confirm the change to using a secure apps passcode.


Secure Apps
Enter passcode to access the secure apps.
Enter your new passcode
Passcode length must be at least 4 and contain at least 1 letter and at least 1 number.
ABC
space
return

4. The device user enters a new secure apps passcode and clicks **Done**.


Secure Apps
Enter passcode to access the secure apps.
Re-enter your new passcode
Passcode length must be at least 4 and contain at least 1 letter and at least 1 number.
ABC
space
return

5. The device user reenters the new passcode and clicks **Done**.

The device user will use the secure apps passcode for all further authentication to secure apps, unless the device user changes the authentication method using **Settings > Secure Apps > Authentication** in MobileIron Go.

Touch ID or Face ID with fallback to AppConnect passcode -- device user perspective

You can allow the device user to use Touch ID/Face ID instead of a secure apps passcode to access secure apps. Two options are available:

- Touch ID or Face ID with fallback to device passcode
- Touch ID or Face ID with fallback to AppConnect passcode

Although not the common choice, some customers use Touch ID or Face ID with fallback to AppConnect passcode when they have a compelling reason to not require a strong device passcode for device users.

NOTE: Screenshots in this chapter show only Touch ID, not Face ID, but Face ID behavior is similar.

The overall device user experience for a newly registered user is:

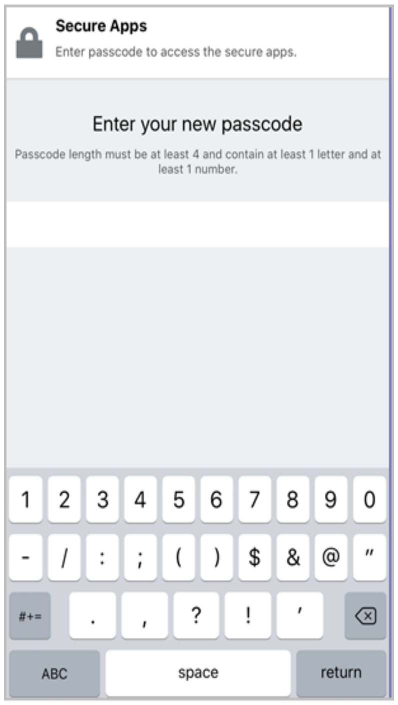
1. The device user creates an AppConnect passcode
   After the device user registers with MobileIron Go, MobileIron Go prompts the device user to create an AppConnect passcode.
2. The device user chooses whether to use Touch ID/Face ID.
   After creating the AppConnect passcode, MobileIron Go gives the user the option to use Touch ID or Face ID to access secure apps
3. The device user uses Touch ID/Face ID when the auto-lock time expires
   When the auto-lock time has expired, and the device user can use Touch ID or Face ID when re-accessing secure apps.
4. The device user changes Touch ID/Face ID choice
   The device user can later use MobileIron Go settings to change his choice about using Touch ID/Face ID.

See also: Touch ID or Face ID for accessing secure apps for the administrative perspective.

The device user creates an AppConnect passcode

MobileIron Go prompts the device user to create an AppConnect passcode when the device user has registered a device and then either:

- Accesses secure apps for the first time or
- Taps **Log In** (to secure apps) on the MobileIron Go home screen


Secure Apps
Enter passcode to access the secure apps.
Enter your new passcode
Passcode length must be at least 4 and contain at least 1 letter and at least 1 number.

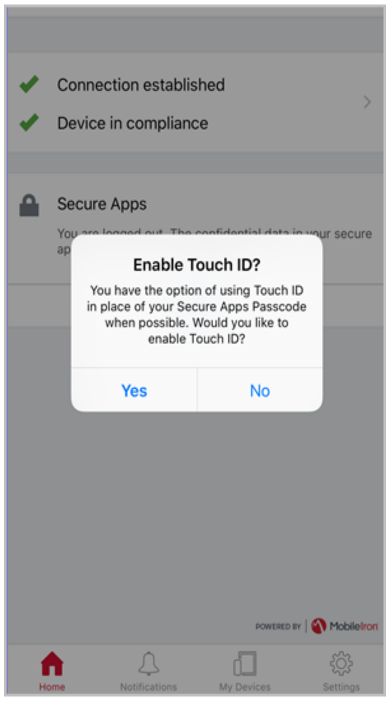
The device user chooses whether to use Touch ID/Face ID

NOTE: Screenshots in this chapter show only Touch ID, not Face ID, but Face ID behavior is similar.

After creating the AppConnect passcode, MobileIron Go gives the device user the choice to use Touch ID or Face ID with fallback to the AppConnect passcode, or to use only the AppConnect passcode for accessing secure apps. However, MobileIron Go gives this choice *only if* the device user has already done the following in the device's **Settings > Touch ID & Passcode**:

- Turned on the device passcode.
- Enabled Touch ID on the device, and created a fingerprint.

If the device user has taken these actions, MobileIron Go displays the following:


Connection established

Device in compliance

Secure Apps

You are logged out. The confidential data in your secure ap

Enable Touch ID?

You have the option of using Touch ID in place of your Secure Apps Passcode when possible. Would you like to enable Touch ID?

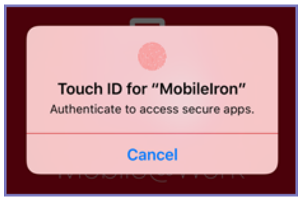
If the device user taps

- **Yes**, he will use Touch ID/Face ID when re-accessing secure apps after the auto-lock time expires. In all other cases for accessing secure apps he will enter the AppConnect passcode. These other cases include, for example, the first time an AppConnect app is launched or when the user logs out of secure apps in MobileIron Go.
- **No**, he will use the AppConnect passcode for all further authentications to secure apps.

The device user uses Touch ID/Face ID when the auto-lock time expires

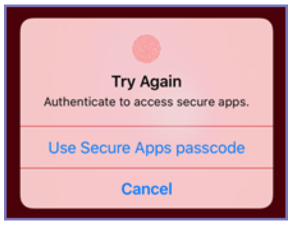
NOTE: Screenshots in this chapter show only Touch ID, not Face ID, but Face ID behavior is similar.

When the auto-lock time has expired, and the device user attempts to re-access secure apps, MobileIron Go displays the following:


Touch ID for "MobileIron"

Authenticate to access secure apps.

The device user enters the Touch ID or Face ID to access secure apps. If entering the Touch ID or Face ID fails, the device user is prompted to try again, and given the option to use (fallback to) the secure apps passcode:



Try Again

Authenticate to access secure apps.

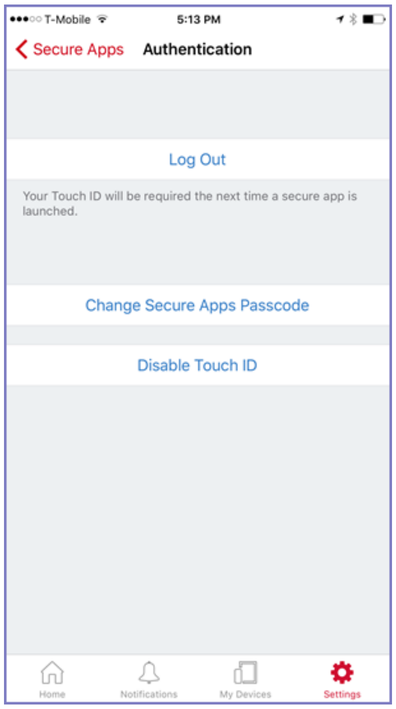
NOTE: Tapping either **Cancel** or **Use Secure Apps passcode** causes MobileIron Go to prompt the device user for the secure apps passcode.

The device user changes Touch ID/Face ID choice

NOTE: Screenshots in this chapter show only Touch ID, not Face ID, but Face ID behavior is similar.

The device user can change the choice to use Touch ID using **Settings > Secure Apps > Authentication** in MobileIron Go**.**

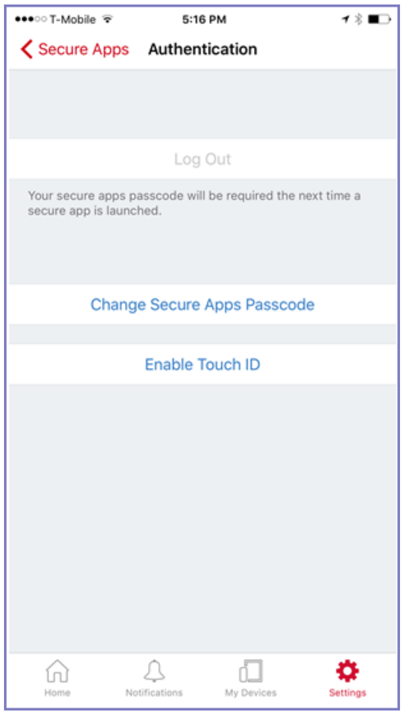
For example, if the device user is using Touch ID or Face ID, the screen displays the following:



Your Touch ID will be required the next time a secure app is launched.

If the device user taps **Disable Touch ID**, MobileIron Go will prompt for the secure apps passcode for all further access to secure apps.

To enable Touch ID/Face ID later, the device user can again navigate to **Settings > Secure Apps > Authentication** and tap **Enable Touch ID**.



Your secure apps passcode will be required the next time a secure app is launched.