



MobileIron Core 10.7.0.0 AppConnect and AppTunnel Guide

June 23, 2020

For complete product documentation see:
[MobileIron Core Product Documentation Home Page](#)

Copyright © 2012 - 2020 MobileIron, Inc. All Rights Reserved.

Any reproduction or redistribution of part or all of these materials is strictly prohibited. Information in this publication is subject to change without notice. MobileIron, Inc. does not warrant the use of this publication. For some phone images, a third-party database and image library, Copyright © 2007-2009 Aeleeeta's Art and Design Studio, is used. This database and image library cannot be distributed separate from the MobileIron product.

"MobileIron," the MobileIron logos and other trade names, trademarks or service marks of MobileIron, Inc. appearing in this documentation are the property of MobileIron, Inc. This documentation contains additional trade names, trademarks and service marks of others, which are the property of their respective owners. We do not intend our use or display of other companies' trade names, trademarks or service marks to imply a relationship with, or endorsement or sponsorship of us by, these other companies.



Contents

Contents	3
New Features and Enhancements	14
AppConnect and AppTunnel Overview	15
What are AppConnect-enabled apps?	15
AppConnect apps from MobileIron	16
Third-party and in-house AppConnect apps	16
AppTunnel overview	16
HTTP/S tunneling	17
TCP tunneling (also known as Advanced AppTunnel)	17
The AppConnect passcode	17
AppConnect apps and authentication to enterprise app servers	18
Authentication using Kerberos Constrained Delegation	19
Certificate authentication for Android AppConnect apps	19
Certificate authentication for iOS AppConnect apps	19
Authentication through MobileIron Access	19
App-specific configuration for AppConnect apps	20
AppConnect for Android overview	20
Wrapping modes	21
The MobileIron client app, the Secure Apps Manager, and the AppConnect wrapper	21
Supported Android device processors	22
Supported Android operating systems	22
Samsung Knox container (Knox Workspace) and AppConnect apps	22
AppConnect for Android component support and compatibility	22
Data loss prevention for secure apps for Android	23
Data encryption for secure apps for Android	23
Special badging for secure apps for Android	23



AppConnect for Android apps	23
Types of AppConnect Apps	24
AppConnect apps that MobileIron provides for Android	24
Docs@Work	24
Email+	25
Web@Work	25
File Manager	25
Other documentation about MobileIron-provided AppConnect apps	25
When an Android device user can use AppConnect for Android	25
AppConnect for iOS overview	26
Component support and compatibility	26
Wrapping support for mobile development platforms	26
Data loss prevention for secure apps for iOS	26
Data encryption for secure apps for iOS	26
AppConnect-related data	27
App-specific data	27
MobileIron UEM client for iOS and AppConnect apps	28
App check-in and MobileIron UEM client	28
The AppConnect passcode auto-lock time and MobileIron UEM client	29
Dual-mode apps	30
AppConnect apps that MobileIron provides for iOS	30
When an iOS device user can use AppConnect for iOS	31
Configuring AppConnect and AppTunnel	32
Configuration overview	32
Basic configuration	32
Adding third-party and in-house secure apps	33
Adding AppTunnel support	33
Adding compliance actions	34
AppConnect configuration tasks	34



Adding secure apps for deployment	34
AppConnect global policy	35
AppConnect passcode requirements	36
Configuring the AppConnect global policy	37
AppConnect global policy field description	37
Self-service AppConnect passcode recovery	53
AppConnect passcode strength	55
Mechanism to force all device users to change their AppConnect passcodes	57
Interaction with the lockdown policy regarding Android camera access	57
AppConnect container policies	58
AppConnect app authorization	58
Data loss prevention settings	59
Automatically created AppConnect container policies	59
Configuring AppConnect container policies	60
AppConnect container policy field description	61
Enabling secure apps	64
Enabling licensing options for Android secure apps	64
Enabling licensing options for iOS secure apps	64
Enabling AppTunnel	65
Configuring an AppTunnel service	65
About the AppTunnel service name	65
AppConnect app configuration	67
Automatically created AppConnect app configuration	67
Automatically provided key-value pairs	68
Configuring an AppConnect app configuration	68
Checking the device's labels	68
Adding a device to a label	68
AppConnect app configuration field description	69
Configuring the Open With Secure Email App option	74



	75
Configuring compliance actions	75
Quick start configuration for AppConnect for Android	76
Uploading the Secure Apps Manager to Core for Android AppConnect quick start	76
Uploading the AppConnect apps to Core for Android AppConnect quick start	77
Enabling Core licensing options for Android AppConnect quick start	78
Configuring the AppConnect global policy for Android AppConnect quick start	79
Configuring the AppConnect container policy for Android AppConnect quick start	80
Configuring settings specific to the app for Android AppConnect quick start	81
Configuring email attachment control for Android AppConnect quick start	83
Quick start configuration for AppConnect for iOS	83
Adding AppConnect apps to Core for iOS AppConnect quick start	84
Enabling Core licensing options for iOS AppConnect quick start	85
Configuring the AppConnect global policy for iOS AppConnect quick start	85
Configuring the AppConnect container policy for iOS AppConnect quick start	86
Configuring settings specific to the app for iOS AppConnect quick start	87
Configuring email attachment control for iOS AppConnect quick start	88
Using AppConnect for Android	90
Hybrid web app support	90
Fingerprint login for AppConnect apps for Android	91
Required product versions for fingerprint login for AppConnect for Android	92
Requirements for fingerprint login for AppConnect for Android	92
Configuring fingerprint login for AppConnect for Android (Core)	92
Device User impact of fingerprint login for AppConnect for Android	93
Device user experience at registration	93
Device user experience if already registered	94
Device user options for enabling or disabling fingerprint login	94
Less common device user scenarios for fingerprint login for AppConnect for Android	94
Security versus convenience of passcode and fingerprint for AppConnect for Android	96



AppTunnel with TCP tunneling support for Android secure apps	98
Types of apps that can use AppTunnel with TCP tunneling	99
When to use AppTunnel with HTTP/S tunneling versus TCP tunneling	100
Configuring AppTunnel with TCP tunneling for Android secure apps	100
Configuring an AppTunnel TCP service	101
About the AppTunnel TCP service name	101
Configuring the AppTunnel TCP service in the AppConnect app configuration	102
Configuring per-app idle session timeout for AppTunnel with TCP tunneling	103
Certificate authentication using AppConnect with TCP tunneling for Android secure apps	104
App and enterprise server requirements	104
Configuring certificate authentication using AppTunnel with TCP tunneling for Android secure apps	105
Overview	105
High-level tasks for certificate authentication using AppTunnel with TCP tunneling	106
Setting up the certificate for authenticating the user to the enterprise server	107
Specifying the AppTunnel services that use certificate authentication	107
Specifying which certificate to use to authenticate the user to the enterprise server	108
AppTunnel and TLS protocol versions in Android secure apps	108
Configuring the TLS protocol for AppTunnel	110
Lock, unlock, and retire impact on AppConnect for Android	110
Lock impact	110
Unlock the AppConnect container impact	111
Retire impact	111
Lock Android AppConnect apps when screen is off	112
Copy/Paste for AppConnect for Android	112
Comparison with AppConnect for iOS copy/paste policy	114
Copying from non-AppConnect apps to AppConnect apps	114
Interaction with Exchange setting	115
Sharing content from AppConnect for Android apps to non-AppConnect apps	115
Web-related DLP policies	116



Web DLP policy for browser launching	116
DLP allowing links from non-AppConnect apps to open in Web@Work	117
Web DLP versus Non-AppConnect apps can open URLs in Web@Work DLP	117
DLP policy for media player access	118
Media file requirements	118
Device-initiated security controls for AppConnect for Android	119
Configure the actions on the AppConnect global policy	119
Interaction with the Exchange setting	119
Custom keyboards in AppConnect apps	120
App whitelist	121
Key-value pair for the app whitelist	121
App whitelist examples	122
How the app whitelist is evaluated	123
Configuring an app whitelist	123
Secure File Manager features	124
Secure folder access	124
About allowing a secure app to ignore the auto-lock time	124
App requirements to ignore the auto-lock time	125
What the device user sees when an app ignores the auto-lock time	125
Situations that wipe Android AppConnect app data	126
Accessible Android apps to preserve the user experience	126
Secure Apps Manager Android permissions	127
Disabling analytics data collection for AppConnect for Android	128
Using AppConnect for iOS	129
Open-In data loss prevention policy details	129
Open In behavior in wrapped apps versus SDK apps	129
iOS native email use and the Open In DLP policy	132
Open In and native email with an AppConnect version prior to AppConnect 4.0 for iOS	132
Open In and native email with AppConnect 4.0 for iOS through most recently released version	133



Putting iOS native email into the Open In Whitelist	134
AirDrop use and the Open In DLP policy	134
App extension use and the Open In DLP policy	134
Whitelisting services integrated into iOS in the Open In DLP policy	134
Overriding the Open In policy for an app	135
Open From data loss prevention policy	136
Custom keyboard control	136
Screen blurring	137
Dictation with the native keyboard is not allowed for wrapped apps	138
Heightened security for AppConnect apps using the Secure Enclave	138
Situations that wipe AppConnect for iOS app data	139
Device-initiated (local) compliance for iOS jailbreak detection	139
Compliance actions for device-initiated jailbreak detection	140
Configuring device-initiated compliance for jailbreak detection	140
Creating a compliance action	140
Specifying the compliance action in the security policy	142
Touch ID or Face ID for accessing secure apps	142
Comparison of the two Touch ID/Face ID options	143
Security versus convenience of passcode and Touch ID/Face ID options	145
Touch ID or Face ID with fallback to device passcode	148
Switches to Mobile@Work eliminated with Touch ID or Face ID with fallback to device passcode ...	148
Improved user experience	149
Device passcode requirements	149
Device User impact in Mobile@Work	150
Configuring Touch ID or Face ID with fallback to device passcode	150
Less common device user scenarios	151
Touch ID or Face ID with fallback to AppConnect passcode	153
Improved user experience	154
Configuring Touch ID or Face ID with fallback to AppConnect passcode	154



Device User impact of Touch ID or Face ID with fallback to AppConnect passcode	155
Certificate pinning for AppConnect apps	156
About certificates used in certificate pinning	156
About domains in certificate pinning	157
Certificate pinning domains for root and intermediate CA certificates	157
Certificate pinning domains for leaf certificates	157
Configuring certificate pinning	158
Uploading the trusted certificates	158
Creating a Client TLS configuration	158
Modifying an AppConnect app configuration, Web@Work setting, or Docs@Work setting	159
Creating an AppConnect app configuration for the app if one does not already exist	159
Configuring the Client TLS configuration in the AppConnect app configuration	160
Viewing certificate pinning information in Mobile@Work	160
Certificate authentication from AppConnect apps to enterprise services	161
Impact on AppTunnel use	161
Setting up certificate authentication from an AppConnect app	162
Creating an AppConnect app configuration for the app if one does not already exist	162
Configuring the key-value pairs for the certificate and URL matching rule	163
Details about MI_AC_CLIENT_CERT_#_RULE	164
Impact to tunneling when using a global HTTP proxy	166
AppConnect Key-value Pairs Summary	168
AppConnect for Android key-value pairs	168
AppConnect Global policy key-value pairs	168
AppConnect app configuration key-value pairs	170
Secure Apps Manager app configuration key-value pairs	170
AppConnect for iOS key-value pairs	171
Troubleshooting AppConnect and AppTunnel for Android	174
Logging for AppConnect apps for Android	174
Turning on device log encryption on Android devices	174



State and encryption mode of Android secure apps	175
Status of AppConnect-related policies and configurations for an app	176
Troubleshooting AppConnect and AppTunnel for iOS	177
Logging for AppConnect apps for iOS	177
Overview of logging for AppConnect apps for iOS	177
Log levels	178
How the log level appears in messages	179
Log file details	180
Log data collection overview	181
Configuring logging for an AppConnect app	181
Creating a new label	182
Applying labels	182
Log level configuration impact on the device	183
Activating verbose or debug logging on the device	184
Emailing log files from Mobile@Work	184
Removing log level configuration when no longer needed	185
Secure apps status display in Mobile@Work	185
Navigating to the secure apps status display	186
The secure apps status display contents	186
Status details for a specific secure app	186
AppTunnel configuration troubleshooting display in Mobile@Work	190
Navigating to the AppTunnel configuration troubleshooting display	190
Troubleshooting with the AppTunnel configuration display fields	192
Client Certificate display	193
Specifying a trusted root certificate in the Standalone Sentry	195
Specifying a valid client certificate in the AppConnect app configuration	195
Rules display	196
Sentry display	197
Sentry Certificate display	199



Uploading a valid Sentry certificate to Standalone Sentry	200
AppTunnel configuration troubleshooting checklist	201
Status of AppConnect-related policies and configurations for an app	202
Viewing certificates stored in Mobile@Work	203
AppTunnel diagnostics in SDK-built apps	204
Secure Apps on Android Devices - User Perspective	205
Downloading and installing the secure apps	205
Creating the secure apps passcode	206
Choosing a more complex AppConnect passcode	206
Recovering the AppConnect passcode when forgotten	207
Secure apps notifications	207
Secure apps status bar icons	208
Camera, gallery, and media player warning messages	208
Secure apps on iOS Devices - User Perspective	210
Secure apps passcode management	210
Creating a secure apps passcode	211
Creating a more complex secure apps passcode	213
Logging in with the secure apps passcode	215
Logging out or resetting passcode for secure apps	215
Resetting the secure apps passcode - administrator initiated	216
When the device user exceeds the maximum number of attempts	217
Touch ID or Face ID with fallback to device passcode – device user perspective	218
Choosing Touch ID or Face ID with fallback to device passcode to access secure apps	219
The device user chooses Touch ID/Face ID	219
The device user chooses passcode	219
Changing from secure apps passcode to Touch ID/Face ID to access secure apps	221
Changing from Touch ID/Face ID to secure apps passcode to access secure apps	222
Touch ID or Face ID with fallback to AppConnect passcode – device user perspective	224
The device user creates an AppConnect passcode	225



The device user chooses whether to use Touch ID/Face ID	225
The device user uses Touch ID/Face ID when the auto-lock time expires	226
The device user changes Touch ID/Face ID choice	227



New Features and Enhancements

This guide documents the following new features and enhancements:

- **Enable split tunneling using MobileIron Tunnel:** iOS only. If you are transitioning from UIWebView to WKWebView and your app currently uses AppTunnel rules, a new option, **Enable Split Tunneling using MobileIron Tunnel**, is available on the MobileIron Core Admin Portal. The new option is available in the AppConnect App Configuration, Docs@Work configuration, and Web@Work configuration. Before enabling the option, ensure that MobileIron Tunnel is deployed. Enabling the option allows the configured AppTunnel rules to be managed through MobileIron Tunnel rather than through AppTunnel. The workaround is available due to the planned deprecation of the UIWebView API by Apple.

The feature requires Mobile@Work 12.3.0 and MobileIron Tunnel 4.1.0 for iOS.

For information about the UIWebView API deprecation, see [UIWebView Deprecation and AppConnect Compatibility](#).

For information about configuring AppConnect App Configuration, see **Enable Split Tunneling using MobileIron Tunnel** in [AppConnect app configuration field description](#).



AppConnect and AppTunnel 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. Policies and configurations set up in a MobileIron unified endpoint management (UEM) platform are used to manage the 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 and AppTunnel Overview](#)

AppConnect apps from MobileIron

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

License requirements for each MobileIron AppConnect app varies. The license requirements are listed in [Enabling secure apps](#).

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

License requirements for each MobileIron AppConnect app varies. The license requirements are listed in [Enabling secure apps](#).

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 UEM 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 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 for iOS and Android 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 secure apps](#)
- See "AppTunnel with Standalone Sentry" in the MobileIron Sentry Guide for MobileIron Core on the [MobileIron Sentry Product Documentation Home Page](#).
- For information about deploying MobileIron Tunnel for iOS, see *MobileIron Tunnel for iOS Guide for Administrators* on the [MobileIron Tunnel for iOS Product Documentation Home Page](#).

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 the administrative portal for the MobileIron UEM. 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 each device 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 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 the MobileIron UEM client 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 secure 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 with Core or Cloud, 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.
- If **Enable MobileIron Access** is selected in the AppConnect app configuration, AppTunnel traffic is trusted by MobileIron Access. The AppConnect app does not require Tunnel to authenticate through Access.

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](#).
- For information about Enable Mobile Access, see [AppConnect app configuration field description](#).

App-specific configuration for AppConnect apps

On the administrative portal for the UEMI, you can configure settings that are specific to an AppConnect app. Because MobileIron Core 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 documentation should specify the necessary configuration for the app.

AppConnect for Android overview

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

- [Wrapping modes](#)
- [The MobileIron client app, the Secure Apps Manager, and the AppConnect wrapper](#)
- [Supported Android device processors](#)
- [Supported Android operating systems](#)
- [Samsung Knox container \(Knox Workspace\) and AppConnect apps](#)
- [AppConnect for Android component support and compatibility](#)
- [Data loss prevention for secure apps for Android](#)
- [Data encryption for secure apps for Android](#)
- [Special badging for secure apps for Android](#)



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. Kiosk mode information is in "Android Kiosk Support" in the *MobileIron Core Device Management Guide for Android and Android enterprise Devices*.

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:

- the MobileIron client app for Android (MobileIron Go or Mobile@Work)
- the 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 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.

Therefore, for MobileIron Core deployments, upgrade devices to the corresponding version of Secure Apps Manager if you upgrade an app on the device to use a new wrapper version.



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 Core supports, see the *MobileIron Core and Connector Release Notes and Upgrade Guide*.

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.

NOTE: In a feature called AppConnect for Knox, Mobile@Work for Android uses Samsung Knox Platform features to provide an added layer of security. Note that AppConnect for Knox uses only the AppConnect container. The device cannot have the Samsung Knox container. For information about AppConnect for Knox, see "AppConnect for Samsung Knox devices" in the *MobileIron Core Device Management Guide for Android and Android enterprise Devices*

AppConnect for Android component support and compatibility

For the list of Secure Apps Manager versions that Core supports, see the *MobileIron Core and Connector Release Notes and Upgrade Guide*.

For the list of Mobile@Work versions supported with a particular Secure Apps Manager version, see the *AppConnect Secure Apps for Android Release Notes and Upgrade Guide* for the Secure Apps Manager version.



Regarding support of Secure Apps Manager versions with wrapper versions:

- When you upgrade to a new Secure Apps Manager, you do not need to re-wrap all your apps. A new Secure Apps Manager is supported with apps wrapped with the newest wrapper plus the two most recent older wrappers. That is, Secure Apps Manager is backward compatible.
- An app wrapped with a newer wrapper 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. Therefore, upgrade devices to the corresponding version of Secure Apps Manager if you upgrade an app on the device to use a new wrapper version.

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

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", meaning an app has root access).

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 the MobileIron UEM 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 for Administrators*
- *MobileIron Email+ for Android Guide for Administrators*
- *MobileIron Web@Work for Android Guide for Administrators*

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 UEM platform via a MobileIron UEM client:
 - The MobileIron unified endpoint management (UEM) platform are MobileIron Cloud, MobileIron Core.
 - The MobileIron UEM clients are MobileIron Go, Mobile@Work.
- 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.
 - 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 has been out of contact with Core 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 the MobileIron UEM client app for iOS. Unlike AppConnect for Android, AppConnect for iOS has no separate Secure Apps.

- MobileIron Go for MobileIron Cloud
- Mobile@Work for MobileIron Core

Component support and compatibility

For the supported versions of the various components in an AppConnect deployment, including the MobileIron UEM and MobileIron UEM client, 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, Open From), drag and drop, 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.
- Protected by the device passcode if the device user uses Touch ID / Face ID with fallback to device passcode to access AppConnect apps.
- Protected by the AppConnect passcode if the device user uses Touch ID / Face ID with fallback to AppConnect passcode to access AppConnect apps.

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 1. 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 UEM client for iOS and AppConnect apps

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

- Periodically does an app check-in with the MobileIron UEM 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.

The MobileIron UEM clients are, MobileIron Go for MobileIron Cloud deployments and Mobile@Work for MobileIron Core deployments

App check-in and MobileIron UEM client

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

- the AppConnect global policy for the device.
- AppConnect container policies for each of the AppConnect apps that have run on 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.

The MobileIron UEM client does an app check-in in the following situations:



- The device user launches an AppConnect app for the first time.
 - In this situation, the MobileIron UEM client 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 the UEM client does an app check-in, the UEM client comes to the foreground and the AppConnect app goes to the background momentarily. Once the UEM client has completed the app check-in, the AppConnect app returns to the foreground.

Note The Following:

- The Force Device Check-in feature on the MobileIron UEM does not sync the policies and settings related to AppConnect for iOS. The app check-in interval in the AppConnect global policy on MobileIron Core and in the AppConnect Device configuration on MobileIron Cloudcontrols these updates. However, in the MobileIron UEM client for iOS on the device, the Check for Updates option does sync the policies and settings related to AppConnect.
- When control switches to Mobile@Work due to an app check-in, Mobile@Work gets AppConnect policy updates from Core. However, Core indicates in the device details display that the policies are only "sent" or "pending" until the next app check-in. At the *next* app check-in, Core finds out whether an AppConnect app has applied the policies. If it has, Core indicates the policies are "applied" at that time.

The AppConnect passcode auto-lock time and MobileIron UEM client

The MobileIron UEM client (MobileIron Go or Mobile@Work) 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 Mobile@Work 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 UEM client 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 UEM and the MobileIron UEM client is installed on the device.
- You have configured MobileIron UEM 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 the MobileIron UEM client 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 Core.
The user must use the Mobile@Work for iOS app to register the device with MobileIron Core. 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. The MobileIron UEM client reports device information to the MobileIron UEM. The MobileIron UEM 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.

NOTE: AppConnect for iOS is not supported when a device is configured for single-app mode, which is described in "Single-app mode policies" in the *MobileIron Core Device Management Guide* for iOS.



Configuring AppConnect and AppTunnel

- [Configuration overview](#)
- [AppConnect configuration tasks](#)
- [Adding secure apps for deployment](#)
- [AppConnect global policy](#)
- [AppConnect container policies](#)
- [Enabling secure apps](#)
- [Enabling AppTunnel](#)
- [Configuring an AppTunnel service](#)
- [AppConnect app configuration](#)
- [Configuring the Open With Secure Email App option](#)
- [Configuring compliance actions](#)
- [Quick start configuration for AppConnect for Android](#)
- [Quick start configuration for AppConnect for iOS](#)

Configuration overview

The steps required to configure AppConnect depend on which aspects you intend to enable and deploy.

- [Basic configuration](#)
- [Adding third-party and in-house secure apps](#)
- [Adding AppTunnel support](#)
- [Adding compliance actions](#)

Basic configuration

Complete the following steps to implement a basic AppConnect configuration:

1. Add the MobileIron secure apps you intend to deploy.
These are AppConnect apps provided by MobileIron.
See [Adding secure apps for deployment](#).
2. Configure the AppConnect global policy.
See [AppConnect global policy](#).
3. Configure the AppConnect container policy.
See [AppConnect container policies](#).
4. Enable any MobileIron secure apps you intend to deploy.
See [Enabling secure apps](#).



5. Configure the app-specific configuration if required by the app.
See [AppConnect app configuration](#).
6. If you are using AppConnect-enabled email clients, configure the email attachment control option called Open With Secure Email App.
See [Configuring the Open With Secure Email App option](#)

Adding third-party and in-house secure apps

If you intend to deploy secure apps developed by your organization or a third-party provider, complete the following steps:

1. Complete the steps in [Configuration overview](#).
2. Enable AppConnect third-party and in-house apps.
See [Enabling secure apps](#).
3. If you are using AppConnect-enabled email clients, configure the email attachment control option called Open With Secure Email App.
See [Configuring the Open With Secure Email App option](#)

Adding AppTunnel support

Add AppTunnel support to secure the data that moves between your secure apps and your corporate data sources.

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 Core 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.

Procedure

The high-level tasks for configuring AppTunnel are:

1. Complete the steps in [Basic configuration](#).
2. Complete the steps in [Adding third-party and in-house secure apps](#), if applicable.
3. Enable AppTunnel on MobileIron Core, if you are deploying third-party or in-house apps.
See [Enabling AppTunnel](#).
4. Configure an AppTunnel service on Standalone Sentry.
See [Configuring an AppTunnel service](#).
5. Configure the AppTunnel rules in an AppConnect app configuration for each app using AppTunnel.
See [AppConnect app configuration](#).



Related topics

- “Configuring Standalone Sentry for AppTunnel” in the *MobileIron Sentry Guide*
- “Working with app tunnels” in the *MobileIron Sentry Guide* for actions you can take on an app tunnel. For example, you can block an app tunnel, so that an AppConnect app on a device cannot access the backend resource.

Adding compliance actions

You have the option of specifying AppConnect compliance actions as part of a security policy. To specify these compliance actions:

1. Complete the steps in [Configuration overview](#).
2. Complete the steps in [Configuration overview](#), if applicable.
3. Complete the steps in [Configuration overview](#), if applicable.
4. Configure compliance actions. See [Configuring compliance actions](#).

AppConnect configuration tasks

All the tasks related to AppConnect configuration are listed here. **See [Configuring AppConnect and AppTunnel](#) to determine which tasks you need to complete and in what order.**

- [Adding secure apps for deployment](#)
- [AppConnect global policy](#)
- [AppConnect container policies](#)
- [Enabling secure apps](#)
- [Enabling AppTunnel](#)
- [Configuring an AppTunnel service](#)
- [AppConnect app configuration](#)
- [Configuring the Open With Secure Email App option](#)
- [Configuring compliance actions](#)

Adding secure apps for deployment

You use the App Catalog on the Admin Portal to deploy secure apps. The App Catalog has two kinds of apps for both iOS and Android: in-house apps and public apps. In-house apps are uploaded to the App Catalog. Public apps are imported to the App Catalog from either Google Play or iTunes.

The following table shows the whether secure apps can be in-house or public apps:



TABLE 2. PLATFORM SUPPORT FOR IN-HOUSE AND PUBLIC SECURE APPS

OS	In-house secure apps can be...	Public secure apps can be...
Android	<ul style="list-style-type: none"> Secure apps from MobileIron Secure apps developed by your organization Secure apps developed by and received from a third party 	<ul style="list-style-type: none"> Not supported for secure apps
iOS	<ul style="list-style-type: none"> Secure apps from MobileIron Secure apps developed by your organization Secure apps developed by and received from a third party 	<ul style="list-style-type: none"> Third-party secure apps available in iTunes

Related topics

“Working with apps for iOS devices” and “Working with apps for Android devices” in the Apps@Work Guide

AppConnect global policy

The AppConnect global policy applies to all AppConnect apps on devices. These AppConnect apps include:

- third-party and in-house AppConnect apps
- Web@Work
- Docs@Work
- Email+
- other AppConnect apps that MobileIron provides

MobileIron Core applies a default AppConnect global policy automatically to all devices. You can modify the default AppConnect global policy. You can also create custom AppConnect global policies and apply those to specific devices.

IMPORTANT: If you are using AppConnect on iOS devices but not on Android devices, apply a separate AppConnect Global policy to Android devices. Do not use the same AppConnect Global policy for both iOS and Android devices. In the AppConnect Global policy for Android devices, ensure that for **AppConnect** the **Disabled** option is selected.

In the AppConnect global policy, you configure:

- whether AppConnect is enabled for the devices
- AppConnect passcode requirements



- out-of-contact timeouts
- the app check-in interval
- the default end-user message for when an app is not authorized
- whether AppConnect apps with no AppConnect container policy are authorized by default
See [AppConnect global policy](#)
- default settings for data loss prevention policies

AppConnect passcode requirements

On the AppConnect global policy, you specify whether the device user is required to enter an AppConnect passcode to access the AppConnect apps on the device. For the highest possible security when using AppConnect, MobileIron recommends that each device has both a device passcode and an AppConnect passcode. You can also allow users to use Touch ID or Face ID (iOS) or fingerprint (Android) instead of the AppConnect passcode to access secure apps. For more information about whether to require an AppConnect passcode, see:

- [The AppConnect passcode](#)
- [Data encryption for secure apps for Android](#)
- [Data encryption for secure apps for iOS](#)
- [Touch ID or Face ID for accessing secure apps](#)
- [Fingerprint login for AppConnect apps for Android](#)

NOTE: For Android, if users create an AppConnect passcode more than 60 minutes after registering the device, they must first enter their MobileIron Core credentials. After users create the AppConnect passcode, the AppConnect container is created.

When you require an AppConnect passcode, you also specify:

- Passcode Type
- Minimum Passcode Length
- Minimum Number of Complex Characters
- Maximum Passcode Age
- Auto-Lock Time
- Passcode history
- Maximum Number of Failed Attempts
- Passcode strength requirements

If the device user fails to correctly enter the AppConnect passcode after a certain number of attempts, the user cannot access AppConnect apps. Specifically:



- On iOS devices, device users must enter their Core credentials and then create a new AppConnect passcode.
- On Android devices, send an **Unlock AppConnect Container** command to the device from the Admin Portal. The command removes the secure apps passcode. The user can then create it again.

Related topics

- [Self-service AppConnect passcode recovery](#)
- [Mechanism to force all device users to change their AppConnect passcodes](#)

Configuring the AppConnect global policy

You configure the AppConnect global policy on MobileIron Core in **Policies & Configs > Policies**.

Procedure

1. In the Admin Portal, select **Policies & Configs > Policies**.
2. Edit the default AppConnect global policy, or select **Add New > AppConnect** to create a new one.
3. Enter the requested information.
4. Click **Save**.
5. If you created a new policy, apply the appropriate labels to the AppConnect global policy.
If you are using the default AppConnect global policy, it automatically applies to all devices.

Related topics

For a description of the fields in the AppConnect global policy, see [AppConnect global policy field description](#)

AppConnect global policy field description

Use the following guidelines to create or edit an AppConnect global policy:

TABLE 3. APPCONNECT GLOBAL POLICY FIELDS

Item	Description	Default Value
Name	<p>Required. Enter a descriptive name for this policy. This text is displayed to identify this policy throughout the Admin Portal. This name must be unique within this policy type.</p> <p>TIP: Though using the same name for different policy types is allowed (e.g., Executive), consider keeping the names unique to ensure clearer log entries.</p>	Default AppConnect Global Policy
Status	<p>Select Active to turn on this policy.</p> <p>Select Inactive to turn off this policy.</p>	Active
Priority	<p>Specifies the priority of this custom policy relative to the other custom policies of the same type. This priority determines which policy is applied if more than one policy is associated with a specific device. Select Higher than or Lower than, then select an existing policy from the dropdown list. For example, to give Policy A a higher priority than Policy B, you would select Higher than and Policy B.</p> <p>Because this priority applies only to custom policies, this field is not enabled when you create the first custom policy of a given type.</p> <p>For more information about policy priorities, see “Prioritizing policies” in <i>Getting Started with MobileIron Core</i>.</p>	
Description	Enter an explanation of the purpose of this policy.	Default AppConnect Global Policy
AppConnect	<p>Select Enabled to enable AppConnect on the device.</p> <p>Select Disabled to disable AppConnect on the device.</p> <p>When you select Enabled, the screen displays the rest of its fields.</p>	Disabled
AppConnect Passcode		
Passcode Type	<p>Specify the type of passcode:</p> <ul style="list-style-type: none"> • Numeric The passcode is allowed to have only digits in it. However, the device user can choose to create an alphanumeric passcode. 	Alphanumeric



TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
	<ul style="list-style-type: none"> • Alphanumeric The passcode must contain at least one digit and one letter. • Don't Specify The passcode is allowed to have characters of any type. 	
Minimum Passcode Length	Select a number between 1 and 16 to specify the minimum length for the passcode.	4
Minimum Number of Complex Characters	<p>Select a number between 1 and 10 to specify the minimum number of special characters that must be included in the passcode. Select “-” to require no special characters in the passcode.</p> <p>This option is only applicable when the passcode type is alphanumeric.</p> <p>NOTE: A special character is any character which is not 0-9, a-z, or A-Z. For example, \$, \, and ä are special characters.</p>	1
Maximum Passcode Age	<p>Enter a value between 1 and 730.</p> <p>This value is the number of days until the device user must change the secure apps passcode. The value is updated on a device when the next device check-in occurs. After the passcode age is exceeded (that is, the passcode expires), the device user is prompted for a new passcode the next time the device user attempts to login to secure apps. Device users must create a new passcode before they can access secure apps.</p> <p>If you do not want the passcode to expire, leave the field blank, which is the default.</p>	None
Auto-Lock Time	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.	15 minutes
Passcode history	<p>Select a value from 1 to 10, or “-”.</p> <p>This value specifies the number of most recently used secure apps passcodes that device users cannot use when changing their passcode.</p>	1



TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
	<p>The default value is 1, which means that when users create a new passcode, the only restriction is that they cannot reuse their current passcode.</p> <p>If you do not want a passcode history, select “-”. In this case, the user can reuse any previous passcode, including the current passcode.</p> <p>If you change this field value from none to a value between 1 and 10:</p> <ul style="list-style-type: none"> On iOS devices, the next time that users change the passcode, Mobile@Work puts the new passcode in the history. Therefore, after this policy change, users can reuse the current passcode the first time they change the passcode. On Android devices, the Secure Apps Manager puts the current passcode in the history <i>the next time that the user logs in</i>. Therefore, after this policy change, users who are <i>already logged in</i> can reuse the current passcode the first time they change the passcode. 	
Maximum Number of Failed Attempts	<p>Select a value between 2 and 10. Select “-” if you do not want to limit failed attempts.</p> <p>iOS</p> <p>If device users fail to correctly enter the AppConnect passcode after a certain number of attempts, they cannot access AppConnect apps.</p> <p>Device users must enter their Core credentials and then create a new AppConnect passcode.</p> <p>Android</p> <p>If device users fail to correctly enter the AppConnect passcode after a certain number of attempts, the AppConnect app is blocked or retired depending on the action selected for Maximum Number of Failed Attempts Action</p> <p>Send an Unlock AppConnect Container command to the Android device from the Admin Portal. The command removes the secure apps passcode. Users can then create it again.</p> <p>NOTE: If the maximum is greater than 5, after the</p>	10

TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
	<p>5th attempt, the user can attempt to reenter the secure apps passcode only after waiting progressively longer time periods.</p> <p>Related topics</p> <p>Self-service AppConnect passcode recovery</p>	
Maximum number of failed attempts action	<p>Android only.</p> <p>Block: Select to block the AppConnect app if the AppConnect passcode retry attempts exceed the configured maximum number of failed attempts.</p> <p>Retire: Select to retire the AppConnect app if the AppConnect passcode retry attempts exceed the configured maximum number of failed attempts. When AppConnect apps are retired, they become unauthorized (blocked), and the secure data is deleted (wiped). The app remains functional with only the unsecure data.</p> <p>NOTE: On Android devices, fingerprint login does not work if the maximum failed attempts exceeds the configured value.</p>	Block
Passcode is required for iOS devices	Select this field if you require device iOS users to enter an AppConnect passcode to use any AppConnect app.	Not selected
Use Touch ID or Face ID when supported	<p>This option is available only if you selected Passcode is required for iOS devices.</p> <p>Select this field to allow device users to enter their Touch ID (fingerprint) or Face ID, if available, to access secure apps.</p> <p>Important: When you select this option, another option appears that begins When using Touch ID or Face ID, fall back to: Most customers keep the default which is Device passcode. Selecting AppConnect passcode is less frequently used.</p> <p>Related topics</p> <p>Touch ID or Face ID for accessing secure apps</p>	Not selected
When using Touch ID or Face	These options are available only if you selected Use Touch ID or Face ID when supported .	Device Passcode



TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
ID, fall back to:	<p>Most customers keep the default which is Device passcode. This 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.</p> <p>With the AppConnect passcode option, when the auto-lock time for AppConnect apps expires, the device user uses Touch ID or Face ID rather than the AppConnect passcode to re-access AppConnect apps. If entering the Touch ID or Face ID fails, the user enters (falls back to) the AppConnect passcode to access secure apps. The device user also uses the AppConnect passcode for other situations requiring AppConnect authentication such as the first time an AppConnect app is launched or when the user logs out of secure apps in Mobile@Work.</p> <p>IMPORTANT: Use the option AppConnect passcode only if you have a compelling reason to not require your users to have a strong device passcode.</p> <p>Related topics</p> <p>Touch ID or Face ID for accessing secure apps</p>	
Allow iOS users to recover their passcode	<p>This option is available only if you selected Passcode is required for iOS devices.</p> <p>Select this option to allow a device users to recovery their AppConnect passcode themselves.</p> <p>This option defaults to allowed. If you disable this option, <i>no method is available to recover a forgotten AppConnect passcode on iOS devices.</i> The device user must delete and reinstall Mobile@Work and each AppConnect app.</p> <p>Related topics</p> <p>Self-service AppConnect passcode recovery</p>	Selected
Lock AppConnect apps	<p>This option is available only if you selected Passcode is required for iOS devices.</p>	Not selected

TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
automatically when the screen is off	<p>Select this option to automatically log out device users from AppConnect apps when the device screen is turned off due to either inactivity or user action.</p> <p>To access AppConnect apps after unlocking the screen (whether or not unlocking the screen requires user authentication), the device user must re-enter the AppConnect passcode (or Touch ID/Face ID).</p> <p>This setting requires:</p> <ul style="list-style-type: none"> • Mobile@Work 10.0.0 for iOS through the most recently released version as supported by MobileIron. • AppConnect apps built or wrapped with AppConnect 4.1 for iOS through the most recently released version as supported by MobileIron. <p>Previous versions of these components ignore this setting.</p>	
Passcode is required for Android devices	Select this field if you require Android device users to enter an AppConnect passcode to use any AppConnect app.	Selected
Allow Android users to recover their passcode	<p>This option is available only if you selected Passcode is required for Android devices.</p> <p>This option defaults to not allowed.</p> <p>Related topics</p> <p>Self-service AppConnect passcode recovery</p>	Not selected
Use fingerprint authentication when supported	<p>This option is available only if you selected Passcode is required for Android devices.</p> <p>Select this option to give the device user the convenience of using a fingerprint instead of an AppConnect passcode to access AppConnect apps on Android devices.</p> <p>Related topics</p> <p>Fingerprint login for AppConnect apps for Android</p>	Not selected
Lock AppConnect apps	This option is available only if you selected Passcode is required for Android devices .	Not selected



TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
automatically when the screen is off	<p>Select this option to automatically log out device users from AppConnect apps when the device screen is turned off due to either inactivity or user action.</p> <p>To access AppConnect apps after unlocking the screen (whether or not unlocking the screen requires user authentication), the device user must re-enter the AppConnect passcode (or fingerprint).</p> <p>This setting is supported with Secure Apps Manager 8.3.0 through the most recently released version as supported by MobileIron. Previous versions of the Secure Apps Manager ignore this setting.</p> <p>Related topics</p> <p>Lock Android AppConnect apps when screen is off</p>	
Check for AppConnect passcode strength	<p>Select this option if you want to set a required level of AppConnect passcode strength.</p> <p>When you select this option, a slider displays. Use the slider to select the desired AppConnect passcode strength, or enter a value between 0 and 100 in the text field.</p> <p>Related topics</p> <p>AppConnect passcode strength</p>	Not selected
Passcode Strength	<p>This option is available only if you selected Check for AppConnect passcode strength.</p> <p>Use the slider to select the desired AppConnect passcode strength, or enter a value between 0 and 100 in the text field.</p> <p>Related topics</p> <p>AppConnect passcode strength</p>	35
End User Terms of Service		
Enable End User Terms of Service	<p>When selected, end users must accept the terms of service before they can access AppConnect apps.</p> <p>However, selecting this option does not present users with the terms of service if either of the following are true:</p> <ul style="list-style-type: none"> • The terms of service is not configured. • The AppConnect passcode is not required. That is: 	Not selected

TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
	<ul style="list-style-type: none"> - Passcode is required for Android devices is not selected for Android devices. - Passcode is required for iOS devices is not selected for iOS devices. <p>For details about configuring the terms of service, see "Terms of service" in the <i>MobileIron Core Device Management Guide</i>.</p> <p>This feature requires:</p> <ul style="list-style-type: none"> • For iOS devices: Mobile@Work 10.0.2 for iOS through the most recently released version as supported by MobileIron. • For Android devices: Mobile@Work 10.1 for Android and Android Secure Apps Manager 8.4.0 through the most recently released versions as supported by MobileIron. 	
End User Terms of Service Frequency	<p>Select one of the following:</p> <ul style="list-style-type: none"> • Always: End users must accept the terms of service each time they are prompted to enter their AppConnect passcode or biometric identification. • Once: End users must accept the terms of service only one time. On iOS devices, this occurs when they create their AppConnect passcode. On Android devices, this occurs when they are first prompted to enter their AppConnect passcode or biometric identification. <p>If the terms of service is updated in the user's language after the user has accepted it, the user will be asked to accept it once again.</p>	Always
AppConnect Security Controls On Device		
Device Out Of Contact		
Wipe AppConnect Apps After	<p>Specify a value from 1 through 90 days. Leave the field empty if you do not want to wipe AppConnect apps when the device is out of contact with MobileIron Core.</p> <p>Once the AppConnect global policy is applied to the device, wiping the AppConnect apps occurs on the device after the specified time without reconnecting to</p>	30 days

TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
	MobileIron Core.	
Android		
Device Compromised	<p>Android only:</p> <p>Select Wipe AppConnect data if you want to retire all secure apps when the device is compromised (rooted). When secure apps are retired, they become unauthorized (blocked), and their data is deleted (wiped).</p> <p>After the device has checked in and received the AppConnect global policy, no further interaction is required from Core. Mobile@Work detects when the device is compromised and retires the secure apps.</p> <p>Select None to cause no action when the device is compromised.</p> <p>Related topics</p> <p>Device-initiated security controls for AppConnect for Android</p>	None
USB Debug Enabled	<p>Android only:</p> <p>Select Wipe AppConnect data if you want to retire all secure apps when USB debugging is enabled on the device. When secure apps are retired, they become unauthorized (blocked), and their data is deleted (wiped).</p> <p>After the device has checked in and received the AppConnect global policy, no further interaction is required from Core. Mobile@Work detects when USB debugging is enabled and retires the secure apps.</p> <p>Select None to cause no action when USB debugging is enabled.</p> <p>Related topics</p> <p>Device-initiated security controls for AppConnect for Android</p>	None
Remove AppConnect apps (and apps data) when device is retired	<p>Android only:</p> <p>Select this option to remove AppConnect apps and associated apps data when the device is retired. The policy silently removes the apps from Samsung Knox</p>	None



TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
	devices, and prompts the user to uninstall the apps from other types of devices.	
iOS		
Enable mutli-user auto sign-out after X minutes of inactivity	<p>iOS only:</p> <p>Select this option to configure automatic sign-out on multi-user iOS devices.</p> <p>Valid values are from 5 minutes to 120 minutes.</p> <p>This feature requires Mobile@Work 11.0 for iOS through the most recently released version as supported by MobileIron.</p> <p>Related topics</p> <p>"Setting automatic sign-out for multi-user devices" in the <i>MobileIron Core Device Management Guide for iOS and macOS Devices</i>.</p>	Not selected
App Authorization		
App Check-in Interval	<p>iOS only:</p> <p>Select the maximum number of minutes until devices running AppConnect apps receive updates of their AppConnect global policy, their AppConnect app configuration, and their AppConnect container policies.</p> <p>Note that these policies and settings are not updated on the device when:</p> <ul style="list-style-type: none"> the device checks in at its regular sync interval. you force a device check-in from the Devices & Users screen. <p>However, in the Mobile@Work for iOS app on the device, the Check for Updates option <i>does</i> sync the policies and settings related to AppConnect.</p> <p>Regarding Android:</p> <p>The app check-in interval does not apply to Android. However, the AppConnect-related policies and settings are updated on the device when the device checks in. Device check-in occurs:</p>	60 minutes

TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
	<ul style="list-style-type: none"> according to the sync interval specified on the device's sync policy. when you force a device check-in from the Devices & Users screen. when the device user uses the Force Device Check-in feature in Mobile@Work on the device. 	
Unauthorized Message	<p>Enter the default message that Mobile@Work displays if the app is not authorized on the device. If you do not enter a default message, the system provides one.</p> <p>NOTE: If AppConnect apps are unauthorized (blocked) due to a security policy violation, the out-of-compliance message is displayed instead of this message.</p>	None
Data Loss Prevention Policies		
Apps without an AppConnect container policy	<p>Select Authorize if you want AppConnect apps to be authorized by default. If you do not select this option, app authorization is determined by the labels applied to the AppConnect container policy and the device.</p> <p>If you select this option, then you can also select:</p> <ul style="list-style-type: none"> the iOS data loss prevention policies the Android screen capture policy 	Not selected
iOS		
Copy/Paste To	<p>iOS only:</p> <p>Select Allow 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.</p> <p>When you select this option, then select either:</p> <ul style="list-style-type: none"> All apps Select All apps if you want the device user to be able to copy content from the AppConnect app and paste it into any other app. AppConnect apps Select AppConnect apps if you want the 	Not selected



TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
	<p>device user to be able to copy content from the AppConnect app and paste it only into other AppConnect apps.</p> <p>Select</p> <p>Related topics</p> <p>Comparison with AppConnect for iOS copy/paste policy</p>	
Print	<p>iOS only:</p> <p>Select Allow 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.</p>	Not selected
Open In	<p>iOS only:</p> <p>Select Allow if you want AppConnect apps to be allowed to use the Open In (document interaction) feature by default. You can override this option in each app's AppConnect container policy.</p> <p>When you select this option, then select either:</p> <ul style="list-style-type: none"> • All apps • AppConnect apps <p>Select AppConnect apps to allow an AppConnect app to send documents to only other AppConnect apps.</p> • Whitelist <p>Select Whitelist if you want the app to be able to send documents only to the apps that you specify.</p> <p>Enter the bundle ID of each app, one per line.</p> <p>For example:</p> <pre>com.myAppCo.myApp1 com.myAppCo.myApp2</pre> <p>The bundle IDs that you enter are case sensitive.</p> <p>Related topics</p>	Not selected

TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
	<ul style="list-style-type: none"> • Open-In data loss prevention policy details • Sharing content from AppConnect for Android apps to non-AppConnect apps . 	
Open From	<p>iOS only:</p> <p>Select Allow if you want AppConnect apps to be allowed to use the Open From (document interaction) feature by default. You can override this option in each app's AppConnect container policy.</p> <p>When you select this option, then select either:</p> <ul style="list-style-type: none"> • All apps Select to allow an AppConnect app to receive documents from any app. • AppConnect apps Select AppConnect apps to allow an AppConnect app to receive documents from only other AppConnect apps. • Whitelist Select Whitelist if you want an AppConnect app to receive documents only from the apps that you specify. Enter the bundle ID of each app, one per line. For example: com.myAppCo.myApp1 com.myAppCo.myApp2 The bundle IDs that you enter are case sensitive. <p>Related topics</p> <ul style="list-style-type: none"> • Open From data loss prevention policy • Sharing content from AppConnect for Android apps to non-AppConnect apps . 	<p>Not selected</p> <p>Enabled if Apps without an AppConnect container policy is enabled.</p>
Drag and Drop	<p>iOS only:</p> <p>Select Allow if you want the device user to be able to drag content from AppConnect apps to other apps. You can override this option in each app's individual</p>	Not selected

TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
	<p>AppConnect container policy.</p> <p>When you select this option, then select either:</p> <ul style="list-style-type: none"> • All apps Select All apps if you want the device user to be able to drag content from the AppConnect app and drop it into any other app. • AppConnect apps Select AppConnect apps if you want the device user to be able to drag content from the AppConnect app and drop it only into other AppConnect apps. 	
Android		
Copy/Paste	<p>Android only:</p> <p>Specify one of the following options:</p> <ul style="list-style-type: none"> • No restrictions The device user can copy and paste between any apps, whether the apps are AppConnect apps or unsecured apps. The device exhibits standard copy/paste behavior. This option is the default. Clipboard use: The device uses the standard Android clipboard for all copy/paste activity. That is, AppConnect apps and unsecured apps all use the same clipboard. • Among AppConnect apps Copy and paste is not possible between AppConnect apps and unsecured apps. The device user can copy and paste among AppConnect apps, and within an AppConnect app. The user can also copy and paste among unsecured apps and within an unsecured app. This option prevents data leaks into or out of the secure container. Clipboard use: AppConnect apps share a clipboard, and unsecured apps share a separate clipboard. • Within an AppConnect app The device user can copy and paste within each AppConnect app. However, the user cannot copy and paste among AppConnect apps, or between AppConnect apps and unsecured apps. The user 	No restrictions



TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
	<p>can also copy and paste among unsecured apps and within an unsecured app. This option is the most restrictive.</p> <p>Clipboard use: Each AppConnect app has its own clipboard. Unsecured apps share one clipboard among all unsecured apps.</p> <p>Related topics</p> <p>Copy/Paste for AppConnect for Android</p>	
Camera	<p>Android only:</p> <p>Select Allow to allow camera photo access for all the AppConnect apps on an Android device.</p> <p>When you select this setting, an AppConnect app can, for example, use a camera app to take a photo with the camera and allow the device user to save the photo.</p> <p>Related topics</p> <p>Interaction with the lockdown policy regarding Android camera access.</p>	Not selected
Gallery	<p>Android only:</p> <p>Select Allow to allow all the AppConnect apps on an Android device to access images from the gallery.</p> <p>When you select this setting, an AppConnect app can, for example, allow a device user to attach images from the gallery to an email.</p>	Not selected
MediaPlayer	<p>Android only:</p> <p>Select Allow to allow all the AppConnect apps to stream media to media players.</p> <p>When you select Allow, AppConnect apps can stream the following file types to media players:</p> <ul style="list-style-type: none"> • MP3 audio files • WAV audio files • MP4 video files <p>The supported file size depends on the Android Secure Apps version as well as the device.</p> <p>Related topics</p>	Not selected

TABLE 3. APPCONNECT GLOBAL POLICY FIELDS (CONT.)

Item	Description	Default Value
	DLP policy for media player access.	
Screen capture	Android only: Select Allow if you want AppConnect apps to allow screen capture by default. You can override this option in each app's AppConnect container policy.	Not selected
Web	Android only: Select Allow 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 , only Web@Work can display the page. Related topics Web DLP policy for browser launching.	Not selected
Non-AppConnect apps can open URLs in Web@Work	Android only: Select Allow 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. Related topics DLP allowing links from non-AppConnect apps to open in Web@Work.	Not selected

Self-service AppConnect passcode recovery

You can allow self-service AppConnect passcode recovery on both iOS and Android devices. Self-service AppConnect passcode recovery allows device users to recover their AppConnect passcode themselves if they forgot it. Device users first enter their MobileIron Core registration credentials before creating a new AppConnect passcode.

Although allowing self-service AppConnect passcode recovery can decrease support calls for passcode assistance and improve the device user experience, your security requirements can be more important to you. Evaluate your priorities for security and device user experience and make the right choice for your enterprise.

However, self-service AppConnect passcode recovery behaves differently on the two platforms, as summarized in the following table:



TABLE 4. SELF-SERVICE APPCONNECT PASSCODE RECOVERY ON iOS VERSUS ANDROID

	iOS	Android
Default value in AppConnect global policy	Self-service AppConnect passcode recovery is allowed .	Self-service AppConnect passcode recovery is not allowed.
What happens when self-service recovery is not allowed?	<p>No method is available to recover a forgotten AppConnect passcode.</p> <p>The device user must delete and reinstall Mobile@Work and each AppConnect app.</p> <p>Therefore, if you do not allow self-service AppConnect passcode recovery, consider increasing the number of failed passcode attempts on the AppConnect global policy.</p>	<p>You can send an Unlock AppConnect Container command from the MobileIron Core Admin Portal. This command removes the AppConnect passcode. The device user then creates a new AppConnect passcode.</p>
Touch ID / Face ID impact	<p>You can choose to use Touch ID or Face ID, when supported on the device, regardless of whether you allow or disable self-service AppConnect passcode recovery.</p> <p>Disabling or allowing AppConnect passcode recovery is not relevant when the device is using Touch ID or Face ID with fallback to device passcode.</p>	Not applicable.



TABLE 4. SELF-SERVICE APPCONNECT PASSCODE RECOVERY ON iOS VERSUS ANDROID (CONT.)

	iOS	Android
Fingerprint impact	Not applicable	<p>You can choose to use fingerprint, when supported on the device, regardless of whether you allow or disable self-service AppConnect passcode recovery.</p> <p>When the device user creates a new AppConnect passcode, the user must again choose whether to enable fingerprint login.</p>
Maximum number of failed attempts on the AppConnect global policy	<p>If you allow self-service AppConnect passcode recovery, the device user can create a new AppConnect passcode when reaching the maximum number of failed attempts.</p> <p>If you do not allow self-service AppConnect passcode recovery, no method is available to recover a forgotten AppConnect passcode. Therefore, consider increasing the maximum number of failed passcode attempts.</p>	<p>If you allow self-service AppConnect passcode recovery, it is not available when the device user reaches the maximum number of failed attempts.</p> <p>You must send an Unlock AppConnect Container command from the MobileIron Core Admin Portal to allow the device user to create a new AppConnect passcode and access AppConnect apps.</p> <p>If you do not allow self-service AppConnect passcode recovery, you also must send an Unlock AppConnect Container command when the device user reaches the maximum number of failed attempts.</p>
Password history on the AppConnect global policy	When a device user creates a new AppConnect passcode through self-service recovery, the passcode history rule is enforced.	When a device user creates a new AppConnect passcode, whether due to an Unlock AppConnect Container command or through self-service recovery, the passcode history rule is enforced.

AppConnect passcode strength

You can set the desired AppConnect passcode strength to enforce how strong an AppConnect passcode must be. Setting the AppConnect passcode strength prevents device users from using AppConnect passcodes that are weak and therefore easy to guess. However, setting the AppConnect passcode strength too high makes using AppConnect apps inconvenient for the device user because they have to enter a more complicated or longer AppConnect passcode. Therefore, when you choose the AppConnect passcode strength requirement, consider both your security needs and your device user convenience.

Note The Following:



- Enabling, disabling, or changing the passcode strength requires the device user to reset the AppConnect passcode.
- The AppConnect passcode strength setting has no impact on the device passcode. Even if the device users are using [Touch ID or Face ID with fallback to device passcode](#) to access AppConnect apps, the device passcode is not impacted by the AppConnect passcode strength.
- On iOS devices running Mobile@Work prior to version 9.7.0, the AppConnect Passcode Type on the AppConnect global policy must be either Alphanumeric or Don't Specify. Password strength is not supported on these iOS devices when the passcode type is Numeric. On Android devices, password strength is supported with all AppConnect passcode types.

To set the AppConnect passcode strength, choose a value between 0 and 100 as follows:

TABLE 5. APPCONNECT PASSCODE STRENGTH VALUES

Strength value	Description	Examples
0 - 20	Weak: risky password	<ul style="list-style-type: none"> • Few characters: zxcvbn • Sequences: abcdefghijk987654321 • Names: briansmith4mayor • Words: viking • Words with number substitutions: ScoRpi0ns
21 - 40	Fair: protection from throttled online attacks Throttled online attacks are attacks to guess the passcode which are: <ul style="list-style-type: none"> • on the device • rate-limited Rate-limited attacks are limited to some number of attempts per time period.	<ul style="list-style-type: none"> • Few characters but with special characters: qwER43@! • Words plus numbers: temppass22 • Names plus numbers: ryanhunter2000 • Words with special character and number substitutions: R0\$38uD99 • Names with capitalization: verlineVANDERMARK
41 - 60	Good: protection from unthrottled online attacks Unthrottled online attacks are attacks to guess the passcode which are: <ul style="list-style-type: none"> • on the device • not rate-limited 	<ul style="list-style-type: none"> • Longer words with special character and number substitutions: Tr0ub4dour&3 • Longer phrases with numbers and special characters: neverforget13/3/1997 • Longer letter, number, and special character combinations: asdfghju7654rewqOEUIDHG&*()LS_
61 - 80	Strong: moderate protection from offline slow-hash scenario	<ul style="list-style-type: none"> • Longer random letters and numbers: zevusr3



TABLE 5. APPCONNECT PASSCODE STRENGTH VALUES (CONT.)

Strength value	Description	Examples
	An offline slow-hash scenario is a sophisticated algorithm for guessing a passcode. The algorithm runs offline from the device after copying passcode-related files from the device.	esqu3Wil tgbvdnjuk <ul style="list-style-type: none"> Longer phrases with numbers and special characters: Compl3xChar\$
81 - 100	Very strong: strong protection from offline slow-hash scenario	<ul style="list-style-type: none"> Very long random characters: eheuczkqyq rWibMFACxAUGZmxhVncy Ba9ZyWABu99 [BK#6MBgbH88Tofv)vs\$w Long phrases: correcthorsebatterystaple Long phrases with substitutions: coRrecth0rseba++ery9.23.2007staple\$

Mechanism to force all device users to change their AppConnect passcodes

Device users are prompted to change their AppConnect passcodes when you change any of the following settings on the AppConnect global policy:

- AppConnect passcode type
- AppConnect passcode length
- AppConnect passcode strength settings

The device users must change their AppConnect passcodes regardless whether their passcode already meets the new requirements.

With this mechanism, you have a way to force all devices users to change their AppConnect passcode. This capability is useful if, for example:

- Your security requirements change, and you want to require a more complex passcode, such as a longer passcode.
- You are concerned that some users' AppConnect passcodes have been compromised, but you do not know exactly which users.

Interaction with the lockdown policy regarding Android camera access

The lockdown policy for the device has an option to enable or disable the camera. The lockdown policy applies to all apps on the device, not just AppConnect apps. The interactions between the lockdown policy and the AppConnect global policy are:

- If the lockdown policy prohibits camera use, AppConnect apps cannot use the camera. Camera use is prohibited even if you allow camera access on the AppConnect global policy.
- If the lockdown policy allows camera use, AppConnect apps can access photos from the camera only if you allow camera access on the AppConnect global policy.

The following table summarizes this interaction of the lockdown policy and the AppConnect global policy:

TABLE 6. CAMERA OPTIONS INTERACTIONS IN LOCKDOWN POLICY AND APPCONNECT GLOBAL POLICY

	AppConnect global policy: Camera access allowed	AppConnect global policy: Camera access prohibited
Lockdown policy: Camera enabled	AppConnect apps can use the camera.	AppConnect apps cannot use the camera.
Lockdown policy: Camera disabled	AppConnect apps cannot use the camera.	AppConnect apps cannot use the camera.

AppConnect container policies

The AppConnect container policy:

- authorizes an AppConnect app.
- specifies the data loss prevention settings for an AppConnect app.
- can be automatically created by MobileIron Core.

NOTE: For each AppConnect app, make sure only one AppConnect container policy applies to each device.

AppConnect app authorization

Each AppConnect app requires an AppConnect container policy. The presence of an AppConnect container policy for a device is what authorizes the app on the device. You apply a label to the AppConnect container policy to apply it to a device.

If you later remove the AppConnect container policy, or remove the device's label from the policy:

- an iOS AppConnect app becomes *retired*. A retired app becomes unauthorized on the device and the app deletes (wipes) all its sensitive data.
- an Android AppConnect app becomes unauthorized. If the app is unauthorized, when the device user tries to run it, the Secure Apps Manager displays a message that the app is unauthorized.



Related topics

- [Situations that wipe AppConnect for iOS app data](#)
- [Situations that wipe Android AppConnect app data](#)

Data loss prevention settings

In the AppConnect container policy, you also configure data loss prevention (DLP) settings. Specifically, you configure whether you want the app to be allowed to use these features:

- Copy / paste (iOS only)
- Print (iOS only)
- Open In (document interaction) (iOS only)
- Open From (document interaction) (iOS only)
- Drag and Drop (iOS only)
- Screen capture (Android only)

An app's AppConnect container policy overrides the corresponding settings on the AppConnect global policy.

Automatically created AppConnect container policies

When you upload an AppConnect app to MobileIron Core's App Catalog, Core automatically creates an AppConnect container policy as follows:

- For Android AppConnect apps:
MobileIron Core always takes this automatic action. If the app has specified DLP settings, Core uses those settings. Otherwise, Core creates an AppConnect container policy with all the values set to not allowed.
- For iOS AppConnect apps built with the AppConnect for iOS SDK or Cordova Plugin:
Core takes this automatic action only if an in-house app has specified its desired default values for the policy in its IPA file. This automatic action does not occur when you specify an Apple App Store AppConnect app as a recommended app.
- For wrapped iOS AppConnect apps:
Core always takes this automatic action, setting all the DLP values to not allowed.

The name of the AppConnect container policy is:

TABLE 7. NAME OF AUTOMATICALLY-CREATED APPCONNECT CONTAINER POLICY

For iOS AppConnect apps	Default <bundle ID of app> Container Policy
For Android AppConnect apps	Default <package ID of app> Container Policy

NOTE: In the Admin Portal, on **Policies & Configs > Configurations**, the name of the app, not the name of the AppConnect container policy, displays in the name column.



You can override these DLP values by editing the app's AppConnect container policy. MobileIron Core keeps in sync the labels that you apply to the app and the labels that you apply to the AppConnect container policy that Core automatically created.

Configuring AppConnect container policies

The following describes the steps to configure an AppConnect container policy.

Procedure

1. In the Admin Portal, select **Policy & Configs > Configurations**.
2. Select the existing container policy for the app, or select **Add New > AppConnect > Container Policy** to create a new one.

FIGURE 1. APPCONNECT CONTAINER POLICY

New AppConnect Container Policy

An app is authorized only if an AppConnect container policy for the app is present on the device. This policy also allows you to define app-specific settings.

Name

Description

Application

☐ Exempt from AppConnect passcode policy

SECURITY POLICIES

▼ iOS Data Loss Prevention

- ☐ Allow Print
- ☐ Allow Copy/Paste To
- ☐ Allow Open In
- ☒ Allow Open From
 - ☒ All apps
 - ☐ AppConnect apps
 - ☐ Whitelist
- ☐ Allow Drag and Drop

▼ Android Data Loss Prevention

- ☐ Allow Screen Capture

Cancel Save

3. Enter the requested information.
4. Click **Save**.
5. Select the new app policy.
6. Select **More Actions > Apply To Label**.
7. Select the labels to which you want to apply this AppConnect container policy.
8. Click **Apply**.

Be sure to apply one of the labels that you selected to the device. To check the device's labels:



1. Go to **Devices & Users > Devices**.
2. Expand the device details panel by clicking the up arrow for the desired device.
3. In the Device Details panel, select **Label Membership**.

Related topics

For a description of the fields in the AppConnect container policy, see [AppConnect container policy field description](#).

AppConnect container policy field description

Use the following guidelines to create or edit an AppConnect container policy:

TABLE 8. APPCONNECT CONTAINER POLICY FIELDS

Item	Description
Name	<p>Enter brief text that identifies this AppConnect container policy.</p> <p>NOTE: If MobileIron Core automatically created this policy:</p> <ul style="list-style-type: none"> • You cannot edit the name. • The name is not the same as the name that appears in the name column in Policy & Configs > Configurations.
Description	Enter additional text that clarifies the purpose of this AppConnect container policy.
Application	<p>Android:</p> <p>Select an Android AppConnect app from the MobileIron Core App Catalog.</p> <p>iOS:</p> <p>Select an iOS AppConnect app from the MobileIron Core App Catalog or enter the bundle ID of an iOS AppConnect app. A bundle ID is case sensitive.</p> <p>NOTE: The dropdown selection includes an iOS AppConnect app only if both of the following statements are true:</p> <ul style="list-style-type: none"> • The app was added to the Core App Catalog as an in-house app. • The app specifies default data loss prevention policy settings (copy/paste, document interaction, print).
Exempt from AppConnect passcode policy	<p>iOS only:</p> <p>Select this option if you want to allow the device user to use the app without entering the AppConnect passcode or Touch ID / Face ID.</p> <p>NOTE: When you select this option, situations still occur when the device user must enter the AppConnect passcode. For example, if the user launches an AppConnect app that is not already running, the user is prompted to enter the AppConnect passcode.</p>
iOS Data Loss Prevention	



TABLE 8. APPCONNECT CONTAINER POLICY FIELDS (CONT.)

Item	Description
Allow Print	<p>iOS only:</p> <p>Select Allow Print if you want AppConnect apps to be allowed to use print capabilities.</p>
Allow Copy/Paste To	<p>iOS only:</p> <p>Select Allow Copy/Paste To if you want the device user to be able to copy content from the AppConnect app to other apps.</p> <p>When you select this option, then select either:</p> <ul style="list-style-type: none"> • All apps Select All apps if you want the device user to be able to copy content from the AppConnect app and paste it into any other app. • 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. <p>Related topics</p> <p>Comparison with AppConnect for iOS copy/paste policy</p>
Allow Open In	<p>iOS only:</p> <p>Select Allow Open In if you want AppConnect apps to be allowed to use the Open In (document interaction) feature.</p> <p>When you select this option, then select either:</p> <ul style="list-style-type: none"> • All apps Select All apps if you want the app to be able to send documents to any other app. • AppConnect apps Select AppConnect apps to allow an AppConnect app to send documents to only other AppConnect apps. • Whitelist Select Whitelist if you want the app to be able to send documents only to the apps that you specify. Enter the bundle ID of each app, one per line, or in a semi-colon delimited list. For example: com.myAppCo.myApp1 com.myAppCo.myApp2;com.myAppCo.myApp3 The bundle IDs that you enter are case sensitive. <p>Related topics</p> <ul style="list-style-type: none"> • Open-In data loss prevention policy details • Sharing content from AppConnect for Android apps to non-AppConnect apps
Allow Open From	<p>iOS only:</p>

TABLE 8. APPCONNECT CONTAINER POLICY FIELDS (CONT.)

Item	Description
	<p>Enabled by default.</p> <p>Select Allow Open From if you want AppConnect apps to be allowed to use the Open From (document interaction) feature by default. You can override this option in each app's AppConnect container policy.</p> <p>When you select this option, then select either:</p> <ul style="list-style-type: none"> • All apps Select to allow an AppConnect app to receive documents from any app. • AppConnect apps Select AppConnect apps to allow an AppConnect app to receive documents from only other AppConnect apps. • Whitelist Select Whitelist if you want an AppConnect app to receive documents only from the apps that you specify. Enter the bundle ID of each app, one per line, or in a semi-colon delimited list. For example: com.myAppCo.myApp1 com.myAppCo.myApp2;com.myAppCo.myApp3 The bundle IDs that you enter are case sensitive. <p>Related topics</p> <ul style="list-style-type: none"> • Open From data loss prevention policy • Sharing content from AppConnect for Android apps to non-AppConnect apps
Allow Drag and Drop	<p>iOS only:</p> <p>Select Allow Drag and Drop if you want the device user to be able to drag content from the AppConnect app to other apps.</p> <p>When you select this option, then select either:</p> <ul style="list-style-type: none"> • All apps Select All apps if you want the device user to be able to drag content from the AppConnect app and drop it into any other app. • AppConnect apps Select AppConnect apps if you want the device user to be able to drag content from the AppConnect app and drop it only into other AppConnect apps.
Android Data Loss Prevention	
Allow Screen Capture	<p>Android only:</p> <p>Select Allow Screen Capture if you want the app to allow screen capture.</p>



Enabling secure apps

If you are deploying secure apps developed by MobileIron, you need to enable those products. If you are deploying secure apps developed by your organization or a third party, you need to enable the additional product that supports those apps.

1. In the Admin Portal, go to **Settings > System Settings > Additional Products > Licensed Products**.
2. Select the option for each product, as specified in the following tables.
3. Click **Save**.

Enabling licensing options for Android secure apps

The following table shows which Android secure apps you can deploy for each option. Select each option only if your organization has purchased it.

TABLE 9. ANDROID SECURE APPS YOU CAN DEPLOY WITH EACH LICENSING OPTION

If you enable Docs@Work, you can deploy:	If you enable AppConnect for third-party and in-house apps, you can deploy:	If you enable Web@Work, you can deploy:
<ul style="list-style-type: none"> • Docs@Work • Secure Android Email+ • TouchDown for SmartPhones* • Web@Work 	<ul style="list-style-type: none"> • Secure Android Email+ • TouchDown for SmartPhones* • FileManager • Other third-party AppConnect apps* • In-house AppConnect apps 	<ul style="list-style-type: none"> • Web@Work

*In addition to purchasing the Additional Products option, these apps have an additional cost.

Enabling licensing options for iOS secure apps

The following table shows which iOS secure apps you can deploy for each option. Select each option only if your organization has purchased it.

TABLE 10. IOS SECURE APPS YOU CAN DEPLOY WITH EACH LICENSING OPTION

If you enable Docs@Work, you can deploy:	If you enable AppConnect for third-party and in-house apps, you can deploy:	If you enable Web@Work, you can deploy:
<ul style="list-style-type: none"> • Docs@Work • Web@Work 	<ul style="list-style-type: none"> • Email+ for iOS* • Insight* • Other third-party AppConnect apps* • In-house AppConnect apps 	<ul style="list-style-type: none"> • Web@Work



*In addition to purchasing the Additional Products option, these apps have an additional cost.

Enabling AppTunnel

If you are deploying secure apps developed by your organization or a third party, you need to enable an additional product to secure data-in-motion with these apps. Enable this option only if your organization has purchased AppTunnel.

1. In the Admin Portal, go to **Settings > System Settings > Additional Products > Licensed Products**.
2. Select **AppConnect for Third-party and In-house Apps**.
3. Select **AppTunnel for Third-party and In-house Apps**.
4. Click **Save**.

Note The Following:

- Do **not** select this option if you are using AppTunnel *only* for Docs@Work or Web@Work.
- The data-in-motion for AppConnect-enabled ActiveSync email apps such as TouchDown for SmartPhones, Email+ for Android, and Email+ for iOS is secure without using AppTunnel. However, if you are using AppTunnel for MobileIron cloud notification service (CNS) for Email+ for iOS, select **AppTunnel for Third-party and In-house Apps**.

Related topics

MobileIron Email+ for iOS Guide for Administrators

Configuring an AppTunnel service

An AppTunnel service defines the backend service to which an AppConnect AppTunnel is created.

You create the AppTunnel service on the MobileIron Core Admin Portal in **Services > Sentry**. Edit the Standalone Sentry entry that is configured for AppTunnel to add the AppTunnel service.

See "Standalone Sentry for AppTunnel" in the *MobileIron Sentry Guide for MobileIron Core* for information about configuring AppTunnel and an AppTunnel service. Standalone Sentry product documentation is available on the [Standalone Sentry Product Documentation Home Page](#).

About the AppTunnel service name

When you configure an AppTunnel service, you give the service a service name. The service name is used in the AppConnect app configuration. The app configuration uses the service name to restrict the app to accessing servers in the **Server List** field associated with the service name. The service name is similarly used in:



- the Web@Work setting for configuring tunneling for Web@Work for Android or iOS
- the Docs@Work setting for configuring tunneling for Docs@Work

The service name is one of the following:

- A unique name for the service that the AppConnect app on the device accesses

One or more of your internal app servers provide the service. You list the servers in the **Server List** field associated with the service name.

For example, some possible service names are:

- SharePoint
- Human Resources

A service name cannot contain these characters: 'space' \ ; * ? < > " |.

Special prefixes:

- For app tunnels that point to CIFS-based content servers, the service name must begin with **CIFS_**.
- For AppTunnel with TCP tunneling, the name must begin with **TCP** (case-insensitive).

Example: **TCP_Finance**

- **<ANY>**

Select **<ANY>** for the service name to allow tunneling to any URL that the app requests. Typically, you select **<ANY>** if an AppConnect app's app configuration specifies a URL with wildcards for tunneling, such as *.myCompany.com. The Sentry tunnels the data for any URL request that the app makes that matches the URL with wildcards.

The Sentry tunnels the data to the app server that has the URL that the app specified. The **Server List** field is therefore not applicable when the Service Name is **<ANY>**.

For example, consider when the app requests URL myAppServer.mycompany.com, which matches *.mycompany.com in the app configuration. The Sentry tunnels the data to myAppServer.myCompany.com

Web@Work typically uses the **<ANY>** service, so that it can browse to any of your internal servers.

NOTE: Do not select this option for tunneling to CIFS-based content servers. Select **<CIFS_ANY>** instead.

- **<TCP_ANY>**

Select **<TCP_ANY>** for the service name to allow AppTunnel with TCP tunneling to any backend server that the app requests.

- **<CIFS_ANY>**

Select **<CIFS_ANY>** or the service name to allow tunneling to any URL for a CIFS-based content server. Typically, you select **<CIFS_ANY>** if the URL for a CIFS-based content server contains wildcards for tunneling, such as *.myCompany.com.

AppConnect app configuration

An AppConnect app configuration specifies:

- app-specific configuration for the app.
- AppTunnel settings for the app.

IMPORTANT: For each AppConnect app, make sure only one AppConnect app configuration applies to each device.

The following describe how to configure an AppConnect app configuration:

- [Automatically created AppConnect app configuration](#)
- [Automatically provided key-value pairs](#)
- [Configuring an AppConnect app configuration](#)
- [AppConnect app configuration field description](#)

Automatically created AppConnect app configuration

When you upload an AppConnect app to the MobileIron Core App Catalog, Core creates an AppConnect app configuration automatically as follows:

- For Android AppConnect apps:
Core always takes this automatic action. If the app has specified configuration requirements, Core uses that configuration. Otherwise, Core creates an AppConnect app configuration with no configuration values.
- For iOS AppConnect apps built using the AppConnect for iOS SDK or Cordova Plugin:
Core takes this automatic action only if an in-house app has specified configuration requirements in its IPA file. This automatic action does not occur when you specify an Apple App Store AppConnect app as a recommended app.
- For wrapped iOS AppConnect apps:
Core does not take this automatic action.

The following table lists the name of the automatically created AppConnect app configuration.

TABLE 11. NAME OF AUTOMATICALLY-CREATED APPCONNECT APP CONFIGURATION

OS of the AppConnect app	Name of automatically-created AppConnect app configuration
For iOS AppConnect apps	Default <bundle ID of app> Configuration
For Android AppConnect apps	Default <package ID of app> Configuration

NOTE: In the Admin Portal, on **Policies & Configs > Configurations**, the name of the app, not the name of the AppConnect app configuration, displays in the name column.



Automatically provided key-value pairs

MobileIron Core takes a special action for some iOS AppConnect apps in the Apple App Store that you specify as recommended apps. The special action occurs when you enter the bundle ID of one of these apps in the Application field of an app configuration and then save the app configuration. Core automatically populates the key-value pairs for the recommended app. Core does not overwrite any key-value pairs that you manually added. You can then edit the app configuration to change the provided key-value pairs, if necessary.

Configuring an AppConnect app configuration

If an AppConnect app configuration is not automatically created, create the configuration on the Core Admin Portal.

Procedure

1. In the Admin Portal, select **Policy & Configs > Configurations**.
2. Select **Add New > AppConnect > App Configuration** to create an AppConnect app configuration.
3. Update the form as needed.
4. Click **Save**.
5. Select the new AppConnect app configuration.
6. Select **More Actions > Apply To Label**.
7. Select the labels to which you want to apply this AppConnect app configuration.
8. Click **Apply**.

IMPORTANT: Be sure to apply one of the labels that you selected to the device.

Related topics

- [AppConnect app configuration field description](#)
- [Checking the device's labels](#)
- [Adding a device to a label](#)

Checking the device's labels

The following describes how to check a device's labels.

Procedure

1. Go to **Devices & Users > Devices**.
2. Select the device.
3. In the Device Details Pane, select **Label Membership**.

Adding a device to a label

The following describes how to add a device to a label.



Procedure

1. Go to **Devices & Users > Devices**.
2. Select the device.
3. Select **More Actions > Apply To Label**.
4. Select the labels to apply to the device.
5. Click **Apply**.

AppConnect app configuration field description

Use the following guidelines to create or edit an AppConnect app configuration.

TABLE 12. APPCONNECT APP CONFIGURATION FIELDS

Item	Description
Name	<p>Enter brief text that identifies this AppConnect app configuration.</p> <p>NOTE: If MobileIron Core automatically created this AppConnect app configuration:</p> <ul style="list-style-type: none"> • You cannot edit the name. • The name is not the same as the name that appears in the name column in Policy & Configs > Configurations.
Description	Enter additional text that clarifies the purpose of this AppConnect app configuration.
Application	<p>Android:</p> <p>Select an Android AppConnect app from the MobileIron Core App Catalog.</p> <p>iOS:</p> <p>Select an iOS AppConnect app from the MobileIron Core App Catalog or enter the bundle ID of an iOS AppConnect app. A bundle ID that you enter is case sensitive.</p> <p>NOTE: The dropdown selection includes an iOS AppConnect app only if both of the following statements are true:</p> <ul style="list-style-type: none"> • The app was added to the Core App Catalog as an in-house app. • The app specifies default app-specific configurations.
Client TLS	<p>If the app is using certificate pinning, select Enable Client TLS Configuration and choose the appropriate Client TLS configuration from the dropdown.</p> <p>Related topics</p>



TABLE 12. APPCONNECT APP CONFIGURATION FIELDS (CONT.)

Item	Description
	Certificate pinning for AppConnect apps
AppTunnel Rules	<p>Configure AppTunnel rules settings for this app.</p> <p>First, configure the Standalone Sentry to support AppTunnel. See Configuring AppConnect and AppTunnel.</p> <p>When the app tries to connect to the URL and port configured here, the Sentry creates a tunnel to the app server.</p> <p>NOTE: This section is not available when the AppConnect app configuration is for the Secure Apps Manager. The Secure Apps Manager is the app required for Android devices running AppConnect apps. AppTunnel configuration is not applicable to the Secure Apps Manager.</p>
Enable MobileIron Access	<p>The setting is available only if MobileIron Access is configured in the Admin Portal in Services > Access. Otherwise, the setting is grayed out.</p> <p>If the option is selected, MobileIron Access trusts the HTTPS traffic via AppTunnel. Tunnel is not needed in this setup.</p> <p>For information about MobileIron Access and how to set up the service with MobileIron Core, see the <i>MobileIron Access Guide</i>.</p> <p>NOTE: If Enable Split Tunneling using MobileIron Tunnel is selected, HTTPS authentication traffic, which would have previously used AppTunnel to Access, goes through Tunnel instead.</p>
Enable Split Tunneling using MobileIron Tunnel	<p>iOS only. Requires Mobile@Work 12.3.0 and MobileIron Tunnel 4.1.0 for iOS.</p> <p>Before enabling the option, ensure that MobileIron Tunnel is deployed and a Tunnel VPN configuration is applied to the AppConnect app. For information about deploying MobileIron Tunnel for iOS, see the <i>MobileIron Tunnel for iOS Guide for Administrators</i>.</p> <p>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:</p> <ul style="list-style-type: none"> • AppTunnel rules are configured to tunnel app data. • Enable MobileIron Access is selected. <p>Enabling the option allows the configured AppTunnel rules to be managed through MobileIron Tunnel rather than through AppTunnel</p>



TABLE 12. APPCONNECT APP CONFIGURATION FIELDS (CONT.)

Item	Description
	<p>For information about the UIWebView API deprecation, see UIWebView Deprecation and AppConnect Compatibility.</p> <p>NOTE: Rules configured in the Tunnel VPN configuration impact whether app data to the enterprise resource is tunneled. Consider the following case:</p> <ul style="list-style-type: none"> 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. <p>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.</p>
<p>To add an AppTunnel rule, click Add+ .</p> <p>To delete an AppTunnel rule, click the X at the end of the row.</p>	
Sentry	Select a Sentry configured for AppTunnel from the drop-down list.
Service	<p>Select a service name from the drop-down list.</p> <p>This service name specifies an AppTunnel service configured in the AppTunnel Configuration section of the specified Sentry.</p> <p>NOTE: If you entered a URL with wildcards in the URL Wildcard field, you can only select <ANY> or <CIFS_ANY> as the service. The <ANY> or <CIFS_ANY> service must be configured in the AppTunnel Configuration section of the Sentry configured for AppTunnel.</p> <p>If the service on the Sentry is configured with its Server Auth set to Kerberos, the AppConnect app uses Single Sign On. That is, the device user does not enter any further credentials when the app accesses its enterprise app server.</p>
URL Wildcard	<p>Enter one of the following:</p> <ul style="list-style-type: none"> an app server's hostname Example: finance.yourcompany.com a hostname with wildcards. The wildcard character is *. Example: *.yourcompanyname.com <p>If the app requests to access this hostname, the Sentry tunnels the app data to an app server. The Sentry and Service fields that you specify in</p>



TABLE 12. APPCONNECT APP CONFIGURATION FIELDS (CONT.)

Item	Description
	<p>this AppTunnel Rule row determine the target app server.</p> <p>Note The Following:</p> <ul style="list-style-type: none"> The app data is tunneled only if the app's request matches this hostname and the port number specified in the Port field of this AppTunnel row. Exception: For iOS apps using AppConnect releases prior to AppConnect for iOS SDK 2.5 and AppConnect for iOS Wrapper 2.7, only the hostname, not the port number determines whether the app data is tunneled. A hostname with wildcards works only with the service <ANY>, <TCP_ANY>, or <CIFS_ANY>. Unlike services with specific service names, these services do not have associated app servers. The Sentry tunnels the data to the app server that has the URL that the app specified. The order of these AppTunnel Rule rows matters. If you specify more than one AppTunnel Rule row, the first row that matches the hostname (and port, for Android) that the app requested is chosen. That row determines the Sentry and Service to use for tunneling. Do not include a URI scheme, such as http:// or https://, in this field.
Port	<p>Enter the port number that the app requests to access.</p> <p>The app data is tunneled only if the app's request matches the hostname in the URL Wildcard field and this port number.</p> <p>Exception: For iOS apps using AppConnect releases prior to AppConnect for iOS SDK 2.5 and AppConnect for iOS Wrapper 2.7, only the hostname, not the port number determines whether the app data is tunneled.</p> <p>Note The Following:</p> <ul style="list-style-type: none"> If you do not enter a port number, the port in the app's request is not used to determine whether data is tunneled. Entering a port number in this field is required when both of the following are true: <ul style="list-style-type: none"> The hostname in the URL Wildcard field does not contain a wildcard. The service is not <ANY> or <CIFS_ANY>.
Identity Certificate	Select the Certificate Enrollment setting that you created for AppTunnel.

TABLE 12. APPCONNECT APP CONFIGURATION FIELDS (CONT.)

Item	Description
	<p>This selection determines the certificate that the device presents to the Standalone Sentry for authentication.</p> <p>Related topics</p> <p>“Device and server authentication” in the MobileIron Sentry Guide</p>
Configurations	<p>Specify app-specific configuration settings as key-value pairs.</p> <p>To add a key-value pair, click Add+ .</p> <p>To delete a key-value pair, click the X at the end of the row.</p>
Key	<p>Enter the key. The key is any string that the app recognizes as a configurable item.</p> <p>For example: userid, appURL</p>
Value	<p>Enter the value. The value is either:</p> <ul style="list-style-type: none"> a string <p>Example</p> <p>\$USERID\$ https://someEnterpriseURL.com</p> <p>The string can have any value that is meaningful to the app. It can also include one or more of these MobileIron Core variables:</p> <p>\$USERID\$, \$PASSWORD\$, \$EMAIL\$, \$USER_CUSTOM1\$, \$USER_CUSTOM2\$, \$USER_CUSTOM3\$, \$USER_CUSTOM4\$, \$GOOGLE_AUTOGEN_PASSWORD\$, \$FIRST_NAME\$, \$LAST_NAME\$, \$DISPLAY_NAME\$, \$DEVICE_CLIENT_ID\$, \$DEVICE_ID\$, \$DEVICE_IMEI\$, \$DEVICE_IMSI\$, \$DEVICE_MAC\$, \$DEVICE_SN\$, \$DEVICE_UDID\$, \$DEVICE_UUID\$, \$DEVICE_UUID_NO_DASHES\$, \$MI_APPSTORE_URL\$, \$RANDOM_16\$, \$RANDOM_32\$, \$RANDOM_64\$, \$REALM\$, \$TIMESTAMP_MS\$, \$USER_DN\$, \$USER_LOCALE\$, \$USER_UPN\$</p> <p>Custom attribute variables are also supported:</p> <p>\$CUSTOM_DEVICE_<attribute name>\$ \$CUSTOM_USER_<attribute name>\$</p> <p>If you do not want to provide a value, enter \$NULL\$. The \$NULL\$ value tells the app that the app user will need to provide the value.</p> <p>If you specify \$PASSWORD\$, also enable Save User Password under Settings > System Settings > Users & Devices > Registration. However, only devices that register <i>after</i> you enable Save User Password will receive the password.</p> <ul style="list-style-type: none"> a Certificate Enrollment or Certificate setting

TABLE 12. APPCONNECT APP CONFIGURATION FIELDS (CONT.)

Item	Description
	<p>NOTE: For client-provided certificate enrollment settings, Mobile@Work for iOS or Secure Apps Manager for Android, not Core, provides the certificate to the app.</p> <p>Certificate Enrollment and Certificate settings that you configured in Policy & Configs > Configurations appear in the dropdown list. When you choose a Certificate Enrollment or Certificate setting, MobileIron Core sends the contents of the certificate as the value. If the certificate is password-encoded, Core automatically sends another key-value pair. The key's name is the string <i><name of key for certificate>_MI_CERT_PW</i>. The value is the certificate's password.</p>

Configuring the Open With Secure Email App option

When you use AppConnect for Android, device users can use a secure email app. In this case, you typically configure Standalone Sentry to deliver email attachments to these Android devices, because the attachments remain in the secure container.

When you use AppConnect for iOS, you can use the native iOS email client or a third-party AppConnect-enabled email client. With the native iOS email client, you can configure Standalone Sentry so that attachments open only with the Docs@Work app. With third-party AppConnect-enabled email clients, you can configure Standalone Sentry to deliver the emails with attachments to the secure client. In both cases, the attachments can then only be shared with other apps according to your data loss prevention policies.

Therefore, when using secure email clients, you typically configure Standalone Sentry to use the email attachment control setting called Open With Secure Email App.

Procedure

1. In the Admin Portal, go to **Services > Sentry**.
2. Select a Standalone Sentry for which you have enabled ActiveSync.
3. Click the Edit icon.
4. Verify that **Enable ActiveSync** is selected.
5. In the **ActiveSync Configuration** section, in the **Attachment Control Configuration** section, select **Enable Attachment Control**.
6. For **iOS And Android Using Secure Email Apps**, select **Open With Secure Email App**.
7. Click **Save**.



Configuring compliance actions

The security policy that is applied to a device determines what situations make a device non-compliant. For each situation, the security policy specifies a compliance action. These actions can be either default compliance actions or custom compliance actions.

Some compliance actions impact AppConnect apps as follows:

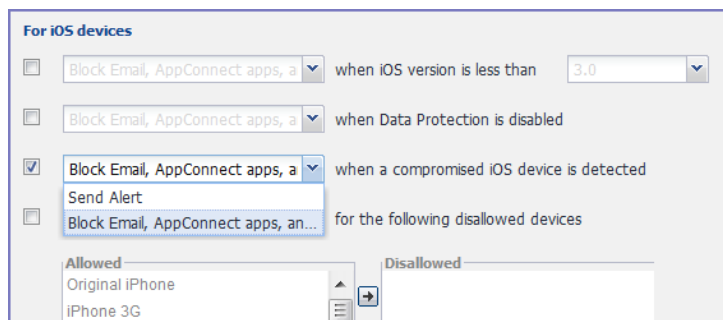
- Immediately block access to the web sites configured to use the AppTunnel feature.
- Unauthorize AppConnect apps.
- Delete (wipe) the secure data of AppConnect apps.

For details about compliance actions that impact AppConnect apps, see “Compliance actions policy violations” in the *MobileIron Core Device Management Guide for iOS and macOS Devices* or the *MobileIron Core Device Management Guide for Android and Android enterprise Devices*.

Procedure

To specify a compliance action:

1. Go to **Policies & Configs > Policies** on the Admin Portal.
2. Select a security policy.
3. Click Edit.
4. Select an access control setting in the **Access Control** section.
For example, select **when a compromised iOS device is detected**.



The screenshot shows the 'For iOS devices' section of the MobileIron Admin Portal. It contains a list of compliance actions with checkboxes and dropdown menus. The third action, 'when a compromised iOS device is detected', is selected with a checkmark. A dropdown menu is open for this action, showing 'Block Email, AppConnect apps, a...' and 'Send Alert'. Below the list, there are two sections: 'Allowed' and 'Disallowed'. The 'Allowed' section lists 'Original iPhone' and 'iPhone 3G'. The 'Disallowed' section is empty.

5. Select a default or custom compliance action from the dropdown list.
6. Click **Save**.

Quick start configuration for AppConnect for Android

Complete these tasks to quickly set up AppConnect for Android on MobileIron Core. However, for more information about each task, including details about all the settings, see the provided references and also details in [Using AppConnect for Android](#).

Before you begin

1. Obtain the AppConnect Apps for Android that MobileIron provides.
Download the Secure Apps Manager from:
<https://support.mobileiron.com/mi/android-sam/current/>
Download Secure Android Email+, Web@ Work, Docs@Work, FileManager, and TouchDown for Smartphones, from:
<https://support.mobileiron.com/support/CDL.html>
2. Obtain in-house and third-party AppConnect apps.
3. Put all the APK files where they are available for upload to MobileIron Core.

Procedure

1. [Uploading the Secure Apps Manager to Core for Android AppConnect quick start.](#)
2. [Uploading the AppConnect apps to Core for Android AppConnect quick start](#)
3. [Enabling Core licensing options for Android AppConnect quick start.](#)
4. [Configuring the AppConnect global policy for Android AppConnect quick start.](#)
5. [Configuring the AppConnect container policy for Android AppConnect quick start.](#)
6. [Configuring settings specific to the app for Android AppConnect quick start.](#)
7. [Configuring email attachment control for Android AppConnect quick start.](#)

Uploading the Secure Apps Manager to Core for Android AppConnect quick start

You upload the Secure Apps Manager to MobileIron Core in the same manner you add any in-house app. After uploading the Secure Apps Manager, you can distribute it to devices by applying the app to labels that contain the appropriate devices.

Procedure

1. In the MobileIron Core Admin Portal, go to **Apps > App Catalog > Add+ > In-House**.
2. Click **Browse** and browse to the Secure Apps Manager.
3. Click **Next**.



4. Optionally make selections, clicking **Next**, and then **Finish**.
5. Select the Secure Apps Manager entry on the **Apps > Apps Catalog** screen.
6. Click **Actions > Apply to Labels**.
7. Select the appropriate labels and click **Apply**.

Next steps

Continue to [Uploading the AppConnect apps to Core for Android AppConnect quick start](#).

Related topics

- “Managing Mobile Apps for Android” in the Apps@Work Guide
- [The MobileIron client app, the Secure Apps Manager, and the AppConnect wrapper](#)

Uploading the AppConnect apps to Core for Android AppConnect quick start

You upload Android AppConnect apps to MobileIron Core in the same manner you add any in-house app. After uploading the apps, you can distribute the apps to devices by applying the apps to labels that contain the appropriate devices.

Procedure

1. In the MobileIron Core Admin Portal, go to **Apps > App Catalog > Add+ > In-House**.
2. Click **Browse** and browse to the AppConnect app.
3. Click **Next**.
4. Optionally make selections, clicking **Next**, and then **Finish**.
5. Select the app entry on the **Apps > Apps Catalog** screen.
6. Click **Actions > Apply to Labels**.
7. Select the appropriate labels and click **Apply**.
8. Repeat for each AppConnect app.

Next steps

Continue to [Enabling Core licensing options for Android AppConnect quick start](#).

Related topics

“Managing Mobile Apps for Android” in the Apps@Work Guide.



Enabling Core licensing options for Android AppConnect quick start

On the Admin Portal, in **Settings > System Settings > Additional Products > Licensed Products**, you specify whether you have a license for:

- AppConnect for third-party and in-house apps
- AppTunnel for third-party and in-house apps
- Docs@Work
- Web@Work

The following table shows which Android secure apps you can deploy for each option. Select each option only if your organization has purchased it.

TABLE 13. ANDROID SECURE APPS YOU CAN DEPLOY WITH EACH LICENSING OPTION

If you enable Docs@Work, you can deploy:	If you enable AppConnect for third-party and in-house apps, you can deploy:	If you enable Web@Work, you can deploy:
<ul style="list-style-type: none"> • Docs@Work • Secure Android Email+ • TouchDown for SmartPhones[*] • Web@Work 	<ul style="list-style-type: none"> • Secure Android Email+ • TouchDown for SmartPhones[*] • FileManager • Other third-party AppConnect apps[*] • In-house AppConnect apps 	<ul style="list-style-type: none"> • Web@Work

^{*}In addition to purchasing the Additional Products option, these apps have an additional cost.

Procedure

1. In the Admin Portal, go to **Settings > System Settings > Additional Products > Licensed Products**.
2. Select the appropriate options.
3. Click **Save**.

Next steps

Continue to [Configuring the AppConnect global policy for Android AppConnect quick start](#).



Configuring the AppConnect global policy for Android AppConnect quick start

Using AppConnect for Android requires that you configure an AppConnect global policy. This policy specifies settings that are not specific to a particular AppConnect app. For example, you configure the AppConnect passcode requirements, data loss protection requirements, and whether device users can use AppConnect apps that have no AppConnect container policy on the device.

This procedure shows using a new AppConnect global policy, but you can also edit the default AppConnect global policy.

Procedure

1. In the Admin Portal, select **Policies & Configs > Policies**.
2. Select **Add New > AppConnect**.
3. Set the **AppConnect** field to **Enabled**.
4. Complete the form.
Most fields default to suitable values.
5. In the **Security Policies** section, select **Authorize** for the field **Apps without an AppConnect container policy**.
6. Click **Save**.

NOTE: If you created a new AppConnect global policy, continue to the next step to apply it to the appropriate labels. You do not need to apply the default AppConnect global policy to a label.

7. Select the policy.
8. Select **More Actions > Apply To Label**.
9. Select the labels to which you want to apply this policy.
10. Click **Apply**.

Next steps

Continue to [Configuring the AppConnect container policy for Android AppConnect quick start](#).

Related topics

[AppConnect global policy](#)



Configuring the AppConnect container policy for Android AppConnect quick start

This task is necessary only if:

- You did not select **Authorize** for the field **Apps without an AppConnect container policy** in the AppConnect global policy. When **Authorize** is not selected, an AppConnect container policy is required to authorize an AppConnect app on a device.
- You want to apply a different screen capture data loss protection (DLP) setting for this app than the setting specified in the AppConnect global policy. This setting in the AppConnect container policy overrides the setting in the AppConnect global policy.

MobileIron Core automatically creates an AppConnect container policy for the Android AppConnect app when you upload the app to the App Catalog. However, you must also create an additional AppConnect container policy for the app if you want to apply a different screen capture DLP setting to different devices.

Note The Following:

- Make sure only one AppConnect container policy for a specific app is applied to each device.
- Core keeps in sync the labels that you apply to the app and the labels that you apply to the automatically-created AppConnect container policy.
- Therefore, make sure that the same labels are not applied to the automatically-created AppConnect container policy and the AppConnect container policy that you manually create.

Procedure

1. In the Admin Portal, select **Policy & Configs > Configurations**.
2. Select the automatically-created AppConnect container policy for the app.
3. Click **Edit**.

Alternatively, to create a new AppConnect container policy for the app:

- a. Select **Actions > Save as**.
 - b. Enter a name for the AppConnect container policy.
 - c. Enter a description for the AppConnect container policy.
4. Select **Allow Screen Capture** if you want to allow screen capture for this app. Your selection overrides the choice for the screen capture setting on the AppConnect global policy.

NOTE: No other settings apply to Android. Also, the ability to open a document is always restricted to the secure container on Android devices.

5. Click **Save**.
6. If creating a new AppConnect container policy:



- a. Select the policy you just created.
- b. Select **More Actions > Apply to Label**.
- c. Select the labels to which you want to apply this policy.
- d. Click **Apply**.
- e. Select the automatically-created AppConnect container policy.
- f. Select **More Actions > Remove from Label**.
- g. Select the labels that you applied to the policy that you just created.
- h. Click **Remove**.

Next steps

Continue to [Configuring settings specific to the app for Android AppConnect quick start](#).

Related topics

[AppConnect container policies](#)

Configuring settings specific to the app for Android AppConnect quick start

Using an AppConnect app configuration, you can configure settings that are specific to an AppConnect app. Because MobileIron Core provides these settings to the app, device users do not have to manually enter configuration details that the app requires. Each AppConnect app's documentation should specify the necessary configuration. The configuration uses key-value pairs.

MobileIron Core automatically creates an AppConnect app configuration for the Android AppConnect app when you upload the app to the App Catalog. However, you can create additional AppConnect app configurations for the app if you want to apply different configurations to the app on different devices.

Note The Following:

- Make sure only one AppConnect app configuration for a specific app is applied to each device.
- Core keeps in sync the labels that you apply to the app and the labels that you apply to the automatically-created AppConnect app configuration. Therefore, make sure that the same labels are not applied to the automatically-created AppConnect app configuration and the AppConnect app configurations that you manually create.



Procedure

1. In the Admin Portal, select **Policy & Configs > Configurations**.
2. Select the automatically-created AppConnect app configuration for the app.
3. Click **Edit**.
Alternatively, to create a new AppConnect app configuration for the app:
 - a. Select **Actions > Save as**.
 - b. Enter a name for the app configuration.
 - c. Enter a description for the app configuration.
4. In the **App-specific Configurations** section, edit the value of existing key-value pairs.
5. To add a key-value pair:
 - a. Click **Add+**.
 - b. Enter the key name as provided by the app's documentation.
 - c. Enter the key value.
6. Click **Save**.
7. If creating a new AppConnect app configuration:
 - a. Select the app configuration you just created.
 - b. Select **More Actions > Apply to Label**.
 - c. Select the labels to which you want to apply this app configuration.
 - d. Click **Apply**.
 - e. Select the automatically-created AppConnect app configuration.
 - f. Select **More Actions > Remove from Label**.
 - g. Select the labels that you applied to the policy that you just created.
 - h. Click **Remove**.

Next steps

Continue to [Configuring email attachment control for Android AppConnect quick start](#).

Related topics

[AppConnect app configuration](#)



Configuring email attachment control for Android AppConnect quick start

This step is required only if:

- You are deploying an AppConnect for Android email app, such as Email+ for Android or Touchdown for SmartPhones.
- You want to use the email attachment control feature in the Standalone Sentry,

Procedure:

1. Go to **Services > Sentry** in the Admin Portal.
2. Select the Standalone Sentry that handles ActiveSync for the devices.
3. Select the edit icon.
4. In the section **Attachment Control Configuration**, select **Enable Attachment Control**.
5. For **iOS And Android Using Secure Email Apps**, select **Open With Secure Email App**.
6. Click **Save**.

Related topics

“Email Attachment Control with Standalone Sentry” in the *MobileIron Sentry Guide for MobileIron Core*.

Quick start configuration for AppConnect for iOS

Complete these tasks to quickly set up AppConnect for iOS on MobileIron Core. However, for more information about each task, including details about all the settings, see the provided references and also details in [Using AppConnect for iOS](#).

These quick start tasks are for adding AppConnect apps from the Apple App Store to the MobileIron Core App Catalog for distribution to iOS devices using the Apps@Work web clip.

Before you begin

Set up the Apps@Work web clip. See “Setting up Apps@Work for iOS and macOS” in the *MobileIron Core Device Management Guide for iOS and macOS Devices*.

Procedure

1. [Adding AppConnect apps to Core for iOS AppConnect quick start](#).
2. [Enabling Core licensing options for iOS AppConnect quick start](#).
3. [Configuring the AppConnect global policy for iOS AppConnect quick start](#).



4. [Configuring the AppConnect container policy for iOS AppConnect quick start.](#)
5. [Configuring settings specific to the app for iOS AppConnect quick start.](#)
6. [Configuring email attachment control for iOS AppConnect quick start.](#)

Adding AppConnect apps to Core for iOS AppConnect quick start

Many third-party AppConnect apps are available in the Apple App Store. By adding them to the MobileIron Core App Catalog, you can distribute them to devices using Apps@Work.

NOTE: You can also add in-house AppConnect apps to the App Catalog. For details see "Using the wizard to add an in-house iOS or macOS app to the App Catalog" in the Apps@Work Guide.

Procedure

1. In the MobileIron Core Admin Portal, go to **Apps > App Catalog**.
2. From the **Quick Import** drop-down list, select **iOS**.
3. Enter the name of the app in the **Application Name** text box.
4. Click **Search**.
5. Select the app from the list that is displayed.
6. Click **Import**.
7. Click **OK** on the pop-up message, and close the **App Store Search** dialog.

The app is now listed in the **App Catalog**. Information included in the app, such as the name, is automatically configured. All other settings, such as the Apps@Work category for the app, are set to default settings.

TIP: To view and edit the settings for the app, click on the app name in the **App Catalog**.

8. Select the app to apply the app to a label:
9. Click **Actions > Apply to Labels**.
10. Select the appropriate labels.
11. Click **Apply**.
12. Repeat this procedure for each AppConnect app that you want to distribute.

Next steps

Continue to [Enabling Core licensing options for iOS AppConnect quick start](#).

Related topics

"Managing Mobile Apps for iOS" in the Apps@Work Guide.



Enabling Core licensing options for iOS AppConnect quick start

On the Admin Portal, in **Settings > System Settings > Additional Products > Licensed Products**, you specify whether you have a license for:

- AppConnect for third-party and in-house apps
- AppTunnel for third-party and in-house apps
- Docs@Work
- Web@Work

The following table shows which iOS secure apps you can deploy for each option. Select each option only if your organization has purchased it.

TABLE 14. iOS SECURE APPS YOU CAN DEPLOY WITH EACH LICENSING OPTION

If you enable Docs@Work, you can deploy:	If you enable AppConnect for third-party and in-house apps, you can deploy:	If you enable Web@Work, you can deploy:
<ul style="list-style-type: none"> • Docs@Work • Web@Work 	<ul style="list-style-type: none"> • Email+ for iOS* • Insight* • Other third-party AppConnect apps* • In-house AppConnect apps 	<ul style="list-style-type: none"> • Web@Work

*In addition to purchasing the Additional Products option, these apps have an additional cost.

Procedure

1. In the Admin Portal, go to **Settings > System Settings > Additional Products > Licensed Products**.
2. Select the appropriate options.
3. Click **Save**.

Next steps

Continue to [Configuring the AppConnect global policy for iOS AppConnect quick start](#).

Configuring the AppConnect global policy for iOS AppConnect quick start

Using AppConnect for iOS requires that you configure an AppConnect global policy. This policy specifies settings that are not specific to a particular AppConnect app. For example, you configure the AppConnect passcode



requirements, data loss protection requirements, and whether device users can use AppConnect apps that have no AppConnect container policy on the device.

This procedure shows using a new AppConnect global policy, but you can also edit the default AppConnect global policy.

1. In the Admin Portal, select **Policies & Configs > Policies**.
2. Select **Add New > AppConnect**.
3. Set the **AppConnect** field to **Enabled**.
4. Complete the form.
Most fields default to suitable values.
5. In the **Security Policies** section, select **Authorize** for the field **Apps without an AppConnect container policy**.
6. Click **Save**.

NOTE: If you created a new AppConnect global policy, continue to the next step to apply it to the appropriate labels. You do not need to apply the default AppConnect global policy to a label.

7. Select the policy.
8. Select **More Actions > Apply To Label**.
9. Select the labels to which you want to apply this policy.
10. Click **Apply**.

Next steps

Continue to [Configuring the AppConnect container policy for iOS AppConnect quick start](#).

Related topics

[AppConnect global policy](#)

Configuring the AppConnect container policy for iOS AppConnect quick start

This task is necessary only if:

- You did not select **Authorize** for the field **Apps without an AppConnect container policy** in the AppConnect global policy. When **Authorize** is not selected, an AppConnect container policy is required to authorize an AppConnect app on a device.
- You want to apply different data loss protection (DLP) settings for this app than the settings specified in the AppConnect global policy. These settings in the AppConnect container policy override the settings in the AppConnect global policy.



Note The Following:

- Make sure only one AppConnect container policy for a specific app is applied to each device.
- Although this quick start procedure is for Apple App Store apps, note that for some in-house iOS AppConnect apps, MobileIron Core automatically creates an AppConnect container policy. Core keeps the automatically-created policy's labels in sync with the labels you apply to the app. If you create additional AppConnect container policies for such apps, make sure to label the policies so that only one policy is applied to each device.

Procedure

1. In the Admin Portal, select **Policy & Configs > Configurations**.
2. Select **Add New > AppConnect > Container Policy**.
3. Enter a name for the policy.
4. Enter a description for the policy.
5. In the Application field, enter the bundle ID for the app. For example, the bundle ID for Email+ for iOS is `com.mobileiron.ios.emailplus`.
6. Configure the iOS data loss prevention settings according to your requirements.
7. Click **Save**.
8. Select the policy you just created.
9. Select **More Actions > Apply to Label**.
10. Select the labels to which you want to apply this policy.
11. Click **Apply**.

Next steps

Continue to [Configuring settings specific to the app for iOS AppConnect quick start](#).

Related topics

[AppConnect container policies](#)

Configuring settings specific to the app for iOS AppConnect quick start

Using an AppConnect app configuration, you can configure settings that are specific to an AppConnect app. Because MobileIron Core provides these settings to the app, device users do not have to manually enter configuration details that the app requires. Each AppConnect app's documentation should specify the necessary configuration. The configuration uses key-value pairs.

Note The Following:



- Make sure only one AppConnect app configuration for a specific app is applied to each device.
- Although this quick start procedure is for Apple App Store apps, note that for some in-house iOS AppConnect apps, MobileIron Core automatically creates an AppConnect app configuration. Core keeps the configuration's labels in sync with the labels you apply to the app. If you create additional AppConnect app configurations for such apps, make sure to label the AppConnect app configurations so that only one is applied to each device.

Procedure

1. In the Admin Portal, select **Policy & Configs > Configurations**.
2. Click **Add New > AppConnect > Configuration** to create a new AppConnect app configuration.
3. Enter a name for the app configuration.
4. Enter a description for the app configuration.
5. In the **App-specific Configurations** section, add key-value pairs, as specified by the app's documentation.
 - a. Click **Add+**.
 - b. Enter the key name as provided by the app's documentation.
 - c. Enter the key value.
6. Click **Save**.
7. Select the app configuration you just created.
8. Select **More Actions > Apply to Label**.
9. Select the labels to which you want to apply this app configuration.
10. Click **Apply**.

Next steps

Continue to [Configuring email attachment control for iOS AppConnect quick start](#).

Related topics

[AppConnect app configuration](#)

Configuring email attachment control for iOS AppConnect quick start

This step is required only if:

- You are deploying an AppConnect for iOS email app, such as Email+ for iOS.
- You want to use the email attachment control feature in the Standalone Sentry,



Procedure:

1. Go to **Services > Sentry** in the Admin Portal.
2. Select the Standalone Sentry that handles ActiveSync for the devices.
3. Select the edit icon.
4. In the section **Attachment Control Configuration**, select **Enable Attachment Control**.
5. For **iOS And Android Using Secure Email Apps**, select **Open With Secure Email App**.
6. Click **Save**.

Related topics

“Email Attachment Control with Standalone Sentry” in the *MobileIron Sentry Guide for MobileIron Core*.



Using AppConnect for Android

- [When an Android device user can use AppConnect for Android](#)
- [AppConnect for Android apps](#)
- [Hybrid web app support](#)
- [Fingerprint login for AppConnect apps for Android](#)
- [AppTunnel with TCP tunneling support for Android secure apps](#)
- [Configuring AppTunnel with TCP tunneling for Android secure apps](#)
- [Certificate authentication using AppConnect with TCP tunneling for Android secure apps](#)
- [Configuring certificate authentication using AppTunnel with TCP tunneling for Android secure apps](#)
- [AppTunnel and TLS protocol versions in Android secure apps](#)
- [Lock, unlock, and retire impact on AppConnect for Android](#)
- [Lock Android AppConnect apps when screen is off](#)
- [Copy/Paste for AppConnect for Android](#)
- [Sharing content from AppConnect for Android apps to non-AppConnect apps](#)
- [Web-related DLP policies](#)
- [DLP policy for media player access](#)
- [Device-initiated security controls for AppConnect for Android](#)
- [Custom keyboards in AppConnect apps](#)
- [Secure File Manager features](#)
- [Secure folder access](#)
- [About allowing a secure app to ignore the auto-lock time](#)
- [Situations that wipe Android AppConnect app data](#)
- [Accessible Android apps to preserve the user experience](#)
- [Secure Apps Manager Android permissions](#)
- [Disabling analytics data collection for AppConnect for Android](#)

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.



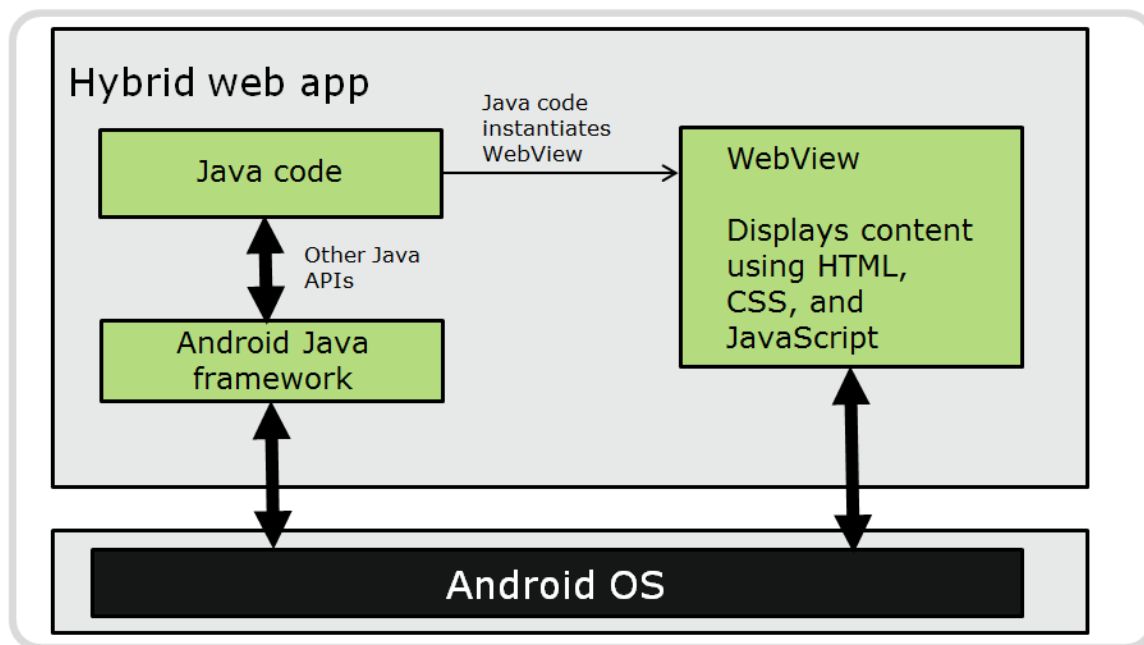
NOTE: Web@Work for Android, the secure browser that MobileIron provides, allows you to run pure web apps in the AppConnect secure container.

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 2. A HYBRID WEB APP ON AN ANDROID DEVICE



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 15. REQUIRED PRODUCT VERSIONS FOR FINGERPRINT LOGIN FOR SECURE APPS

Product	Version
Mobile@Work for Android	9.5.0.0 through the most recently released version as supported by MobileIron.
Secure Apps Manager	8.0 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 UEM:

On the MobileIron Core Admin Portal,

- The fingerprint option is enabled in the AppConnect global policy.
- The block fingerprint option is not enabled in the Security policy.

NOTE: If fingerprint in the security policy is blocked, selecting the fingerprint option in the AppConnect global policy 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 (Core)

Configure fingerprint login for AppConnect apps on the MobileIron Core Admin Portal.

Procedure

1. On the Admin Portal, go to **Policies & Configs > Policies**.
2. Select the appropriate AppConnect global policy.
3. Click **Edit**.

The AppConnect global policy displays.



4. Select **Passcode is required for Android devices**.
5. Select **Use fingerprint authentication when supported**.
6. Click **Save**.
7. Select the appropriate security policy.
8. Scroll down to the **Android** section.
9. Make sure **Block Fingerprint (from Android 5.0 or Samsung MDM 5.3)** is **not** selected.
10. Click **Save**.

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 following 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 UEM 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 16. 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 16. LESS COMMON DEVICE USER SCENARIOS RELATING TO FINGERPRINT LOGIN (CONT.)

Scenario	Behavior on the device
<ol style="list-style-type: none"> 1. Fingerprint login is available for secure apps. 2. A device user creates a new secure apps passcode because the user forgot the passcode. 	<p>The device user must again choose whether to enable fingerprint login.</p> <p>NOTE: This case applies when the device user initiates the "forgot passcode" scenario or the administrator unlocks the AppConnect container from the Admin Portal.</p>
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.	<p>The Secure Apps Manager prompts the device user to change the secure apps passcode after the user next logs in.</p> <p>This behavior is similar to changing any of these secure apps passcode characteristics on the AppConnect global policy:</p> <ul style="list-style-type: none"> - passcode type - minimum passcode length - minimum number of complex characters - passcode strength usage or level changes <p>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.</p>
You change the Block Fingerprint option on the security policy.	<p>The Secure Apps Manager prompts the device user to change the secure apps passcode after the user next logs in.</p> <p>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.</p>

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 the MobileIron UEM 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 17. SECURITY VERSUS DEVICE USER CONVENIENCE OF PASSCODE AND FINGERPRINT OPTIONS

Passcode and fingerprint configuration on MobileIron UEM	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 17. SECURITY VERSUS DEVICE USER CONVENIENCE OF PASSCODE AND FINGERPRINT OPTIONS (CONT.)

Passcode and fingerprint configuration on MobileIron UEM	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](#)

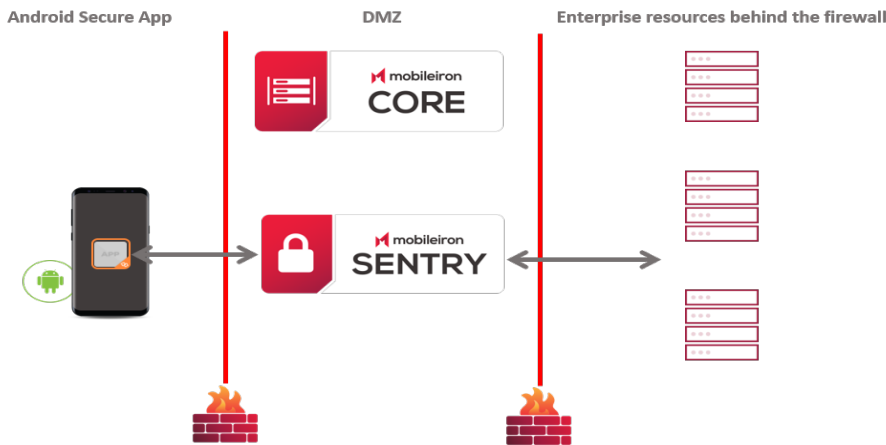
AppTunnel with TCP tunneling support for Android secure apps

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 3. 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 secure 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 secure app. It also shows which generation of the wrapper to use.

TABLE 18. APPTUNNEL SUPPORT FOR HTTP/S VERSUS TCP TUNNELING ON ANDROID SECURE 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.

Configuring AppTunnel with TCP tunneling for Android secure apps

The procedure to configure AppTunnel with TCP tunneling is mostly the same as the procedure to configure AppTunnel with HTTP/S tunneling. The difference involves the AppTunnel service that you configure on the Standalone Sentry. To see just this difference, see [Configuring an AppTunnel TCP service](#).

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 Core 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.

For details about these required tasks, as well as optional tasks, see the “Configuring Standalone Sentry for AppTunnel” in the *MobileIron Sentry Guide*.

Overview

1. Complete the steps in [Basic configuration](#).
2. Complete the steps in [Adding third-party and in-house secure apps](#), if applicable.
3. Enable AppTunnel on MobileIron Core, if you are deploying third-party or in-house apps.
See [Enabling AppTunnel](#).
4. Configure an AppTunnel TCP service on Standalone Sentry.
See [Configuring an AppTunnel TCP service](#)
5. Configure an AppConnect app configuration.
See [Configuring the AppTunnel TCP service in the AppConnect app configuration](#).
6. Control the idle session timeout for the TCP connection between the app and the enterprise server.
See [Configuring per-app idle session timeout for AppTunnel with TCP tunneling](#).
7. Change the TLS protocol version to use TLSv1.2 instead of TLS1.0, if required by the Standalone Sentry.
See [Configuring AppTunnel with TCP tunneling for Android secure apps](#).

Related topics

“Working with app tunnels” in the *MobileIron Sentry Guide* for actions you can take on an app tunnel. For example, you can block an app tunnel, so that an AppConnect app on a device cannot access the backend resource.

Configuring an AppTunnel TCP service

An AppTunnel TCP service defines the backend service that an AppConnect app tunnels to using TCP tunneling.

See “Standalone Sentry for AppTunnel” in the *MobileIron Sentry Guide for MobileIron Core* for information about configuring AppTunnel and an AppTunnel service. Standalone Sentry product documentation is available on the [Standalone Sentry Product Documentation Home Page](#).

About the AppTunnel TCP service name

When you configure an AppTunnel service, you give the service a service name. The service name is used in the AppConnect app configuration. The app configuration uses the service name to restrict the app to accessing servers in the **Server List** field associated with the service name.



The service name is one of the following:

- A unique name for the TCP service that the AppConnect app on the device accesses
One or more of your internal app servers provide the service. You list the servers in the **Server List** field associated with the service name.
For AppTunnel with TCP tunneling, the name must begin with **TCP** (case-insensitive).
Example: **TCP_Finance**
A service name cannot contain these characters: 'space' \ ; * ? < > " |.
- **<TCP_ANY>**
Select **<TCP_ANY>** for the service name to allow AppTunnel with TCP tunneling to any URL that the app requests. Typically, you select **<TCP_ANY>** if an AppConnect app's app configuration specifies a URL with wildcards for tunneling, such as *.myCompany.com. The Sentry tunnels the data for any URL request that the app makes that matches the URL with wildcards.
The Sentry tunnels the data to the app server that has the URL that the app specified. The **Server List** field is therefore not applicable when the Service Name is **<TCP_ANY>**.
For example, consider when the app requests URL myAppServer.mycompany.com, which matches *.mycompany.com in the app configuration. The Sentry tunnels the data to myAppServer.myCompany.com.

Configuring the AppTunnel TCP service in the AppConnect app configuration

The AppConnect app configuration specifies the AppTunnel TCP service that the app uses.

Procedure

1. In the Admin Portal, select **Policies & Configs > Configurations**.
2. Select **Add New > AppConnect > App Configuration**.
Alternatively, edit an existing AppConnect app configuration.
3. Enter a name for the AppConnect app configuration.
For example: My App's App Configuration
4. In the **Application** field, select the secure app from the App Catalog.
5. In the **AppTunnel Rules** section, click **Add+** to add a new AppTunnel rule.
6. Set up the TCP tunnel information as described in the following table:

Item	Description
Sentry	Select a Standalone Sentry configured for app tunneling from the drop-down list.
Service	<p>Select a TCP service name from the drop-down list.</p> <p>This service name specifies an AppTunnel service configured in the AppTunnel Configuration section of the specified Standalone Sentry.</p> <p>NOTE: If you entered a URL with wildcards in the URL Wildcard field, you can only select <TCP_ANY> as the service. The <TCP_ANY> service must be configured in the AppTunnel Configuration section of the Standalone Sentry configured for AppTunnel.</p>
URL Wildcard	<p>Enter one of the following:</p> <ul style="list-style-type: none"> • an enterprise app server's hostname Example: finance.yourcompany.com



Item	Description
	<ul style="list-style-type: none"> a hostname with wildcards. The wildcard character is <code>*</code>. Example: <code>*.yourcompanyname.com</code> <p>If the app requests to access this hostname, the Sentry tunnels the app data to an app server. The Sentry and Service fields that you specify in this AppTunnel row determine the target app server.</p> <p>Note The Following:</p> <ul style="list-style-type: none"> The app data is tunneled only if the app's request matches this hostname and the port number specified in the Port field of this AppTunnel row. The order of these AppTunnel rows matters. If you specify more than one AppTunnel row, the first row that matches the hostname and port that the app requested is chosen. That row determines the Sentry and Service to use for tunneling.
Port	<p>Enter the port number that the app requests to access.</p> <p>The app data is tunneled only if the app's request matches the hostname in the URL Wildcard field and this port number. If you do not enter a port number, the port in the app's request is not used to determine whether data is tunneled.</p> <p>NOTE: Entering a port number in this field is required when both of the following are true:</p> <ul style="list-style-type: none"> The hostname in the URL Wildcard field does not contain a wildcard. The service is not <TCP_ANY>.
Identity Certificate	<p>Select the Certificate Enrollment setting that you created for AppTunnel. This selection determines the certificate that the device presents to the Standalone Sentry for authentication.</p> <p>See "Device and server authentication" in the <i>MobileIron Sentry Guide</i>.</p>

- Click **Save**.
- Select the new AppConnect app configuration.
- Select **More Actions > Apply To Label**.
- Select the labels to which you want to apply this AppConnect app configuration.
- Click **Apply**.

Related topics

[AppConnect app configuration](#)

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 19. IDLE SESSION TIMEOUT KEY-VALUE PAIR

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

Procedure

1. In the Admin Portal, select **Policies & Configs > Configurations**.
2. Select the AppConnect app configuration for the AppConnect app (The **Setting Type** is **AppConfig**).
3. In **App-specific Configurations**, select **Add+** to add a key-value pair.
4. Enter **MI_AC_TCP_IDLE_TIMEOUT_MS** for the key.
5. Enter the idle session timeout value in milliseconds.
6. Click **Save**.

Certificate authentication using AppConnect with TCP tunneling for Android secure apps

Android Secure Apps supports certificate authentication using AppTunnel with TCP tunneling. A secure 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 secure apps

Overview

The procedure to configure certificate authentication with AppTunnel with TCP tunneling starts with the procedure to configure AppTunnel with TCP tunneling. In addition, you add key-value pairs to the app's AppConnect app configuration that specify:

- The AppTunnel TCP services that you configure 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.

The following excerpt from the Standalone Sentry configuration and the AppConnect app configuration for a Finance app and Helpdesk app summarize this additional configuration:

FIGURE 4. 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

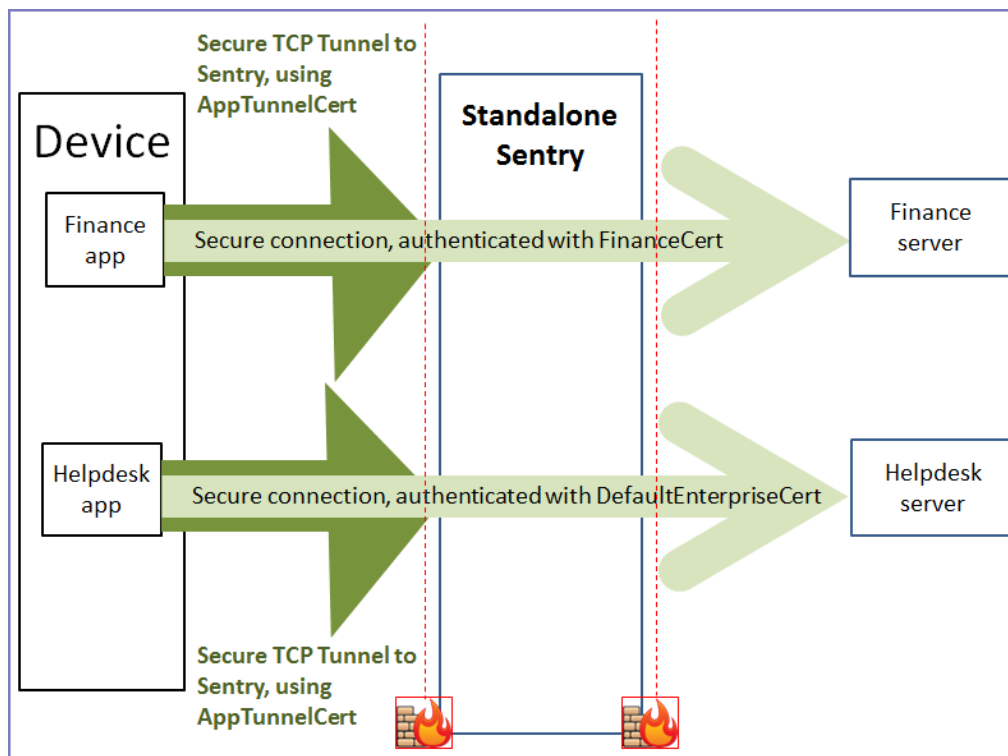
The Finance app and the Helpdesk app:

- Authenticate to the 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 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 5. CERTIFICATE USAGE IN CERTIFICATE AUTHENTICATION WITH TCP TUNNELING



High-level tasks for certificate authentication using AppTunnel with TCP tunneling

Do the following tasks to set up certificate authentication using AppTunnel with TCP tunneling:

1. Configure AppTunnel with TCP tunneling for the app.
See [Configuring AppTunnel with TCP tunneling for Android secure apps](#).
2. [Setting up the certificate for authenticating the user to the enterprise server](#)

3. [Specifying the AppTunnel services that use certificate authentication](#)
4. [Specifying which certificate to use to authenticate the user to the enterprise server](#)

Setting up the certificate for authenticating the user to the enterprise server

You specify the certificate that the app uses to authenticate the user to the enterprise server. The certificate is either an identity certificate or a group certificate.

This certificate can be:

- a specific certificate that is used for a specific enterprise service
- a default certificate used for enterprise services that do not require a specific certificate.
- the same certificate that authenticates the user to the Standalone Sentry.
The app uses the Sentry certificate to authenticate to the enterprise service if you do not specify another certificate, specific or default, for a service.

If you require a specific certificate or an default certificate other than the certificate you already set up for the Standalone Sentry, set up the certificate in the Admin Portal:

1. Go to **Policies & Configs > Configurations**.
2. Select **Add New > Certificate Enrollment**.
3. Configure the Certificate Enrollment setting as described in detail in “Certificate Enrollment settings” in the *MobileIron Core Device Management Guide for Android and Android enterprise Devices*.

Specifying the AppTunnel services that use certificate authentication

The AppConnect app configuration specifies the AppTunnel services that your secure app uses. It refers to the AppTunnel services that you configured on the Standalone Sentry as described in [Configuring the AppTunnel TCP service in the AppConnect app configuration](#).

You also specify in the AppConnect app configuration which of those AppTunnel services use certificate authentication. Do the following steps:

1. In the Admin Portal, select **Policies & Configs > Configurations**.
2. Select the AppConnect app configuration for your secure app.
3. Click **Edit**.
4. In the **App-specific Configurations** section, click **Add+** to add a key-value pair.
5. For the key, enter **ES_CERT_AUTH_SERVICES**, which is case sensitive.
6. For the value, enter the list of AppTunnel services that your app uses. Typically apps use only one AppTunnel service, but using multiple AppTunnel services is supported. Separate the services with a semi-colon.

Examples:

TCP_HELPDESK

TCP_HELPDESK;TCP_WIKI;TCP_FINANCE

In these examples, TCP_HELPDESK, TCP_WIKI and TCP_FINANCE are services defined on the Standalone Sentry in the **AppTunnel Configuration** section.

Make sure that each listed service exactly matches, including case, the AppTunnel service name.

7. Click **Save**.



Specifying which certificate to use to authenticate the user to the enterprise server

To specify the certificate for the user to authenticate to the enterprise server, you add a key-value pair to the AppConnect app configuration, as described in the following table:

TABLE 20. CERTIFICATE AUTHENTICATION KEY-VALUE PAIRS

Key	Value
<p><service_name>_CERT</p> <p>where <service_name> is one of the AppTunnel services that the app uses, which you listed in Specifying the AppTunnel services that use certificate authentication.</p> <p>Example:</p> <p>TCP_HELPDESK_CERT</p> <p>NOTE: The key is case sensitive. Make sure that the <service_name> exactly matches, including case, the AppTunnel service name.</p>	<p>The Certificate Enrollment setting for a certificate used specifically for this AppTunnel service.</p> <p>The configured Certificate Enrollment settings appear in the value field's dropdown list.</p> <p>Note The Following:</p> <ul style="list-style-type: none"> 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.
<p>ES_DEFAULT_CERT</p> <p>NOTE: The key is case sensitive.</p>	<p>The Certificate Enrollment setting for a default certificate used for services that do not require a specific certificate.</p> <p>The configured Certificate Enrollment appear in the value field's dropdown list.</p>

To specify the certificate in the AppConnect app configuration:

1. In the Admin Portal, select **Policies & Configs > Configurations**.
2. Select the AppConnect app configuration for your secure app.
3. Click **Edit**.
4. In the **App-specific Configurations** section, click **Add+** to add a key-value pair.
5. Enter a key named **<service_name>_CERT** for a certificate used specifically for the AppTunnel service, or enter a key named **ES_DEFAULT_CERT** for a default certificate.
6. For the value, select the Certificate Enrollment that you configured for the certificate. The Certificate Enrollment settings appear in the value field's dropdown list.
7. Click **Save**.

AppTunnel and TLS protocol versions in Android secure 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 21. 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

- “Incoming SSL configuration” in the *MobileIron Sentry Guide for MobileIron Core*
- [AppTunnel with TCP tunneling support for Android secure apps](#)
- [Certificate authentication using AppConnect with TCP tunneling for Android secure apps](#)



Configuring the TLS protocol for AppTunnel

You can configure an AppConnect app to use a TLS protocol version other than the default versions by using the key-value pairs described in [AppTunnel and TLS protocol versions in Android secure apps](#).

Procedure

1. In the MobileIron Core Admin Portal, go to **Policies & Configs > Configurations**.
2. Select the appropriate setting for the app.
For Docs@Work, select a Docs@Work setting.
For Web@Work, select a Web@Work setting.
For other secure apps, select an AppConnect app configuration.
3. Click **Edit**.
4. In the **App-specific Configurations** section (called **Custom Configurations** for Docs@Work and Web@Work settings), click **Add+**.
5. For apps wrapped with AppConnect wrapper 8.0 through 8.4, add the key **MI_AC_USE_TLS1.2** with the value **true** if you want to use only TLSv1.2 instead of TLSv1.0.
6. For apps wrapped with AppConnect wrapper 8.5 through the most recently released version as supported by MobileIron, add the key **MI_AC_ENABLE_TLS_FALLBACK** with the value **true** if you want to fallback to using TLSv1.0 if TLSv1.2 is not accepted.
7. Click **Save**.

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 the MobileIron UEM, 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 Core deployments: “Lock” in the *MobileIron Core Device Management Guide for Android and Android enterprise Devices*



Unlock the AppConnect container impact

When you unlock a device from the MobileIron UEM, the device passcode is removed. However, the AppConnect passcode is not impacted. Unlocking the AppConnect container using the **AppConnect Unlock**(Cloud) or the **Unlock AppConnect Container** (Core) 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:

- the MobileIron client (MobileIron Go or Mobile@Work)
- the 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.
See "Self-service AppConnect passcode recovery" in [Configuring the AppConnect global policy](#).
- The device user has exceeded the maximum number of failed attempts for the secure apps passcode.

Related topics

- MobileIron Core deployments: "Unlock" in the *MobileIron Core Device Management Guide for Android and Android enterprise Devices*

Retire impact

Retiring a device unregisters the device from MobileIron UEM.

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.

However, device users must manually uninstall the AppConnect apps and the Secure Apps Manager.

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

Related topics

- MobileIron Core deployments: "AppConnect app authorization" in [Configuring AppConnect container policies](#) for more information about retiring AppConnect app



Lock Android AppConnect apps when screen is off

You can lock device users out of AppConnect apps when the device screen is turned due to either inactivity or user action. When locked out of AppConnect apps, the device user must reenter the AppConnect passcode (or fingerprint) to access AppConnect apps. Locking AppConnect apps when the screen is turned off provides added security to the device.

Reasons for the screen turning off include:

- The device user manually turning off the screen, but not locking it.
When the screen is off but not locked, the device user can turn on the screen without entering any credentials, such as the device password or fingerprint.
- The device user manually locking the screen.
- The device's automatic lock timeout expires, as set in the device's settings.

How you configure this feature depends on the version of Secure Apps Manager running on the device:

TABLE 22. LOCKING ANDROID APPCONNECT APPS WHEN SCREEN IS OFF

Secure Apps Manager version	Configuration
Secure Apps Manager 8.3.0 through the most recently released version as supported by MobileIron	Select Lock AppConnect apps automatically when the screen is off on the AppConnect global policy. See Configuring the AppConnect global policy .

Copy/Paste for AppConnect for Android

You configure the copy/paste DLP policy for AppConnect for Android in the AppConnect Device configuration on MobileIron Cloud or in the AppConnect global policy on MobileIron Core. 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 23. 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 24. 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 <i>from</i> an unsecured app <i>into</i> an AppConnect app.
Restriction levels	The copy/paste policy provides these restriction levels: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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:

- For MobileIron Core deployments, on the AppConnect app configuration for Secure Apps Manager.



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 25. 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 apps and Email+

Sharing content from AppConnect for Android apps to non-AppConnect apps

By default, AppConnect apps cannot share content with non-AppConnect apps. However, you can allow content sharing with non-AppConnect apps. When you allow it, when a device user selects content in an AppConnect app to share, the user is presented with the list of all apps that can handle the content rather than just a list of AppConnect apps that can handle the content. The content can be any of the following:

- text
- images
- video



You cannot allow sharing only from particular AppConnect apps. Sharing is allowed from either all AppConnect apps or not at all.

To allow content sharing with non-AppConnect apps, you add the key `MI_AC_SHARE_CONTENT` with the value `true`.

You add the key-value pair:

- For MobileIron Core deployments, on the AppConnect app configuration for Secure Apps Manager.

This feature requires:

- Secure Apps Manager 8.3 through the most recently released version as supported by MobileIron
- Apps wrapped with wrapper version 8.3 through the most recently released version as supported by MobileIron

Procedure

1. In the MobileIron Core Admin Portal, go to **Policies & Configs > Configurations**.
2. Select the AppConnect app configuration for the Secure Apps Manager.
3. Click **Edit**.
4. In the **App-specific Configurations** section, click **Add+**.
5. Add the key **MI_AC_SHARE_CONTENT** with the value **true**.
6. Click **Save**.

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 26. 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:

- For MobileIron Core deployments, on the AppConnect global policy in the data loss prevention policies section for Android.
- For MobileIron Cloud deployments, 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 27. 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 UEM. You can choose whether to allow AppConnect apps to stream media to media players on the device.

You configure the setting in the following:

- AppConnect global policy on MobileIron Core.
- 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, Mobile@Work 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 Core is not required for these security controls.

Configure the actions on the AppConnect global policy

The AppConnect global policy provides settings to specify whether you want to retire all secure apps when the device is compromised or USB debugging is enabled. However, after the device has checked in and received the AppConnect global policy, no further interaction is required from Core. Mobile@Work detects the non-compliant situation and retires the secure apps.

Because Mobile@Work acts independently of Core when these security violations occur, retiring secure apps occurs before any actions specified on other policies such as the security policy.

To configure that you want the device to detect these security violations and then retire secure apps:

1. In the Admin Portal, go to **Policies & Configs > Policies**.
2. Select the AppConnect global policy that is applied to the devices of interest.
3. Click **Edit**.
4. Scroll to the section **AppConnect Security Controls on Device**.
5. In the **Android** section, select **Wipe AppConnect Data for Device Compromised** and **USB Debug Enabled**, according to your security requirements.
6. Click **Save**.

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

Custom keyboards in AppConnect apps

Device users often install and use third-party custom keyboards rather than the pre-installed keyboards on a device. When used in AppConnect apps, custom keyboards have the potential to leak sensitive data. Therefore, you can:

- Disable the use of all custom keyboards in AppConnect apps.
This action means that if any custom keyboard is enabled on the device, even if it is not currently the default keyboard, the device user cannot access AppConnect apps.
- Disable the use of all custom keyboards in AppConnect apps except for the keyboards that you specify in a whitelist.
This action means that the device user can access AppConnect apps only if the custom keyboards on the whitelist and the pre-installed keyboards are enabled. If any other custom keyboard is enabled on the device, even if it is not currently the default keyboard, the device user cannot access AppConnect apps.

To disable custom keyboards and set up a whitelist, add the following key-value pair

- Key: `MI_DISABLE_CUSTOM_KEYBOARD`
- Value: `true`

To allow only certain custom keyboards, add the following key-value pair:

- Key: `MI_CUSTOM_KEYBOARD_WHITELIST`
- Value: Package names of the whitelisted keyboard apps, separated by semi-colons.

Example

`com.touchtype.swiftkey;com.google.android.inputmethod.latin;com.syntellia.fleksy.keyboard`

You add the key-value pair:

- For MobileIron Core deployments, on the AppConnect app configuration for Secure Apps Manager.



Note The Following:

- These actions have no impact on keyboard use in the Secure Apps Manager.
 - When using this feature, even when the default keyboard is not one of the unallowed keyboards, if any non-whitelisted, custom keyboards are enabled, the device user cannot access AppConnect apps. Therefore, instruct your device users to disable non-whitelisted custom keyboards to access AppConnect apps.
- In the device's **Settings**, in for example **General Management > Language and input**, device users can:
- View the pre-installed keyboards and the custom keyboards that they have installed.
 - Manage the custom keyboards, which involves disabling or enabling them.
 - Choose the default keyboard from the set of enabled custom keyboards and pre-installed keyboards.

App whitelist

Administrators can configure an app whitelist to control if links in an AppConnect app can be opened in non-AppConnect apps. The combination of apps and domains configured in the whitelist determine:

- which links can be opened in non-AppConnect apps.
- the non-AppConnect apps presented to the Android device user in which to open the link.

Configuring the whitelist allows device users to choose a non-AppConnect app, such as WebEx or GoToMeeting, to open from a link in an AppConnect app such as Email+.

NOTE: The whitelist is ignored if the **Web** option in the AppConnect Global policy is enabled. If the **Web** option is enabled, device users can open a link in any non-AppConnect browser on the device.

The value in the whitelist is a combination of the application ID and domain and defines which domains can be opened and the app in which to open the domain. When device users click on a link in an AppConnect app such as Email+, the device user can choose from the following apps:

- All AppConnect apps installed on the device that can open the link.
- If a whitelist is configured, non-AppConnect apps on the device that can open the link.
The non-AppConnect apps that are presented depends on whether the domain in the link is configured in the whitelist and if the application ID for a non-AppConnect app is configured in the whitelist.

Key-value pair for the app whitelist

You create the app whitelist by adding a key-value pair in the AppConnect app configuration for Secure Apps Manager. To configure the whitelist, add the following key: `MI_AC_ALLOW_OPEN`. The value is a matching rule in any of the following formats:

- *domain*: Specifies the domain that can be opened in non-AppConnect apps. The domain can include the wildcard character `*`.



Example :

- Wildcard domain: *.webex.com : Any URL whose host ends with webex.com can be opened in any compatible non-AppConnect app.
- Multiple domains: *.webex.com webex.com: Any URL whose host ends with webex.com and WebEx links can be opened in a compatible non-AppConnect app.
- *domain* in *applicationID*: Specifies the non-AppConnect app in which the domain can be opened.

Example :

- Wildcard domain in an application ID: *.webex.com in com.cisco.webex.meetings: Any URL whose host ends with webex.com can be opened in the WebEx app.
- Multiple domains in an application ID: *.webex.com webex.com in com.cisco.webex.meetings: Any URL whose host ends with webex.com and WebEx links can be opened in the WebEx app.
- *in applicationID*: Specifies the non-AppConnect app in which any link can be opened.

Example :

- Application ID: in com.cisco.webex.meetings: Any URL can be opened in the WebEx app.

The following describes the characteristics of the matching rules:

- The value can configure multiple rules.
- Each rule is separated by a semi colon.
- Each rule can have several domain patterns followed by an optional keyword *in*.
- The application IDs are listed after the *in* keyword.
- If an application ID is not configured, device users see all non-AppConnect apps on the device that can open the link.
- If the rule contains only an application ID, the app is available for selection for any compatible link.

App whitelist examples

The following table provides examples of the value for various scenarios.

TABLE 28. EXAMPLES OF VARIOUS SCENARIOS

Example scenario	Value
Allow WebEx links to open in the native WebEx app	*.webex.com webex.com in com.cisco.webex.meetings
Allow WebEx links to open in any non-AppConnect browser and in the WebEx app	*.webex.com webex.com
Allow any compatible links to open in the native WebEx app	in com.cisco.webex.meetings
Allow WebEx links to open in the native WebEx app and GoToMeeting links to open in Chrome	*.webex.com webex.com in com.cisco.webex.meetings ; www.gotomeeting.com in com.android.chrome



How the app whitelist is evaluated

The whitelist is evaluated each time a user opens a link from an AppConnect app. When users open a link in an AppConnect app:

1. The request is redirected to Secure Apps Manager.
2. The Secure Apps Manager checks whether the **Web** option in the Global AppConnect policy is enabled.
3. If the **Web** option is enabled, users can open the link in a non-AppConnect browser and the whitelist is ignored.
4. If the **Web** option is not enabled, Secure Apps Manager checks the MI_AC_ALLOW_OPEN key value.
5. The Secure Apps Manager builds a list of external apps that are able to view the link.
6. The Secure Apps Manager filters out the non-AppConnect apps that are not in the whitelist.

NOTE: If an application ID is not configured, Secure Apps Manager does not filter out any non-AppConnect app.

7. The device user can select from the AppConnect apps and non-AppConnect apps in the whitelist that can open the link.

NOTE: Whitelist apps are available for selection to device users only if the domain pattern in the link matches exactly with the domain pattern configured in the key-value pair.

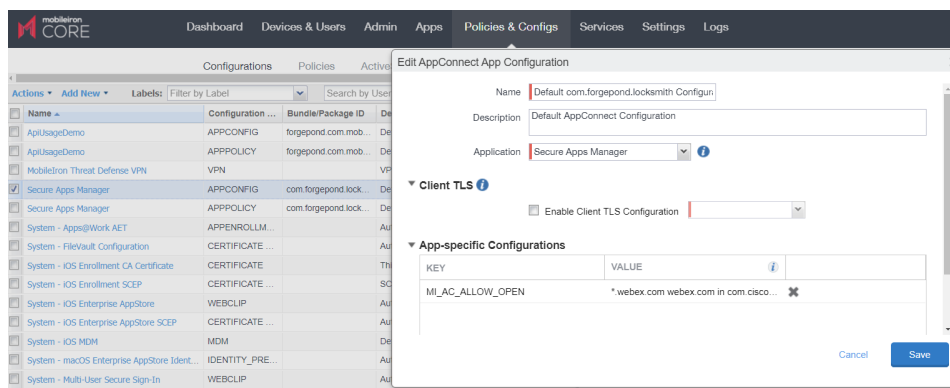
Configuring an app whitelist

You configure the app whitelist in the AppConnect app configuration for Secure Apps Manager.

Procedure

1. In the MobileIron Core Admin Portal, go to **Policies & Configs > Configurations**.
2. Select the AppConnect app configuration for the Secure Apps Manager.
3. Click **Edit**.
4. In the **App-specific Configurations** section, click **Add+**.

FIGURE 6. ADD KEY-VALUE PAIR TO APP-SPECIFIC CONFIGURATION



5. Add the key **MI_AC_ALLOW_OPEN**.



6. Enter a value for the key.
7. Click **Save**.

Related topics

For information about the formats supported for the value, see [App whitelist](#)

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 Core deployments: on the AppConnect global policy.

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:

- You disable AppConnect in the AppConnect global policy for the device.
- The device user uninstalls the MobileIron UEM client (MobileIron Go or Mobile@Work) or the Secure Apps Manager on the device
- You retire the device.
- The number of days specified in the “Wipe AppConnect Data After” field of the device’s AppConnect global policy has passed.
- You remove the Secure Apps Manager in **Apps > App Catalog**.
- You remove the label for a device from the Secure Apps Manager in **Apps > App Catalog**.
- 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 29. 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: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • A secure app requests access to the phone. 	The secure app cannot access the phone.



TABLE 29. SECURE APPS MANAGER PERMISSION REQUESTS (CONT.)

Permission	Reason needed	Requested at these times	Behavior if not granted
		<ul style="list-style-type: none"> The device user has not yet granted access to the phone for secure apps. 	
Contacts	A secure app wants to access the device's contacts.	When both of the following are true: <ul style="list-style-type: none"> 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.

Disabling 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. The data collected involving AppConnect apps are the MobileIron key-value pairs set in MobileIron Core in the AppConnect app configuration of AppConnect apps or the Secure Apps Manager.

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:

- For MobileIron Core deployments, on the AppConnect app configuration for Secure Apps Manager.

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

Procedure

1. In the MobileIron Core Admin Portal, go to **Policies & Configs > Configurations**.
2. Select the AppConnect app configuration for the Secure Apps Manager.
3. Click **Edit**.
4. In the **App-specific Configurations** section, click **Add+**.
5. Add the key **MI_AC_DISABLE_ANALYTICS** with the value **true**.
6. Click **Save**.



Using AppConnect for iOS

- [When an iOS device user can use AppConnect for iOS](#)
- [AppConnect apps that MobileIron provides for iOS](#)
- [MobileIron UEM client for iOS and AppConnect apps](#)
- [Dual-mode apps](#)
- [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 pinning for AppConnect apps](#)
- [Certificate authentication from AppConnect apps to enterprise services](#)
- [Impact to tunneling when using a global HTTP proxy](#)

Open-In data loss prevention policy details

- [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](#)

Related topics

- [Configuring the AppConnect global policy](#)
- [Configuring AppConnect container policies](#)

Open In behavior in wrapped apps versus SDK apps

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

- [MobileIron Core: on the AppConnect global policy or the AppConnect container policy.](#)



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

TABLE 30. OPEN IN BEHAVIOR IN WRAPPED APPS VERSUS SDK APPS

	Open In Not allowed	Open In is allowed only to AppConnect apps or Open In only to apps on 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? Only AppConnect 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
SDK apps		

TABLE 30. OPEN IN BEHAVIOR IN WRAPPED APPS VERSUS SDK APPS (CONT.)

	Open In Not allowed	Open In is allowed only to AppConnect apps or Open In only to apps on whitelist
Apps built with AppConnect 3.0 for iOS and earlier	<p>Open In policy enforced by:</p> <p>The app.</p> <p><i>App behavior can vary depending on the app's implementation.</i></p>	<p>Open In policy enforced by:</p> <p>The app.</p> <p><i>App behavior can vary depending on the app's implementation.</i></p>
Apps built with AppConnect 3.1 through 3.1.3 for iOS	<p>Open In policy enforced by: The AppConnect library, contained in the AppConnect app.</p> <p>App is responsible for:</p> <p>Disabling user interfaces, such a menu items, that provide the Open In capability.</p> <p>What if the app initiates Open In anyway?</p> <p>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. Target app error handling varies. For example, some apps display an error pop-up.</p>	<p>Open In policy enforced by: The AppConnect library, contained in the AppConnect app.</p> <p>App is responsible for:</p> <p>Enabling user interfaces, such as menu items, that provide the Open In capability.</p> <p>What happens when the app initiates Open In?</p> <p>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. Target app error handling varies. For example, some apps display an error pop-up.</p>
Apps built with AppConnect 3.5 for iOS through the most recently released version as supported by MobileIron	<p>Open In policy enforced by: The AppConnect library, contained in the AppConnect app.</p> <p>App is responsible for:</p> <p>Disabling user interfaces, such a menu items, that provide the Open In capability.</p> <p>What if the app initiates Open In anyway?</p> <p>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:</p> <ul style="list-style-type: none"> Target app error handling varies. For example, some target apps 	<p>Open In policy enforced by: The AppConnect library, contained in the AppConnect app.</p> <p>App is responsible for:</p> <p>Enabling user interfaces, such as menu items, that provide the Open In capability.</p> <p>What happens when the app initiates Open In?</p> <p>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:</p> <ul style="list-style-type: none"> 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.

TABLE 30. OPEN IN BEHAVIOR IN WRAPPED APPS VERSUS SDK APPS (CONT.)

	Open In Not allowed	Open In is allowed only to AppConnect apps or Open In only to apps on whitelist
	display an error pop-up. <ul style="list-style-type: none"> 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, see [Putting 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 31. OPEN IN POLICY AND iOS NATIVE EMAIL

Device user action	Open In Not Allowed	Open In is allowed only to AppConnect apps	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 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 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 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 in the whitelist, put **both** of these bundle IDs in the whitelist:

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

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.
- An AppConnect app built with AppConnect 3.0 for iOS SDK or earlier is **not** impacted by the Open In policy. Regardless of how you set the policy, AirDrop is allowed.

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.
Only AppConnect apps allowed	The host app can use only extensions provided by AppConnect apps 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 global policy or AppConnect container policy. 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

In the AppConnect global policy, you can specify that you want to authorize AppConnect apps that do not have an AppConnect container policy. When an app does not have an AppConnect container policy, the data loss prevention (DLP) settings in the AppConnect global policy, including the Open In setting are applied to the app.

If you do not want the app to have the same Open In setting as the AppConnect global policy, you have two choices to override the setting:

- Create an AppConnect container policy for the app and label it appropriately. In it, specify the Open In and other DLP settings that you want for the app. These DLP settings override the settings in the AppConnect global policy.
- 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 to all apps.

In both cases, make sure you understand the potential impact of leaking secure data. This leak can happen:

- directly by an app for which you have configured one of the override choices.
- indirectly by an app that shares a document with an app for which you have configured one of the override choices.

For the indirect case, consider this scenario:

- The AppConnect global policy authorizes AppConnect apps that do not have an AppConnect container policy.
- The AppConnect global policy specifies Open In only to AppConnect apps.
- AppConnect App 1 and AppConnect App 2 do not have AppConnect container policies.
- You add the key `MI_AC_DISABLE_OPEN_IN_ENFORCEMENT` with the value YES to the AppConnect app configuration of App 2.



In this case, App 1 can share a document with App 2, and App 2 can share the document with *non-AppConnect apps*. Therefore, App 1 can indirectly share secure data with unsecured apps.

Open From data loss prevention policy

You select the Open From data loss prevention policy for an AppConnect app:

- MobileIron Core: in the AppConnect global policy or the AppConnect container policy.

The Open From behavior is the same for AppConnect wrapped or SDK apps. AppConnect version 4.3 through the latest version as supported by MobileIron is required for the Open From feature.

Related topics

- [Configuring the AppConnect global policy](#)
- [Configuring AppConnect container policies](#)

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:

- For MobileIron Core deployments, on the AppConnect app 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 32. 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 not allowed	Key is not applicable. Custom keyboard use is 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 is not 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
 - be running Mobile@Work 9.8 for iOS 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: <ul style="list-style-type: none"> 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 the default if you do not include the key.

Related topics

[Configuring an AppConnect app 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:

- In **Settings > System Settings > Additional Products > Licensed Products**, you disable AppConnect for third-party and in-house apps.

NOTE: If you disable Web@Work, Web@Work is retired. If you disable Docs@Work, the Docs@Work app is retired.

- You disable AppConnect in the AppConnect global policy for the device.
- You set the AppConnect global policy for the device to inactive.
- The device's AppConnect global policy does not have "Apps without an AppConnect container policy" checked, and you remove the app's AppConnect container policy from the device. To remove the policy from the device, you can delete it, or remove the device's label from it.
- The device has not completed an AppConnect check-in in the number of days specified in **Wipe AppConnect Data After** in the device's **AppConnect global policy** in Core or in the **AppConnect Device** configuration for iOS in Cloud.
- You retire the device.
- You quarantine the device due to a compliance action.
- The MobileIron UEM client is not present on the device, or present but not registered to MobileIron UEM.
- 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 checks a device for compliance with its security policy each time the device checks in. 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, 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 . However, also continues to enforce compliance actions when the device is connected.

Therefore, device-initiated compliance (local compliance) for jailbreak detection means that you can protect corporate data on devices even when the devices are off-line.

Typically, you configure a compliance action for jailbroken devices that occurs locally on the device, such as making AppConnect apps unauthorized or retired.

Compliance actions for device-initiated jailbreak detection

When Mobile@Work detects that the device is jailbroken, it can take one of the following actions:

- Alert the device user with a banner or notification.
- Block AppConnect apps.
The device user becomes unauthorized to use AppConnect apps.
- Retire AppConnect apps.
The device user becomes unauthorized to use AppConnect apps and the apps' secure data is deleted.

Configuring device-initiated compliance for jailbreak detection

The following is an overview of the steps for creating a device-initiated compliance for jailbreak detection.

1. [Creating a compliance action](#).
2. [Specifying the compliance action in the security policy](#).

Creating a compliance action

Create a compliance action in the MobileIron Core Admin Portal.

Procedure

1. Go to **Policies & Configs > Compliance Actions**.
2. Click **Add+**.



FIGURE 7. COMPLIANCE ACTIONS

Add Compliance Action

Select the actions that will be performed when devices are out-of-compliance.

Name


▼ **ALERT**

☐ Send a compliance notification or alert to the user

▼ **BLOCK ACCESS**

☐ Block email access and AppConnect apps

▼ **QUARANTINE**

 For Android for Work devices, all Android for Work apps and functionality will be hidden.


☐ Quarantine the device

☐ Remove All Configurations

☐ Do not remove Wi-Fi settings for Wi-Fi only devices (iOS and Android only)

☒ Do not remove Wi-Fi settings for all devices (iOS and Android only)

☐ Remove iBooks content, managed apps, and block new app downloads

☐ Enforce Compliance Actions Locally on Devices 

Cancel Save

3. Enter a name for the compliance action.
4. Select **Send a compliance notification or alert to the user** if you want the device user to receive a notification.
5. Select **Block email access and AppConnect apps** if you want the device user to become unauthorized to use AppConnect apps when the device becomes non-compliant.

NOTE: If the device is on-line, this selection also restricts access to email via ActiveSync if you are using a Standalone Sentry for email access. It also blocks AppTunnel tunnels that AppConnect apps and iOS managed apps use.

6. Select **Quarantine the device** if you want to retire AppConnect apps (blocks access to the apps and wipes their data) when the device becomes non-compliant.

NOTE: If the device is on-line, this selection also blocks AppTunnel tunnels that AppConnect apps and iOS managed apps use.

7. Select **Enforce compliance actions locally on devices**.

NOTE: The compliance actions are enforced locally on the device only if the security violation is that the device is jailbroken. Compliance actions for other security violations require interaction with MobileIron Core.

8. Click **Save**.

Related topics

“Compliance actions policy violations” in the *MobileIron Core Device Management Guide* for iOS.



Specifying the compliance action in the security policy

Specify a compliance action in the security policy.

Procedure

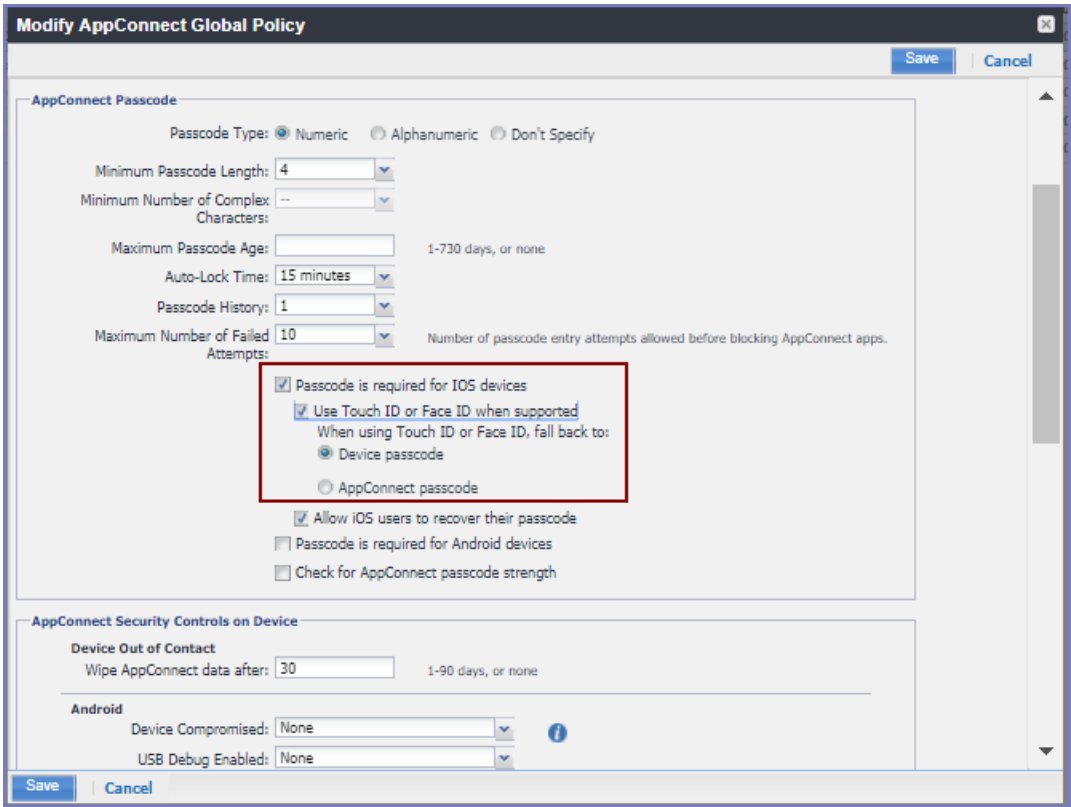
1. Go to **Policies & Configs > Policies**.
2. Select a security policy.
3. Click **Edit**.
4. In the section **Access Control**, in **For iOS devices**, select **When a compromised iOS device is detected**.
5. In the drop-down, select the compliance action that you created.
6. Click **Save**.

Touch ID or Face ID for accessing secure apps

Two options for accessing secure apps using Touch ID or Face ID are available on the AppConnect global policy:

- **Touch ID or Face ID with fallback to device passcode**
Device passcode, the default option, is the option used by almost all customers interested in the convenience of Touch ID for accessing AppConnect apps.
This 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.
When you use this option, a strong device passcode is required for Touch ID or Face ID to provide security for AppConnect apps. Make sure the security policy for the applicable devices requires a strong device passcode. Most customers typically choose this option.
- **Touch ID or Face ID with fallback to AppConnect passcode**
With this option, when the auto-lock time for AppConnect apps expires, the device user uses Touch ID or Face ID rather than the AppConnect passcode to re-access AppConnect apps. If entering the Touch ID or Face ID fails, the user enters (falls back to) the AppConnect passcode to access secure apps. The device user also uses the AppConnect passcode for other situations requiring AppConnect authentication such as the first time an AppConnect app is launched or when the user logs out of secure apps in Mobile@Work. Therefore, with this option, the user must create both a device passcode and an AppConnect passcode. The AppConnect passcode is necessary for all AppConnect authentications except the auto-lock time expiry. The device passcode is necessary because iOS requires it for Touch ID or Face ID to be available. However, the device passcode does not have to be strong.
Use this option only if you have a compelling reason to not require your device users to have a strong device passcode. A weak device passcode creates a security risk. The risk is that an unauthorized user can guess the device passcode, create a fingerprint or Face ID, and then access AppConnect apps.
One situation for choosing this option is if you are required by law to **not** require a strong device passcode on employee-owned devices. If you have such a requirement, but still want to give device users the convenience of using Touch ID or Face ID to access AppConnect apps, choose this option.

FIGURE 8. OPTIONS FOR USING TOUCH ID OR FACE ID



Related topics

- [Comparison of the two Touch ID/Face ID options](#)
- [Security versus convenience of passcode and Touch ID/Face ID options](#)
- [Touch ID or Face ID with fallback to device passcode](#)
- [Touch ID or Face ID with fallback to AppConnect passcode](#)

Comparison of the two Touch ID/Face ID options

The following table compares the Touch ID or Face D options.



TABLE 33. COMPARISON OF TOUCH ID/FACE ID OPTIONS

	When using Touch ID or Face ID, fall back to Device Passcode	When using Touch ID or Face ID, fall back to AppConnect Passcode
Differences between the two options		
When prompted for the Touch ID or Face ID, if the user fails to provide an accepted entry, the user is prompted for:	the device passcode	the AppConnect passcode NOTE: After too many failures, as controlled by iOS, iOS prompts for the device passcode.
Switches to Mobile@Work are eliminated in some cases when using Touch ID or Face ID.	Yes	No
Require a strong device passcode in the security policy.	Yes	No
The user can use Touch ID or Face ID <i>whenever</i> authenticating to secure apps,	Yes	No
The user can use Touch ID or Face ID <i>only</i> when authenticating to secure apps due to the AppConnect auto-lock time expiring.	No	Yes
The user can choose whether to use Touch ID or Face ID for future authentications to secure apps.	Yes	Yes, but the choice only applies to future authentications to secure apps due to the AppConnect auto-lock time expiring
When prompted for the Touch ID or Face ID, if the user taps Cancel :	Access to AppConnect apps is denied.	The user is prompted to enter the AppConnect passcode.
Similarities between the two options		
The user must have set a device passcode and added a Touch ID or Face ID on the device.	Yes	Yes
If the device has no Touch ID or Face ID hardware, the user authenticates to AppConnect apps using an AppConnect passcode.	Yes	Yes
If the device has no fingerprint or Face ID,	Yes	Yes

TABLE 33. COMPARISON OF TOUCH ID/FACE ID OPTIONS (CONT.)

	When using Touch ID or Face ID, fall back to Device Passcode	When using Touch ID or Face ID, fall back to AppConnect Passcode
or does not have the device passcode enabled, the user authenticates to AppConnect apps using an AppConnect passcode.		
If the user chooses not to use Touch ID or Face ID, the user uses an AppConnect passcode to access secure apps	Yes	Yes
In Mobile@Work in Settings > Secure Apps > Authentication , the user can at anytime change his choice about whether to use Touch ID/Face ID to access secure apps.	Yes	Yes

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

Passcode and Touch ID/Face ID configuration on MobileIron Core	Security of AppConnect apps	Convenience for device user
Require both: <ul style="list-style-type: none"> 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 the device is not encrypted unless the app uses the secure file I/O provided in the AppConnect for iOS SDK.	Convenient for accessing the device but inconvenient for accessing AppConnect apps.

TABLE 34. SECURITY VS DEVICE USER CONVENIENCE OF PASSCODE AND TOUCH ID/FACE ID OPTIONS (CONT.)

Passcode and Touch ID/Face ID configuration on MobileIron Core	Security of AppConnect apps	Convenience for device user
<ul style="list-style-type: none"> Require only a device passcode and Use Touch ID or Face ID with fallback to device passcode 	<p>High</p> <p>NOTE: Other device users who have added fingerprints or Face IDs, such as family members, can also access AppConnect apps.</p>	Very convenient for accessing both the device and AppConnect apps.
<ul style="list-style-type: none"> Require an AppConnect passcode and Use Touch ID or Face ID with fallback to AppConnect passcode 	<p>Lower</p> <p>NOTE: Although this option is intended for customers who cannot require a strong device passcode, the device user must create a device passcode to use Touch ID or Face ID. However, the user can choose a weak device passcode. A weak device passcode creates a security risk. The risk is that an unauthorized user can guess the device passcode, create a fingerprint or Face ID, and then access AppConnect apps.</p>	Very convenient for accessing both the device and AppConnect apps.
No passcodes required	<p>Lowest</p> <p>Note The Following:</p> <ul style="list-style-type: none"> 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.

Related topics

For more information about the use of the device passcode and AppConnect passcode in encrypting AppConnect-related data and data that the app saves, see:

- [The AppConnect passcode](#)
- [Data encryption for secure apps for iOS](#)

Touch ID or Face ID with fallback to device passcode

Touch ID or Face ID with fallback to device passcode allows the device user to enter his Touch ID (fingerprint) or Face ID instead of an AppConnect passcode to access secure apps. If entering the Touch ID/Face ID fails, the user enters (falls back to) the device passcode to access secure apps. When you use this option, a strong device passcode is required for Touch ID/Face ID to provide security for AppConnect apps. Make sure the security policy for the applicable devices requires a strong device passcode. Most customers who choose a Touch ID/Face ID option typically choose Touch ID or Face ID with fallback to device passcode.

The user can use his Touch ID or Face ID to access secure apps if:

- You configure the Touch ID or Face ID option on the MobileIron Core Admin Portal in the AppConnect global policy.
- You require a strong device passcode on devices.
- The device has Touch ID or Face ID capability.
- The device user has enabled Touch ID or Face ID on the device *and* the device passcode on the device.

When all of the above bullets are true, Mobile@Work gives the user the choice whether to use Touch ID/Face ID or use an AppConnect passcode to access AppConnect apps

Switches to Mobile@Work eliminated with Touch ID or Face ID with fallback to device passcode

AppConnect apps prompt for the Touch ID or Face ID themselves, without switching to Mobile@Work, in the following cases:

- The device user launched or switched to an AppConnect app after the auto-lock timeout expired.
- The auto-lock timeout expired while the device user was running an AppConnect app.

When these apps prompt for the Touch ID or Face ID, the prompt does not cover the entire screen. The apps, therefore, make sure sensitive data on the app's screen is not readable until the device user enters the Touch ID or Face ID. The following table summarizes this behavior:



TABLE 35. BEHAVIOR OF WRAPPED VERSUS SDK APPS WHEN DISPLAYING TOUCH ID/FACE ID PROMPT

Type of app	Behavior when displaying Touch ID/Face ID prompt
Wrapped apps	Blurs the visible parts of the screen.
SDK apps	Hides sensitive data according to the app's own requirements and user interface strategies

Note that situations other than the auto-lock timeout still cause the app to switch to Mobile@Work. For example, some of these situations are:

- the first time an app is launched.
- when the app check-in interval expires while an AppConnect app is running. Mobile@Work gets AppConnect policy updates for all the AppConnect apps, and then control switches back to the app that was running.
- after the device is powered on and the device user first launches an AppConnect app.
- after the device user logs out of secure apps in Mobile@Work, and then relaunched an AppConnect app.
- The device user relaunched an app after the user or iOS terminated it.

However, these other reasons for switching are less frequent than the switch due to the auto-lock timeout.

Improved user experience

Using Touch ID or Face ID with fallback to device passcode to access secure apps improves the user experience as follows:

- The user no longer needs to create, remember, and enter a secure apps passcode to access secure apps.
- Switching to Mobile@Work due to the auto-lock timeout is eliminated. Instead the app itself prompts for the Touch ID or Face ID, resulting in a simpler experience for the user. This switch to Mobile@Work is eliminated for apps as described in [Switches to Mobile@Work eliminated with Touch ID or Face ID with fallback to device passcode](#).
- Mobile@Work gives the user the choice to use Touch ID/Face ID or a secure apps passcode to access secure apps. 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 or Face ID to access the device. Although all the users can access the device with Touch ID/Face ID, sometimes only one of those users should be allowed to access AppConnect apps. That user can choose to use the AppConnect passcode instead of Touch ID/Face ID for accessing AppConnect apps. This feature therefore ensures that only an appropriate device user accesses AppConnect apps.

Device passcode requirements

A strong device passcode is required for Touch ID/Face ID to provide security for AppConnect apps. Make sure the security policy for the applicable devices requires a strong device passcode.



Procedure

1. Go to **Policies & Configs > Policies** in the MobileIron Core Admin Portal.
2. Select the security policy for the applicable devices.
3. Click **Edit**.
4. In the **Password** section, select **Mandatory**.
5. Select options for a strong password.
These options include **Alphanumeric**, **Minimum Password Length**, and **Minimum Number of Complex Characters**.
6. Click **Save**.

Device User impact in Mobile@Work

Mobile@Work gives the device user the choice to use Touch ID/Face ID or to use the AppConnect passcode for accessing secure apps.

Mobile@Work prompts the device user to make the choice when:

- You have selected the option to use Touch ID or Face ID with fallback to device passcode on the AppConnect global policy.
- The device user has enabled both Touch ID / Face ID and the device passcode on the device in **Settings > Touch ID & Passcode**.
- 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 Mobile@Work home screen

If the device user chooses Touch ID/Face ID, he will use Touch ID/Face ID for all further authentications to secure apps. If the device user does not choose Touch ID/Face ID, he will use the AppConnect passcode for all further authentications to secure apps.

NOTE: The device user can change the authentication method using **Settings > Secure Apps > Authentication** in Mobile@Work.

Related topics

[Secure apps on iOS Devices - User Perspective](#)

Configuring Touch ID or Face ID with fallback to device passcode

First, configure the AppConnect global policy to use Touch ID or Face ID with fallback to device passcode for accessing secure apps.

1. On the Admin Portal, go to **Policies & Configs > Policies**.
2. Select the appropriate AppConnect global policy.
3. Click **Edit**.
The AppConnect global policy displays.



4. Select **Passcode is required for iOS devices**.
5. Select **Use Touch ID or Face ID when supported**.

The option **When using Touch ID or Face ID, fall back to:** appears:

Modify AppConnect Global Policy

Save Cancel

AppConnect Passcode

Passcode Type: ☒ Numeric ☐ Alphanumeric ☐ Don't Specify

Minimum Passcode Length: 4

Minimum Number of Complex Characters: --

Maximum Passcode Age: 1-730 days, or none

Auto-Lock Time: 15 minutes

Passcode History: 1

Maximum Number of Failed Attempts: 10 Number of passcode entry attempts allowed before blocking AppConnect apps.

☒ Passcode is required for iOS devices

☒ Use Touch ID or Face ID when supported

When using Touch ID or Face ID, fall back to:

☒ Device passcode

☐ AppConnect passcode

☒ Allow iOS users to recover their passcode

☐ Passcode is required for Android devices

☐ Check for AppConnect passcode strength

AppConnect Security Controls on Device

Device Out of Contact

Wipe AppConnect data after: 30 1-90 days, or none

Android

Device Compromised: None

USB Debug Enabled: None

Save Cancel

6. Select **Device passcode**.
7. Click **Save**.

Also, if you are using an iOS restrictions setting, make sure that it allows Touch ID to unlock a device. If you are not using an iOS restrictions setting, Touch ID is allowed. You can skip the following steps.

1. On the Admin Portal, go to **Policies & Configs > Configurations**.
 2. Select the appropriate iOS restrictions setting.
 3. Click **Edit**.
- The iOS restrictions setting displays.
4. Make sure that **Allow Touch ID to unlock device** is selected.
 5. Click **Save**.

Less common device user scenarios

These scenarios describe the user experience in less common situations when using Touch ID.



TABLE 36. LESS COMMON DEVICE USER SCENARIOS INVOLVING TOUCH ID

On the AppConnect global policy	On the device	Behavior
Touch ID enabled	Fingerprint available Device passcode turned off	Consider the case when the device user had accessed secure apps using Touch ID before turning off the device passcode. When the device user attempts to access secure apps, the user is prompted to turn on the device passcode. Now consider the case when the device user had never accessed secure apps before turning off the device passcode. The device user will be prompted to create a secure apps passcode.
Touch ID enabled	No fingerprint available	Regardless whether the device passcode is turned on or off on the device, the device user is prompted to create a secure apps passcode when first trying to access secure apps. The device user then uses that secure apps passcode for all secure apps access. If the device user later adds a fingerprint and turns on the device passcode, he continues to use the secure apps passcode.
Touch ID enabled	Fingerprint available, then deleted. Device passcode turned on.	Consider the case when the device user deletes the fingerprint after accessing secure apps with Touch ID. The device user will use the device passcode to access secure apps.
Touch ID enabled	Fingerprint available, then deleted. Device passcode turned on, then turned off.	Consider the case when the device user deletes the fingerprint and turns off the device passcode <i>after</i> accessing secure apps with Touch ID. The device user must turn on the device passcode, and optionally add a fingerprint, to access secure apps.
Touch ID enabled then disabled	Fingerprint available. Device passcode turned on.	Consider the case when you disable Touch ID on the AppConnect global policy after the device user has accessed secure apps using Touch ID. The next time the device user accesses a secure app, he is prompted to create a secure apps passcode.

TABLE 36. LESS COMMON DEVICE USER SCENARIOS INVOLVING TOUCH ID (CONT.)

On the AppConnect global policy	On the device	Behavior
Touch ID enabled, then passcode requirement removed	Fingerprint available. Device passcode turned on.	Consider the case when you remove the passcode requirement on the AppConnect global policy after device user accessed secure apps with Touch ID. The next time the user accesses a secure app, he uses Touch ID one last time. After that, he can access secure apps without a passcode.
Touch ID not enabled, then enabled	Fingerprint available Device passcode turned on	Consider the case when you select the Touch ID option on the AppConnect global policy after the device user accessed secure apps using a secure apps passcode. The device user continues to use the secure apps passcode.
Touch ID enabled	Multiple fingerprints available Device passcode turned on	Any fingerprint available on the device can be used to access secure apps. A device user often chooses to use a secure apps passcode instead of Touch ID in this case. Using a secure apps passcode ensures other users of the device do not have access to secure apps.

Touch ID or Face ID with fallback to AppConnect passcode

With this option, when the auto-lock time for AppConnect apps expires, the device user uses Touch ID or Face ID rather than the AppConnect passcode to re-access AppConnect apps. If entering the fingerprint or Face ID fails, the user enters (falls back to) the AppConnect passcode to access secure apps. The device user also uses the AppConnect passcode for other situations requiring AppConnect authentication such as the first time an AppConnect app is launched or when the user logs out of secure apps in Mobile@Work.

Therefore, with this option, the user must create both a device passcode and an AppConnect passcode. The AppConnect passcode is necessary for all AppConnect authentications except the auto-lock time expiry. The device passcode is necessary because iOS requires it for Touch ID to be available. However, the device passcode does not have to be strong.

Use this option only if you have a compelling reason to not require your device users to have a strong device passcode. A weak device passcode creates a security risk. The risk is that an unauthorized user can guess the device passcode, create a fingerprint or Face ID, and then access AppConnect apps.

One situation for choosing this option is if you are required by law to **not** require a strong device passcode on employee-owned devices. If you have such a requirement, but still want to give device users the convenience of using Touch ID to access AppConnect apps, choose this option.

Improved user experience

The user experience is improved similarly to when using the option to use Touch ID or Face ID with fallback to device passcode. The improved user experience includes:

- Improved device user convenience compared with always entering the AppConnect passcode
When using this option, device users can use Touch ID or Face ID to access AppConnect apps when the auto-lock time expires. For other situations such as when Mobile@Work had been terminated or when a user used Mobile@Work to log out of secure apps, the user still must enter an AppConnect passcode. However, since the auto-lock time expiry is the most frequent situation that requires AppConnect authentication, the device user's experience is improved.
- Device user choice to use Touch ID/Face ID or to always use a secure apps passcode
Mobile@Work gives the user the choice to use Touch ID/Face ID or an AppConnect passcode to access secure apps when the auto-lock time expires. 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 or Face ID to unlock the device. Although all the users can access the device with Touch ID/Face ID, sometimes only one of those users should be allowed to access AppConnect apps. That user can choose to use the AppConnect passcode instead of Touch ID/Face ID for accessing AppConnect apps when the auto-lock time expires.

Configuring Touch ID or Face ID with fallback to AppConnect passcode

Do the following to configure the AppConnect global policy to use Touch ID for accessing secure apps with fallback to the AppConnect passcode.

Procedure

1. On the Admin Portal, go to **Policies & Configs > Policies**.
2. Select the appropriate AppConnect global policy.
3. Click **Edit**.
The AppConnect global policy displays.
4. Select **Passcode is required for iOS devices**.
5. Select **Use Touch ID or Face ID when supported**.
The option **When using Touch ID or Face ID, fall back to:** appears:

FIGURE 9. TOUCH ID OR FACE ID CONFIGURATION

Modify AppConnect Global Policy

AppConnect Passcode

Passcode Type: ☒ Numeric ☐ Alphanumeric ☐ Don't Specify

Minimum Passcode Length: 4

Minimum Number of Complex Characters: --

Maximum Passcode Age: 1-730 days, or none

Auto-Lock Time: 15 minutes

Passcode History: 1

Maximum Number of Failed Attempts: 10 (Number of passcode entry attempts allowed before blocking AppConnect apps.)

☒ Passcode is required for iOS devices

☒ Use Touch ID or Face ID when supported

When using Touch ID or Face ID, fall back to:

☐ Device passcode

☒ AppConnect passcode

☒ Allow iOS users to recover their passcode

☐ Passcode is required for Android devices

☐ Check for AppConnect passcode strength

AppConnect Security Controls on Device

Device Out of Contact

Wipe AppConnect data after: 30 (1-90 days, or none)

Android

Device Compromised: None

USB Debug Enabled: None

Save Cancel

6. Select **AppConnect passcode**.

7. Click **Save**.

Device User impact of Touch ID or Face ID with fallback to AppConnect passcode

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

1. After the device user registers with Mobile@Work, Mobile@Work prompts the device user to create an AppConnect passcode.
2. After creating the AppConnect passcode, Mobile@Work gives the user the option to use Touch ID/Face ID to access secure apps.
3. If the user chose the Touch ID/Face ID option, he can use Touch ID or Face ID when accessing secure apps after the auto-lock time has expired.
4. The device user can later use Mobile@Work settings to change his choice about using Touch ID/Face ID.

Related topics

[Touch ID or Face ID with fallback to AppConnect passcode -- device user perspective](#)

Certificate pinning for AppConnect apps

You can heighten security for the communication between AppConnect apps and enterprise servers or cloud services by using certificate pinning. Certificate pinning can protect an AppConnect app from man-in-the-middle attacks and rogue Certificate Authorities. This protection is possible because in the MobileIron Core Admin Portal, you configure a set of trusted certificates and domain names for the AppConnect app. When the app makes a connection request, the AppConnect library within the app makes sure that:

- the certificate presented by the server in response to the connection request is in the trusted set of certificates, or chains to a certificate in the trusted set.
- the domain in the app's URL request matches a domain name or domain wildcard that you specified for that certificate.

If either of these requirements is not met, the app's connection request fails.

Note The Following:

- Do not use certificate pinning with AppTunnel with HTTP/S tunneling or AppTunnel with TCP tunneling (Advanced AppTunnel).
- You can use certificate pinning with the feature [Certificate authentication from AppConnect apps to enterprise services](#). In this feature, the app sends a certificate to identify and authenticate the app user to the enterprise server or cloud service.

Certificate pinning for AppConnect apps requires:

- Apps built with AppConnect 4.1 for iOS through the most recently released version as supported by MobileIron.
- Mobile@Work 10.0.0 for iOS through the most recently released version as supported by MobileIron.

Related topics

- [About certificates used in certificate pinning](#)
- [About domains in certificate pinning](#)
- [Configuring certificate pinning](#)
- [Uploading the trusted certificates](#)
- [Creating a Client TLS configuration](#)
- [Modifying an AppConnect app configuration, Web@Work setting, or Docs@Work setting](#)
- [Viewing certificate pinning information in Mobile@Work](#)

About certificates used in certificate pinning

Part of configuring AppConnect apps for certificate pinning is uploading the trusted certificates to MobileIron Core so that Core can deliver them to AppConnect apps. Each certificate that you upload can be:



- a root CA certificate
- a intermediate CA certificate, which can include chained intermediate CAs with or without a root CA.
- a leaf certificate

NOTE: A trusted certificate used in certificate pinning cannot include a leaf certificate combined with an intermediate CA or root CA.

The certificate must be an X.509 certificate file (.cer, .crt, .pem, or .der) and encoded as binary DER or ASCII PEM.

About domains in certificate pinning

- [Certificate pinning domains for root and intermediate CA certificates](#)
- [Certificate pinning domains for leaf certificates](#)

Certificate pinning domains for root and intermediate CA certificates

When you specify root CA and intermediate CA certificates for certificate pinning, you also provide the domains of the target enterprise server or cloud service allowed in the AppConnect app's URL request. The AppConnect app's URL request must match one of these domains for the connection to succeed. You can provide specific domains, or domains using the wildcard character *. The following table provides examples.

TABLE 37. ALLOWED DOMAINS IN CERTIFICATE PINNING

Domain	Description
www.mycompany.com	App URL request must match exactly.
*.mycompany.com	App URL request must end in mycompany.com.
*	App URL request can be anything. When a domain is the wildcard character * by itself, the field Allow use of system CA certificates is automatically not selected and disabled.

Certificate pinning domains for leaf certificates

When you specify a leaf certificate for certificate pinning, Core extracts the domains from the certificate. The AppConnect app's URL request must match one of these domains for the connection to succeed. You cannot add to or modify the list of domains.

Core extracts the domains from these fields in the certificate:

- the CN (Common Name) in the Subject field
- the SAN (Subject Alternative Name) fields, if available



Configuring certificate pinning

The overall tasks to configure certificate pinning are:

1. [Uploading the trusted certificates](#)
2. [Creating a Client TLS configuration](#)
3. [Modifying an AppConnect app configuration, Web@Work setting, or Docs@Work setting](#)

Uploading the trusted certificates

Upload to MobileIron Core the certificate that the enterprise server will present to the app in response to a connection request. You upload the certificate into a certificate setting.

Procedure

1. In the Admin Portal, go to **Policies & Configs > Configurations**.
2. Click **Add New > Certificates**.
3. Fill in the entries:
Name: Enter brief text that identifies certificate setting.
Description: Enter additional text that clarifies the purpose of this certificate setting.
Click **Browse** to select the X.509 certificate file (.cer, .crt, .pem, or .der) to upload to Core. The certificate must be encoded as binary DER or ASCII PEM.
4. Click **Save**.

Do not apply a label to the certificate setting. You will apply a label to the AppConnect app configuration, Web@Work setting, or Docs@Work setting that refers to a Client TLS configuration that refers to this certificate setting.

Creating a Client TLS configuration

Create a Client TLS configuration that references the certificate setting.

Procedure

1. In the Admin Portal, go to **Policies & Configs > Configurations**.
2. Click **Add New > Client TLS**.
3. For **Name**, enter a name for the Client TLS configuration.
4. For **Description**, enter text that clarifies the purpose of this Client TLS configuration.
5. Click **Add+**.
A row displays in the **Trusted Certificates** table.
6. For **Name**, select from the dropdown the certificate setting you added in [Uploading the trusted certificates](#).



7. For the **Domain**:

- For intermediate and root CA certificates, enter one or more domains, as described in [Certificate pinning domains for root and intermediate CA certificates](#).
Enter a comma after a domain to make additional entries. Use the Enter key when you have finished entering domains.
- For leaf certificates, you cannot enter domains. Core automatically extracts the domains from the certificate and displays them.

8. Select **Allow use of system CA certificates** to allow the app to also accept system CA certificates if presented by the enterprise service.

System CA certificates are the certificates in the device's trusted certificate store.

NOTE: This option is automatically not selected and disabled when a domain you specify is the wildcard character * by itself.

9. Click **Save**.

The Admin Portal does not allow you to apply a label to the Client TLS configuration. You will apply a label to the AppConnect app configuration, Web@Work setting, or Docs@Work setting that refers to the Client TLS configuration.

NOTE: You cannot delete a certificate setting if it is referenced from a Client TLS configuration.

Modifying an AppConnect app configuration, Web@Work setting, or Docs@Work setting

Do the following tasks to reference the Client TLS configuration from the AppConnect app configuration for the AppConnect app.

- [Creating an AppConnect app configuration for the app if one does not already exist](#)
- [Configuring the Client TLS configuration in the AppConnect app configuration](#)

NOTE: For Web@Work and Docs@Work, follow similar steps in a Web@Work setting and Docs@Work setting. Just as you do in an AppConnect app configuration, you select **Enable Client TLS Configuration**, and select the appropriate Client TLS configuration from the dropdown options.

Creating an AppConnect app configuration for the app if one does not already exist

Do the following if an AppConnect app configuration does not exist for the app.

Procedure

1. In the Admin Portal, go to **Policies & Configs > Configurations**.
2. Look for an **APPCONFIG** setting type for the app.
3. If found, continue to [Configuring the Client TLS configuration in the AppConnect app configuration](#).
4. If not found, click **Add New > AppConnect > App Configuration**.



5. For **Name**, enter a name for the AppConnect app configuration.
6. For **Description**, enter a description for the AppConnect app configuration.
7. For **Application**, enter the bundle ID of the app, or select the app from the dropdown list if it is in the App Catalog.
8. Click **Save**.

Apply the appropriate labels to the new AppConnect app configuration:

1. Select the AppConnect app configuration that you just created.
2. Select **Actions > Apply To Label**.
3. Select the appropriate labels.
4. Click **Apply**.
5. Click **OK**.

Configuring the Client TLS configuration in the AppConnect app configuration

Procedure

1. In the Admin Portal, go to **Policies & Configs > Configurations**.
2. Select the AppConnect app configuration for the app (the **Setting Type** is **APPCONFIG**).
3. Click **Edit**.
4. In the **Client TLS** section, select **Enable Client TLS Configuration**.
5. From the dropdown, select the Client TLS configuration that you created.
6. Click **Save**.

NOTE: You cannot delete a Client TLS configuration if it is referenced from an AppConnect app configuration.

Viewing certificate pinning information in Mobile@Work

You can use Mobile@Work to view certificate pinning information for an AppConnect app.

Procedure

1. Open Mobile@Work on the device.
2. Tap **Settings**.
3. Tap **Secure Apps**.
4. Tap the app of interest.
5. Tap **Client TLS**, which is near the bottom of the screen.
The client TLS screen displays the set of trusted certificates.
6. Tap a certificate.



A screen displays with the domains allowed for the certificate.

7. Tap **Certificate** to see the certificate's details, such as its expiration date.

NOTE: You can also view the certificate's details in Mobile@Work in **Settings > Secure Apps > Stored Certificates**.

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.

You configure which certificate to use, and when to use it, on MobileIron Core, described in [Setting up certificate authentication from an AppConnect app](#). When using this feature, you can also use [Certificate pinning for AppConnect apps](#). Certificate pinning ensures that the certificate presented by the enterprise service in response to the app's connection request is in a trusted set of certificates that you configure.

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

Setting up certificate authentication from an AppConnect app

To set up certificate authentication from an AppConnect app to an enterprise service, you configure the app's AppConnect app configuration on MobileIron Core. You add sets of two key-value pairs to the AppConnect app configuration. The two key-value pairs in each set specify:

- a certificate
- a URL matching rule

You can have any number of sets of these key-value pairs, but keep the number to a minimum to preserve Core performance.

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

Before you begin

Configure MobileIron Core with the certificate that the app will use to authenticate to the enterprise service. You specify the certificate in a Certificate Enrollment.

See “Certificate Enrollment settings” in the *MobileIron Core Device Management Guide* for iOS.

NOTE: It is not necessary to assign labels to the Certificate Enrollment setting to distribute certificates to the appropriate devices. You will configure and apply labels to an AppConnect app configuration that refers to the certificate enrollment setting. That action distributes the certificates to the appropriate devices.

Overview

1. [Creating an AppConnect app configuration for the app if one does not already exist.](#)
2. [Configuring the key-value pairs for the certificate and URL matching rule.](#)

Creating an AppConnect app configuration for the app if one does not already exist

Check if you already created an AppConnect app configuration for the app. If one does not exist, create an app configuration for the AppConnect app.



Procedure

1. Go to **Policies & Configs > Configurations**.
2. Look for an **APPCONFIG** setting type for the app.
3. If found, continue to [Configuring the key-value pairs for the certificate and URL matching rule](#).

If not found, create the AppConnect app configuration:

1. Click **Add New > AppConnect > App Configuration**.
2. For **Name**, enter a name for the AppConnect app configuration.
3. For **Description**, enter a description for the AppConnect app configuration.
4. For **Application**, enter the bundle ID of the app, or select the app from the dropdown list if it is in the App Catalog.
5. Click **Save**.
6. Select **More Actions > Apply to Label**.
7. Select the appropriate label and click **Apply**.

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

Configure key-value pairs for the certificate and URL matching rule in the AppConnect app configuration in the Core Admin Portal.

Procedure

1. Go to **Policies & Configs > Configurations**.
2. Select the AppConnect app configuration for the app (the **Setting Type** is **APPCONFIG**).
3. Click **Edit**.
4. Scroll down to **App-specific Configurations**.
5. Click **Add+**.
6. In the **KEY** field, enter `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.

7. In the **VALUE** field, select the Certificate Enrollment setting from the dropdown list.
8. Click **Add+**.
9. In the **KEY** field, enter `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

10. In the **VALUE** field, enter the URL that will use the certificate for authentication.

Example

```
*.mycompany.com/sales
myserver.mycompany.com/hr/benefits
```

11. Click **Save**.

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 `*`.

For 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 38. 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 <code><host></code>
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	<code><host></code> does not match
http://anotherserver.mycompany.com	No	HTTP, not HTTPS

TABLE 39. MATCHING RULE EXAMPLE: *.MYCOMPANY.COM/SALES

URL Request	Match?	Comments
https://myserver.mycompany.com/sales	Yes	<code><host></code> and <code><subpath></code> match.
https://myserver.mycompany.com/sales/west	Yes	<code><host></code> and <code><subpath></code> match
https://anotherserver.mycompany.com/sales	Yes	<code><host></code> and <code><subpath></code> match
https://myserver.mycompany.com/salesmeeting	No	Entire <code><subpath></code> must match.
https://myserver.mycompany.com	No	Missing required <code><subpath></code>
https://myserver.mycompany.com/s	No	Entire <code><subpath></code> 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 40. 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`.

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.

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 41. 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	<ul style="list-style-type: none"> • 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	To support tunneling in these apps, configure a key-value pair in the app’s



TABLE 41. IMPACT TO TUNNELING WHEN USING A GLOBAL HTTP PROXY (CONT.)

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
recently released version as supported by MobileIron	<p>AppConnect app configuration (or Web@Work setting or Docs@Work setting) as follows:</p> <ul style="list-style-type: none"> key name: <code>global_http_proxy_url</code> value: the URL of the PAC file, which you also enter in the Proxy PAC URL field of the global HTTP proxy policy. Example: <code>http://pac.myproxy.mycompany.com</code> <p>With this key-value pair:</p> <ul style="list-style-type: none"> 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 Summary

Some AppConnect features are configured using key-value pairs on the AppConnect app configuration for an app, or on the Web@Work setting, the Docs@Work setting, or for Android, on the Secure Apps Manager's AppConnect app configuration.

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 summarizes the AppConnect for Android key-value pairs used to configure various AppConnect features.

- [AppConnect Global policy key-value pairs](#)
- [AppConnect app configuration key-value pairs](#)
- [Secure Apps Manager app configuration key-value pairs](#)

For key-value pairs used to configure various AppConnect features for AppConnect apps for iOS, see [AppConnect for iOS key-value pairs](#)

AppConnect Global policy key-value pairs

The following table describes the key-value pairs that are configured in the AppConnect Global policy.



TABLE 42. APPCONNECT FOR ANDROID KEY-VALUE PAIRS

Key name	Description	Reference
MI_AC_LOGOUT_WHEN_SCREEN_OFF	When true , device users are locked out of AppConnect apps when the device screen is turned off.	Lock Android AppConnect apps when screen is off
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



AppConnect app configuration key-value pairs

The following table describes the key-value pairs that are configured in the app's AppConnect app configuration.

TABLE 43. APPCONNECT FOR ANDROID KEY-VALUE PAIRS

Key name	Description	Reference
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 AppTunnel with TCP tunneling for Android secure apps
MI_AC_DISABLE_ANALYTICS	When true , analytics about AppConnect apps are not collected.	Disabling 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 secure apps
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 secure 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 secure apps
<service_name>_CERT	Specifies the certificate enrollment setting for the certificate for authenticating to the enterprise server when using AppTunnel with TCP tunneling.	Configuring certificate authentication using AppTunnel with TCP tunneling for Android secure apps
ES_DEFAULT_CERT	Specifies the certificate enrollment setting for the default certificate for authenticating to the enterprise server when using AppTunnel with TCP tunneling.	Configuring certificate authentication using AppTunnel with TCP tunneling for Android secure apps

Secure Apps Manager app configuration key-value pairs

The following table describes the key-value pairs that are configured in the AppConnect app configuration for Secure Apps Manager.



TABLE 44. APPCONNECT FOR ANDROID KEY-VALUE PAIRS

Key name	Description	Reference
MI_DISABLE_CUSTOM_KEYBOARD	When true , the use of custom keyboards in AppConnect apps is disabled.	Custom keyboards in AppConnect apps
MI_CUSTOM_KEYBOARD_WHITELIST	Lists the package names of whitelisted keyboard apps when custom keyboards are disabled.	Custom keyboards in AppConnect apps
MI_AC_SHARE_CONTENT	When true , AppConnect apps can share content (text, images, or video) with non-AppConnect apps.	Sharing content from AppConnect for Android apps to non-AppConnect apps
MI_AC_ALLOW_OPEN	Specifies the domain and the non-AppConnect app in which to open the domain.	See App whitelist for format and examples of the supported values for the key-value pair.
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 <i>MobileIron AppConnect for Android App Developers Guide</i>

AppConnect for iOS key-value pairs

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

TABLE 45. 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 .	Configuring certificate authentication using AppTunnel with TCP tunneling for Android secure apps
MI_AC_LOG_	Specifies the string that the device user enters to activate the	Logging for



TABLE 45. APPCONNECT FOR iOS KEY-VALUE PAIRS (CONT.)

Key name	Description	Reference
LEVEL_CODE	verbose or debug log level.	AppConnect apps for iOS
MI_AC_ENABLE_LOGGING_TO_FILE	When Yes , an AppConnect app's logs are logged to files on the device.	Logging for AppConnect apps for iOS
MI_AC_WR_ENABLE_LOG_CAPTURE	When Yes , when emailing AppConnect-related log files from Mobile@Work, the logs of a wrapped app are emailed along with the logs of the AppConnect wrapper and the AppConnect librar	Logging for AppConnect apps for iOS
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 45. 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	<p>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.</p> <p>For example:</p> <p><code>www.somewebsite.com,somename.someotherwebsite.com</code></p> <p>Supported with apps built or wrapped with AppConnect 4.2.1 for iOS through the most recently released version as supported by MobileIron.</p>	

Related topics[AppConnect for Android key-value pairs](#)

Troubleshooting AppConnect and AppTunnel for Android

- [Logging for AppConnect apps for Android](#)
- [State and encryption mode of Android secure apps](#)
- [Status of AppConnect-related policies and configurations for an app](#)

Logging for AppConnect apps for Android

Log files can be emailed by using the **Send Log** option in Mobile@Work for Android. You can choose whether the log files are encrypted when they are provided to the email app. The choice affects the log files of the following:

- Mobile@Work for Android
- Secure Apps Manager
- AppConnect-enabled apps (including what the app logs and what the AppConnect wrapper around the app logs)

Encrypted log files can be decrypted only by MobileIron Technical Support. The security policy for a device contains the option for choosing whether the emailed log files are encrypted. The default setting is to **not** encrypt the files. Typically, you turn on encryption. However, if you want to troubleshoot issues with Mobile@Work, Secure Apps Manager, or AppConnect-enabled apps yourself, turn off device log encryption.

NOTE: Regardless of the device log encryption setting, the log files never include passwords, certificate content, license information, or other sensitive authentication data.

Turning on device log encryption on Android devices

1. Go to **Policies & Configs > Policies**.
2. Select the security policy for the appropriate devices.
3. Click **Edit**.



Modifying Security Policy

Attempts:
 Password History:
 Grace Period for Device Lock:

Data Encryption Platforms Supported

Device Encryption: ☐ On ☒ Off

Data Type: ☐ All ☐ Email ☐ PIM ☐ My Docs

File Types: ☐ All ☐ .doc ☐ .xls ☐ .pdf
☐ .txt ☐ Media files
☐ Other(s) (e.g., .cab, .ppt)

SD Card Encryption: ☐ On ☒ Off

Device Log Encryption: ☒ On ☐ Off

TLS

☐ Require strict TLS for Apps@Work (Android Only) i

Windows Platforms Supported

Firewall: ☒ On ☐ Off

Anti-Virus: ☒ On ☐ Off

Auto-Update: ☒ On ☐ Off

Access Control Platforms Supported

For All Platforms

Save | **Cancel**

4. In the **Data Encryption** section, for **Device Log Encryption**, select **On**.
5. Click **Save**.

State and encryption mode of Android secure apps

To see device details about the state and encryption mode of Android secure apps:

1. On the Admin Portal, go to **Devices & Users > Devices**.
2. Expand the device details panel of an Android device, by clicking the up arrow next to the checkbox.
3. Select the **Device Details** tab.
 The following details relate to AppConnect apps on the device:



TABLE 46. STATE AND ENCRYPTION MODE OF ANDROID SECURE APPS ON DEVICE DETAILS

Item	Description
Secure Apps Enabled	true if the AppConnect global policy for the device is enabled. That is, for the AppConnect field, Enabled is selected. Otherwise false .
Secure Apps Encryption Enabled	true if the device user has created a secure apps passcode. Otherwise, false . NOTE: This field is applicable only when you have selected Passcode is required for Android devices on the AppConnect global policy. When an AppConnect passcode is not required, encryption is always enabled, without depending on the device user to create a secure apps passcode.
Secure Apps Encryption Mode	Displays AES-256

Status of AppConnect-related policies and configurations for an app

Information is available on the Admin Portal about the status of AppConnect-related policies and configurations for AppConnect apps on a device.

To see the information:

1. On the Admin Portal, go to **Devices & Users > Devices**.
2. Expand the device details panel of an Android device, by clicking the up arrow next to the checkbox.
3. Select the **Apps** tab.
4. Select an AppConnect app.
Information displays about the policies and configurations on the device for the AppConnect app.



Troubleshooting AppConnect and AppTunnel for iOS

- [Logging for AppConnect apps for iOS](#)
- [Secure apps status display in Mobile@Work](#)
- [AppTunnel configuration troubleshooting display in Mobile@Work](#)
- [Status of AppConnect-related policies and configurations for an app](#)
- [Viewing certificates stored in Mobile@Work](#)
- [AppTunnel diagnostics in SDK-built apps](#)

Logging for AppConnect apps for iOS

- [Overview of logging for AppConnect apps for iOS](#)
- [Log levels](#)
- [How the log level appears in messages](#)
- [Log file details](#)
- [Log data collection overview](#)
- [Configuring logging for an AppConnect app](#)
- [Creating a new label](#)
- [Applying labels](#)
- [Log level configuration impact on the device](#)
- [Activating verbose or debug logging on the device](#)
- [Emailing log files from Mobile@Work](#)
- [Removing log level configuration when no longer needed](#)

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.

IMPORTANT: Do not modify the log level if it impacts many devices in your production configuration because the modification can impact Core performance. Make modifications only to configurations that impact only a few devices.

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 47. APPCONNECT LOG LEVELS

Log level	Description	Contains sensitive data?	Requires the user to enter the debug code?
Error	<p>Provides error, warning, and status messages.</p> <p>This level is the default. It is always turned on.</p> <p>Error messages are for events that block access to part or all of the app.</p> <p>Example: Corrupt or missing data</p> <p>Warning messages are for events that are suspicious, but not quite failures like errors.</p> <p>Example: Unexpected data that is ignored</p> <p>Status messages indicate major changes in the state of the app.</p> <p>Example: User successfully logged in</p>	No	No
Info	<p>Provides error, warning, and status messages, plus more information.</p> <p>Info messages indicate minor changes in the state of the app.</p> <p>Example: AppConnect app check-in times</p>	No	No
Verbose	<p>Provides error, warning, status, and info messages, plus more, possibly sensitive, information.</p> <p>Verbose messages provide more extensive information, possibly including sensitive details.</p> <p>Example: Server URLs</p>	Yes	Yes
Debug	<p>Provides error, warning, status, info, and verbose messages, plus further information, which is possibly sensitive.</p> <p>Debug messages have the most information, possibly including sensitive details.</p> <p>Example: URL request details</p>	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 48. 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	<p>[Error] [Warning] [Status] [Info] [Verbose] [Debug]</p> <p>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].</p>
AppConnect library contained in an AppConnect app	The app's name	<p>[AppConnect:Error] [AppConnect:Warning] [AppConnect:Status] [AppConnect:Info] [AppConnect:Verbose] [AppConnect:Debug]</p> <p>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].</p>
The AppConnect wrapper (only applicable for wrapped AppConnect apps)	The app's name	<p>[AppConnectWrapper:Error] [AppConnectWrapper:Warning] [AppConnectWrapper:Status] [AppConnectWrapper:Info] [AppConnectWrapper:Verbose] [AppConnectWrapper:Debug]</p> <p>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].</p>

Log file details

Details regarding the log files for each app are:

- 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:
 - a. It is renamed to appconnect.log.<timestamp>. Example: appconnect.log.2015-05-28 15:13:21
 - b. Logging begins in a new file named appconnect.log.
 - c. If 20 log files already exist, the oldest file is deleted.

Log data collection overview

To collect log data, you do the following high-level steps:

1. Create an AppConnect app configuration that specifies a non-default log level for the app and specifies logging to files, if desired.
You create a key-value pair that specifies one of four log levels. For the two highest log levels, verbose and debug, you create a key-value pair that is the debug code that activates logging.
You create another key-value pair to specify logging to files.
See [Configuring logging for an AppConnect app](#).
2. Create a new label that you apply to the new AppConnect app configuration and no more than a few devices.
See [Creating a new label](#).
3. Apply labels appropriately.
See [Applying labels](#).
4. If you chose one of the two highest log levels, ask the device user to turn on logging for the app on the device, and to enter the debug code.
See [Log level configuration impact on the device](#) and [Activating verbose or debug logging on the device](#).
5. Use Mobile@Work to email the log files.
See [Emailing log files from Mobile@Work](#)
6. Revert to the default log level.
See [Removing log level configuration when no longer needed](#)

Configuring logging for an AppConnect app

To configure 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, do the following:

1. In the Admin Portal, select **Policies & Configs > Configurations**
2. If the app does not already have an AppConnect app configuration, select **Add New > AppConnect > App Configuration**. Enter a name and description for the new app configuration and the app's bundle ID.
Continue to [Step 5](#).
3. If the app does have an AppConnect app configuration, select it.
4. Click **More Actions > Save As**.
A dialog box with a copy of the AppConnect app configuration displays.
5. In App-specific Configurations, click **Add+** to add a key-value pair.
6. Enter **MI_AC_LOG_LEVEL** in the key field.
The key name is case-sensitive.
7. Enter one of the following in the value field: error, info, verbose, or debug.
This value is not case-sensitive.
8. If you entered verbose or debug, click + to add another key-value pair.
9. Enter **MI_AC_LOG_LEVEL_CODE** in the key field.
The key name is case-sensitive.



10. 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.
 11. In App-specific Configurations, click **Add+** to add a key-value pair.
 12. Enter **MI_AC_ENABLE_LOGGING_TO_FILE** in the key field.
The key name is case-sensitive.
- NOTE: Apps built with AppConnect for iOS SDK versions prior to 2.3 or wrapped with AppConnect for iOS Wrapper versions prior to 2.5 do not support this key. The logs do not get written to files.
13. Enter **Yes** in the value field.
 14. Click **Save**.

Creating a new label

Create a new label for the new AppConnect app configuration.

1. In the Admin Portal, select **Devices & Users > Labels**.
2. Click **Add Label**.
3. Enter a name for the label, such as **AppConnect for iOS logging**.
4. For **Type**, select **Manual**.
5. Click **Save**.

Applying labels

Apply labels so that the AppConnect app configuration with the log level key-value pairs is applied to only a few devices.

IMPORTANT: Modifying an AppConnect app configuration for many devices in your production environment can impact Core performance. For this reason, creating a new label for only a few devices is necessary.

Apply labels as follows:

1. Apply the new label to the new AppConnect app configuration.
2. Apply the new label to a few test devices.
3. If you copied the AppConnect app configuration, make sure the test devices have only the new label, and not the label from the copied AppConnect app configuration.
4. If the app has an AppConnect container policy, apply the new label to the policy so that the test devices receive it.
5. Apply the new label to the AppConnect global policy for the devices, unless the devices use the default AppConnect global policy.
6. If the test devices depend on other configurations or policies, apply the new label to those configurations and policies, too.

To apply the new label to a configuration or policy:



1. Select the policy or configuration.
2. Select **More Actions > Apply To Label**.
3. Select the new label.
4. Click **Apply**.

To add the new label to a device:

1. Select the device.
2. Select **More Actions > Apply To Label**.
3. Select the labels to apply to the device.
4. Click **Apply**.

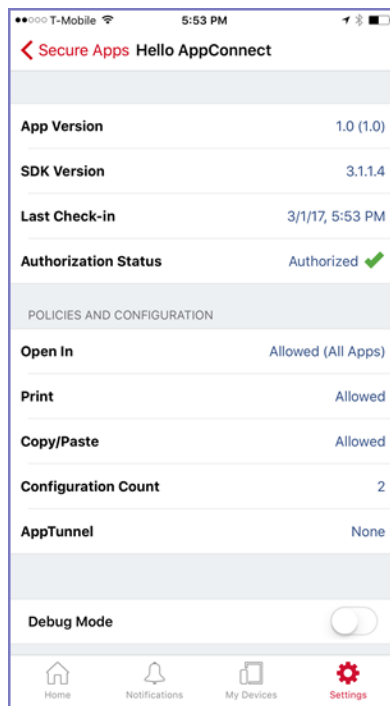
Log level configuration impact on the device

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 Mobile@Work.

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:

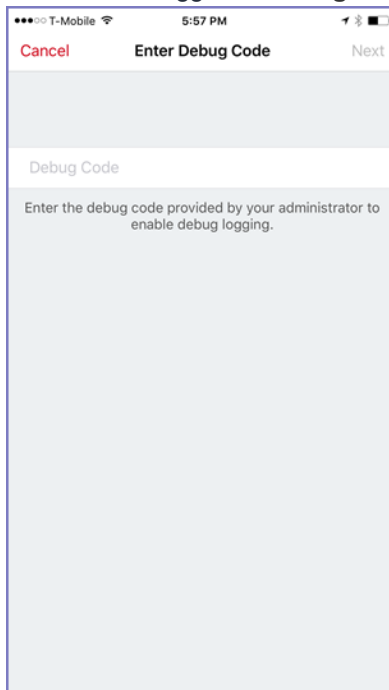


NOTE: The keys MI_AC_LOG_LEVEL and MI_AC_LOG_LEVEL_CODE are not included in the configuration count on an app's detailed status display.

Activating verbose or debug logging on the device

To activate verbose or debug level logging, instruct the device user to do the following:

1. Open Mobile@Work 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**.



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 Mobile@Work

Mobile@Work for iOS can send the log files to an email address of your choice. This feature requires:

- Mobile@Work 9.8 for iOS through the most recently released version as supported by MobileIron
- AppConnect apps using AppConnect 4.0 for iOS

Mobile@Work displays the option to send logs on the app's status details screen, available in Mobile@Work at **Settings > Secure Apps > <app name>**. The option is at the bottom of the screen with this text: **Send <app name> Logs**.

The option is displayed only for apps AppConnect apps using AppConnect 4.0 for iOS. However, the displayed 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.

Removing log level configuration when no longer needed

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 new AppConnect app configuration. This best practice ensures the app does not continue logging sensitive data unnecessarily.

Do the following:

1. In the Admin Portal, select **Policies & Configs > Configurations**
2. Select the app configuration for the app and click Edit.
3. In **App-specific Configurations**, click **X** to remove the key-value pairs.
4. Click Save.

Alternatively, you can also return to using the original AppConnect app configuration for the test devices. Do the following:

1. Remove the new test label from the devices and re-apply the original label.
2. Delete the new AppConnect app configuration.
3. Remove the new label from any configurations or policies that you applied it to.
4. Delete the new label.

Secure apps status display in Mobile@Work

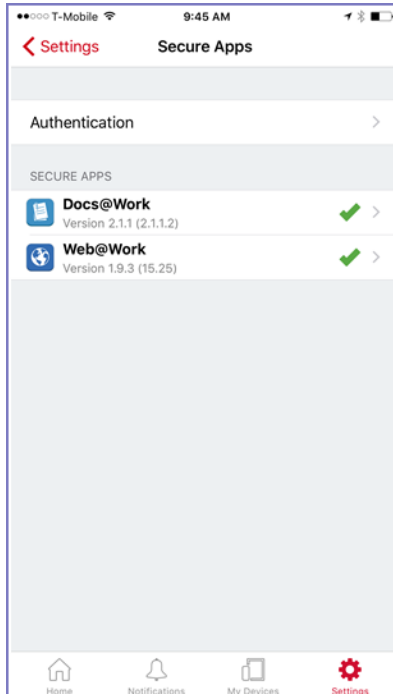
A secure apps status display in Mobile@Work provides detailed information about each secure app, allowing you to troubleshoot issues more easily.



Navigating to the secure apps status display

To see the secure apps status display:

1. Open Mobile@Work on the device.
2. Tap Settings.
3. Tap Secure Apps.



4. All installed secure apps that have been opened at least once appear under the heading “Secure Apps”. If no secure apps have been opened at least once, then this list does not appear.

The secure apps status display contents

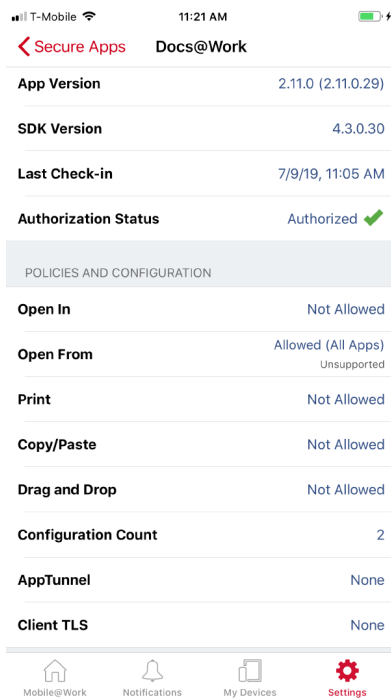
The secure apps status display shows the following information for each secure app:

- The icon of the secure app
- The name of the secure app
- The version number of the secure app
It is the short version number, followed by the long version number in parenthesis.
- An icon that indicates whether the app is authorized

Status details for a specific secure app

To see status details for one of the secure apps in the secure apps status display, tap the app’s entry.

FIGURE 10. STATUS DETAILS FOR A SECURE APP



The following table describes the status details for a secure app:

TABLE 49. STATUS FIELDS FOR A SECURE APP

Field	Description
App Version	The version number of the secure app. It is the short version number, followed by the long version number in parenthesis.
SDK Version	The version of the AppConnect for iOS SDK for apps built with the SDK. The AppConnect for iOS Wrapper version for wrapped apps. This version includes the SDK version used in the Wrapper.
Last Check-in	The date and time when Mobile@Work last fetched the AppConnect policies from MobileIron Core.
Authorization Status	Whether the device is authorized to use the app. Possible values are: <ul style="list-style-type: none"> • Authorized • Unauthorized • Retired
Policies and Configurations For more information, see AppConnect container policies and AppConnect app configuration .	
Open In	Whether Open In is allowed for the app in Core configuration. Possible values are:



TABLE 49. STATUS FIELDS FOR A SECURE APP (CONT.)

Field	Description
	<ul style="list-style-type: none"> • Not Allowed • Allowed (All Apps) • Allowed (Secure Apps Only) • Allowed (Whitelisted Apps)
Open From	<p>Whether Open From is allowed for the app in Core configuration. Possible values are:</p> <ul style="list-style-type: none"> • Not Allowed • Allowed (All Apps) • Allowed (Secure Apps Only) • Allowed (Whitelisted Apps)
Print	<p>Whether print capabilities are allowed for the app in Core configuration. Possible values are:</p> <ul style="list-style-type: none"> • Not Allowed • Allowed
Copy/Paste	<p>Whether the device user can copy from the app to other apps, as specified in Core configuration. Possible values are:</p> <ul style="list-style-type: none"> • Not Allowed • Allowed • Secure Apps



TABLE 49. STATUS FIELDS FOR A SECURE APP (CONT.)

Field	Description
Drag and Drop	Whether the device user can drag content from the app to other apps, as specified in Core configuration. Possible values are: <ul style="list-style-type: none"> • Not Allowed • Allowed • Secure Apps
Configuration Count	The number of key-value pairs that the Core sent to the app. This value corresponds to the number of key-value pairs in the AppConnect app configuration for the app. <p>Note The Following:</p> <ul style="list-style-type: none"> • If one of the key-value pairs in the AppConnect app configuration is a Certificate Enrollment or Certificate setting and the certificate is password-encoded, Core automatically sends another key-value pair for the password. The configuration count includes that key-value pair. • The keys that you use to turn on debug level logging for an AppConnect app are not included in the configuration count. These keys are MI_AC_LOG_LEVEL and MI_AC_LOG_LEVEL_CODE.
AppTunnel	Displays AppTunnel configuration information for the app that it received from MobileIron Core. This information helps troubleshoot your AppTunnel configuration when an app is not successfully tunneling to its app server. <p>Related topics</p> <p>AppTunnel configuration troubleshooting display in Mobile@Work.</p>

If an app has not applied a policy or configuration, the corresponding field in the display also indicates one of the following:

- **Pending**
The app has not yet applied the policy or configuration. The pending status shows until the next time the device user launches the app.
- **Unsupported**
The app does not support the policy or configuration.
- **Error**
The app had an error when applying the policy or configuration.

When you change the policies or configuration on MobileIron Core, Mobile@Work displays the updated status the next time it fetches the policies from Core. This action occurs when the next time any app checks in, or when a force device check-in occurs.



AppTunnel configuration troubleshooting display in Mobile@Work

Mobile@Work displays AppTunnel configuration information for the app that it received from MobileIron Core. This information helps troubleshoot your AppTunnel configuration when an app is not successfully tunneling URL requests to its app server. Check the display's fields to make sure that your AppTunnel configuration has been sent to the device and is what you intended.

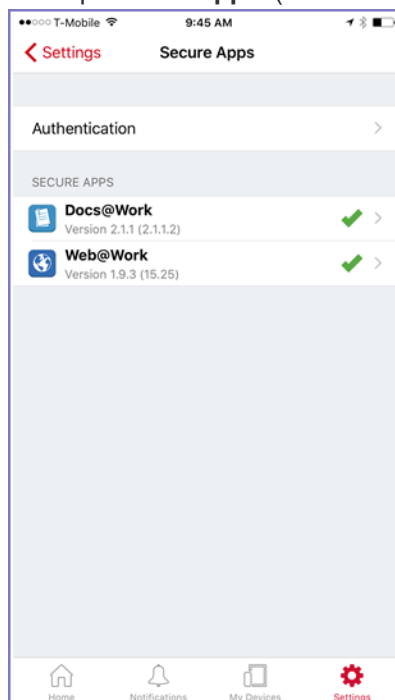
Some highlights of the displayed AppTunnel configuration are:

- Whether Mobile@Work has received the AppTunnel configuration from Core.
- Whether the client certificate, identifying the device user to the Standalone Sentry, has expired.
- The list of AppTunnel rules that indicate which URL requests should be tunneled.

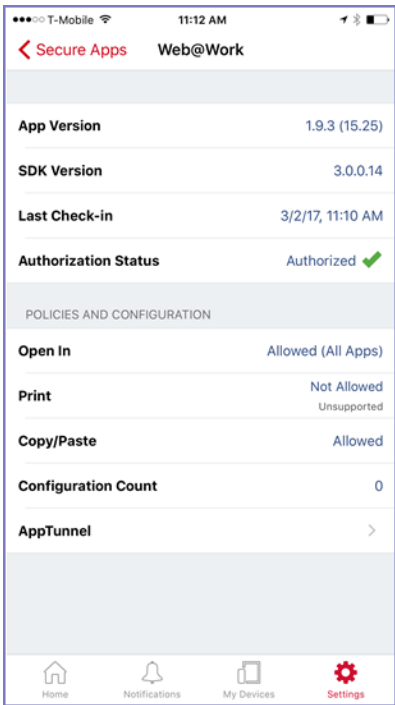
Navigating to the AppTunnel configuration troubleshooting display

Procedure

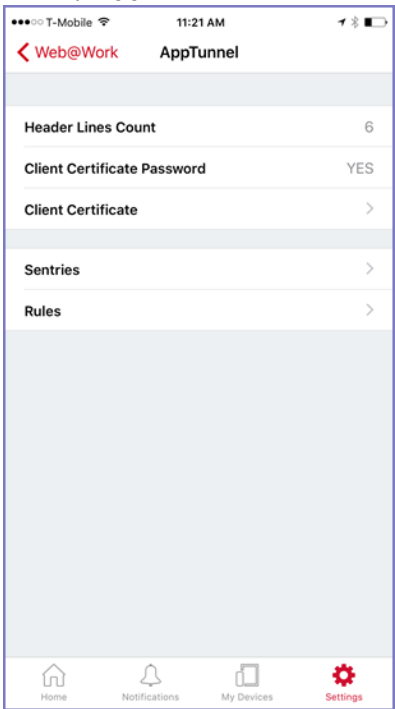
1. Open Mobile@Work on the device.
2. Tap **Settings**.
3. Tap **Secure Apps**. (Screenshots from Mobile@Work 9.1)



4. Tap the secure app you are interested in.



5. Tap **AppTunnel**.



NOTE: If **AppTunnel** is **None**, no AppTunnel configuration is available for the app on Mobile@Work. See [AppTunnel configuration troubleshooting checklist](#).

Troubleshooting with the AppTunnel configuration display fields

Use the AppTunnel configuration display if URL requests that you configured for AppTunnel are not being tunneled. Check the display's fields to make sure that your AppTunnel configuration has been sent to the device and is what you intended.

NOTE: Some screenshots are from Mobile@Work 9.1.

FIGURE 11. APPTUNNEL DISPLAY

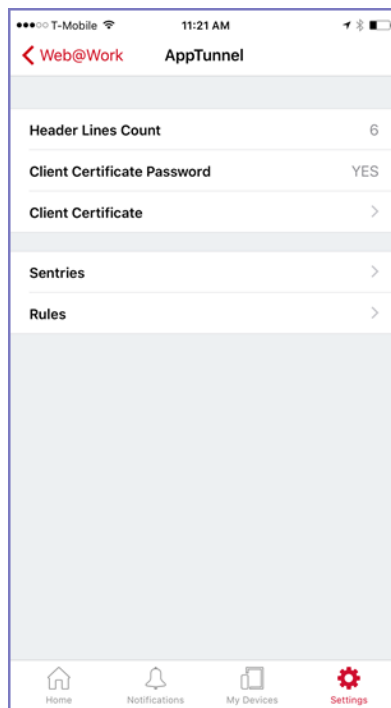


TABLE 50. APPTUNNEL DISPLAY FIELDS AND TROUBLESHOOTING ACTIONS

Field name	Description	Troubleshooting actions
Header Line Count	HTTP/S request header information for Sentry	If the value is zero, the device is not receiving the AppTunnel information from MobileIron Core. Verify your AppTunnel configuration as specified in AppTunnel configuration troubleshooting checklist . If the value is still zero, contact MobileIron Technical Support.
Client Certificate Password	Whether the client certificate that identifies the device user to Sentry is password-enabled.	When using a SCEP certificate, the value should be YES.
Client Certificate	Whether a valid client	Tap to see the identity's certificate information,

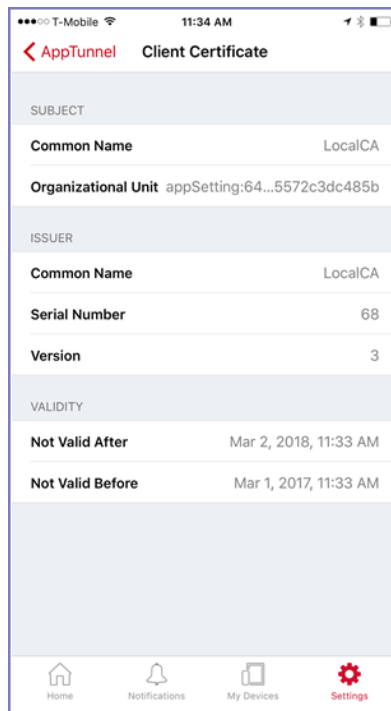
TABLE 50. APPTUNNEL DISPLAY FIELDS AND TROUBLESHOOTING ACTIONS (CONT.)

Field name	Description	Troubleshooting actions
	identity is available. This client identity is used to authenticate the app to the Sentry.	<p>including whether the certificate has expired.</p> <p>If the value is None, check the AppConnect app configuration for the app. Make sure you specified an identity certificate.</p> <p>In the Admin Portal for MobileIron Core:</p> <ol style="list-style-type: none"> 1. Go to Policies & Configs > Configurations. 2. Select the AppConnect app configuration for the app (Setting Type is APPCONFIG) and Click Edit. 3. If you have not created an AppConnect app configuration, select Add New > AppConnect > Configuration. 4. In the AppTunnel Rules section, in the Identity Certificate field, specify a valid client certificate. 5. Click Save. <p>Make sure you have applied the appropriate labels to the AppConnect app configuration.</p>
Sentries	The Sentries that are configured for AppTunnel for this app.	Tap to see the list of Sentries. Make sure they are what you expect this app to use for AppTunnel.
Rules	The AppTunnel rules configured on the app's AppConnect app configuration.	<p>If None, you have not configured AppTunnel rules on the AppConnect app configuration.</p> <p>In the Admin Portal for MobileIron Core:</p> <ol style="list-style-type: none"> 1. Go to Policies & Configs > Configurations. 2. Select the AppConnect app configuration for the app (Setting Type is APPCONFIG) and Click Edit. 3. If you have not created an AppConnect app configuration, select Add New > AppConnect > Configuration. 4. Edit the AppTunnel Rules section. 5. Click Save. <p>Make sure you have applied the appropriate labels to the AppConnect app configuration.</p>

Client Certificate display

Check the client certificate fields, including whether the certificate has expired.





NOTE: If necessary, tap on a field to view the entire string.

TABLE 51. CLIENT CERTIFICATE DISPLAY FIELDS AND TROUBLESHOOTING ACTIONS

Field name	Description	Troubleshooting actions
Subject		<p>You can compare the certificate values to the values on the Core Admin Portal:</p> <ol style="list-style-type: none">1. Go to Logs > Certificate Management.2. Select the certificate of interest for the user.3. Click View. <p>Related topics</p> <ul style="list-style-type: none">• Specifying a trusted root certificate in the Standalone Sentry• Specifying a valid client certificate in the AppConnect app configuration
Common Name		
Organizational Unit		
Issuer		
Common Name		
Serial Number		
Version		
Validity		
Not Valid After	Expiration date	Make sure that the certificate has not expired.
Not Valid Before	Initial date	Make sure that the certificate is valid.

Specifying a trusted root certificate in the Standalone Sentry

The client identity is issued from a Trusted Root Certificate. The Standalone Sentry must be configured with the Trusted Root Certificate for device authentication to the Sentry.

To configure Standalone Sentry with the Trusted Root Certificate, in the Admin Portal for MobileIron Core:

1. Go to **Services > Sentry**.
2. Select the Standalone Sentry.
3. Click **Edit**.
4. Make sure **Enable AppTunnel** is selected.
5. In the **Device Authentication Configuration** section, select **Identity Certificate**.
6. Click **Choose File** to navigate to and select the Trusted Root Certificate.
7. Click **Upload Certificate**.
8. Click **View Certificate** to verify the certificate.
9. Click **Save**.

Related topics

- “Configure device and server authentication on Standalone Sentry” in the AppConnect and AppTunnel Guide.
- “Device and server authentication support for Standalone Sentry” in the MobileIron Sentry Guide.

Specifying a valid client certificate in the AppConnect app configuration

If the client certificate is not valid, specify a valid client identity certificate in the AppConnect app configuration.

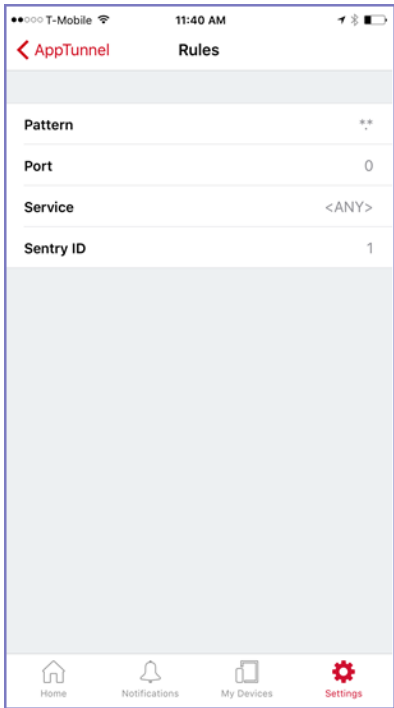
In the Admin Portal for MobileIron Core:

1. Go to **Policies & Configs > Configurations**.
2. Select the AppConnect app configuration for the app (**Setting Type** is **APPCONFIG**) and Click **Edit**.
3. If you have not created an AppConnect app configuration, select **Add New > AppConnect > Configuration**.
4. In the **AppTunnel Rules** section, in the **Identity Certificate** field, specify a valid client certificate.
5. Click **Save**.

Make sure you have applied the appropriate labels to the AppConnect app configuration.



Rules display



This display shows each AppTunnel rule configured on the app's AppConnect app configuration. The following table shows the display fields for each rule and the corresponding fields in the AppConnect app configuration:



TABLE 52. APPTUNNEL RULES DISPLAY FIELDS AND TROUBLESHOOTING ACTIONS

Field name	Description	Troubleshooting actions
Pattern	Corresponds to the URL Wildcard field of the AppConnect app configuration.	<p>Make sure the field contains the hostname that the app is trying to access. The pattern can contain the wildcard *.</p> <p>The app data is tunneled only if the hostname and port number in the app's request matches the Pattern field and Port field.</p> <p>Exception: For iOS apps using AppConnect releases prior to AppConnect for iOS SDK 2.5 and AppConnect for iOS Wrapper 2.7, only the request's hostname, not the port number, determines whether the app data is tunneled.</p>
Port	Corresponds to the Port field of the AppConnect app configuration.	<p>Make sure the field contains the port number that the app is trying to access.</p> <p>The app data is tunneled only if the hostname and port number in the app's request matches the Pattern field and Port field.</p> <p>Exception: For iOS apps using AppConnect releases prior to AppConnect for iOS SDK 2.5 and AppConnect for iOS Wrapper 2.7, only the request's hostname, not the port number, determines whether the app data is tunneled.</p>
Service	<p>Corresponds to the Service field of the AppConnect app configuration.</p> <p>The value specifies an AppTunnel service configured in the AppTunnel Configuration section of the specified Sentry.</p>	<p>Make sure the service corresponds to an AppTunnel service on the Sentry that accesses the intended app server.</p> <p>In the Admin Portal for MobileIron Core:</p> <ol style="list-style-type: none"> 1. Go to Services > Sentry. 2. Select the appropriate Sentry and click Edit. 3. In the AppTunnel Configuration section, make sure the Server List for the service includes the intended app server.
Sentry ID	The MobileIron Core internal ID for the Sentry.	Only for use by MobileIron Technical Support.

Sentry display

This display lists the Sentries that are configured for AppTunnel for this app. Make sure they are what you expect.



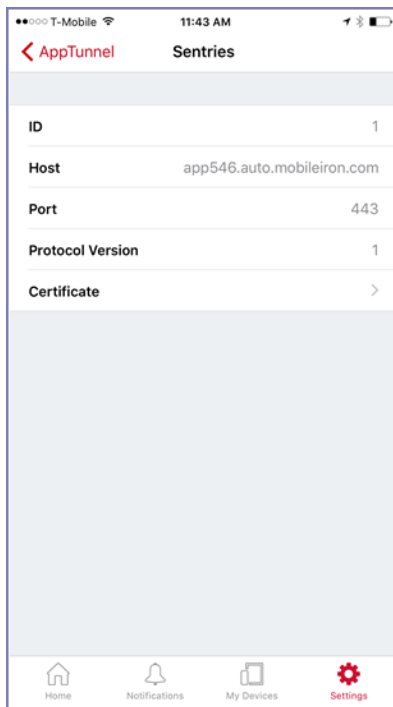


TABLE 53. SENTRIES DISPLAY FIELDS AND TROUBLESHOOTING ACTIONS

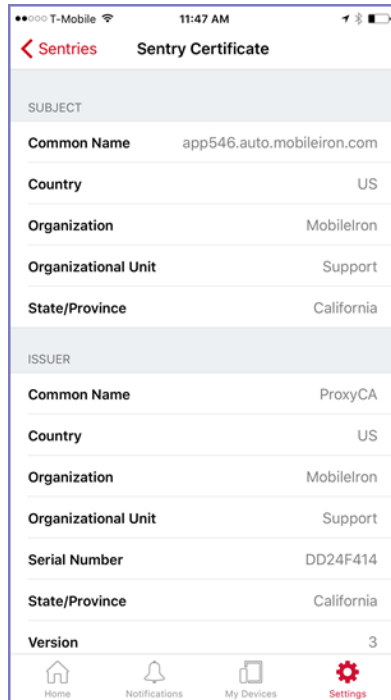
Field name	Description	Troubleshooting actions
ID	The MobileIron Core internal ID for the Sentry.	For use only by MobileIron Technical Support.
Host	Sentry host name	Make sure this Sentry is one you intended for AppTunnel for this app.
Port	Port opened to MobileIron Core.	
Protocol Version	Protocol version between the Sentry and MobileIron Core	For use only by MobileIron Technical Support.
Certificate	<p>This is the certificate that the AppConnect Library in the app uses to know that the Sentry used for AppTunnel is a trusted server.</p> <p>A valid pinned Sentry certificate must be available for tunneling.</p>	<p>Tap to see certificate information, including whether the certificate has expired.</p> <p>To view the Sentry certificate in the Admin Portal for MobileIron Core.</p> <ol style="list-style-type: none"> 1. Go to Services > Sentry. 2. Find the line for the appropriate Sentry. 3. Click View Certificate.

Related topics

“Managing certificates for Standalone Sentry” in the *MobileIron Sentry Guide*.

Sentry Certificate display

Make sure this is the certificate you intended for devices to use to know that the Sentry used for AppTunnel is a trusted server. Check the certificate fields, including whether the certificate has expired. You sometimes have to scroll down the screen to see all the fields.



Note The Following:

- If necessary, tap on a field to view the entire string.
- Scroll down to see additional fields.

TABLE 54. SENTRY CERTIFICATE DISPLAY FIELDS AND TROUBLESHOOTING ACTIONS

Field name	Description	Troubleshooting actions
Subject		<p>You can compare the certificate values to the values on the Core Admin Portal:</p> <ol style="list-style-type: none">Go to Services > Sentry.Find the line for the appropriate Sentry.Click View Certificate. <p>Related topics</p> <p>Uploading a valid Sentry certificate to Standalone Sentry</p>
Common Name		
Country		
Email Address		
Organization		
Organizational Unit		
State/Province		
Issuer		
Common Name		
Country		
Organization		
Organizational Unit		
Serial Number		
State/Province		
Version		
Validity		
Not Valid After	Expiration date	Make sure that the certificate has not expired.
Not Valid Before	Initial date	Make sure that the certificate is valid.

Uploading a valid Sentry certificate to Standalone Sentry

If the certificate is not valid, upload a valid certificate.

In the Admin Portal for MobileIron Core:

1. Go to **Services > Sentry**.
2. Find the line for the appropriate Sentry.
3. Click **Manage Certificate**.
4. Select **Upload Certificate**.
5. Click **Browse**.
6. Select the certificate and click **Upload Certificate**.
7. Click **View Certificate** to verify the certificate.



Related topics

“Standalone Sentry certificate” in the MobileIron Sentry Guide.

AppTunnel configuration troubleshooting checklist

If an app is not successfully tunneling to its app server, check the following in the MobileIron Core Admin Portal:

TABLE 55. APPTUNNEL CONFIGURATION TROUBLESHOOTING CHECKLIST

Admin Portal location	Troubleshooting actions
Settings > Additional Products > Licensed Products	<p>Make sure you have enabled the appropriate products.</p> <p>Make sure you have selected App Tunnel for Third-party and In-house apps, if you are using AppTunnel for any app besides Docs@Work.</p>
Policies & Configs > Policies AppConnect global policy	<p>Check the AppConnect global policy configuration:</p> <ol style="list-style-type: none"> 1. In the AppConnect field, make sure you have selected Enabled. 2. Make sure AppConnect global policy is applied to a label belonging to the device. If you are using the default AppConnect global policy, this step is not necessary. 3. If you do not create an AppConnect container policy for the app, select Authorize for Apps without an AppConnect container policy.
Services > Sentry	<p>Make sure the Standalone Sentry is configured with a certificate that devices use to know that the Sentry used for AppTunnel is a trusted server.</p> <p>To view the Sentry certificate in the Admin Portal for MobileIron Core.</p> <ol style="list-style-type: none"> 1. Go to Services > Sentry. 2. Find the line for the appropriate Sentry. 3. Click View Certificate.



TABLE 55. APPTUNNEL CONFIGURATION TROUBLESHOOTING CHECKLIST (CONT.)

Admin Portal location	Troubleshooting actions
Services > Sentry	Make sure the Standalone Sentry is configured for AppTunnel for the app: <ol style="list-style-type: none"> 1. Make sure Enable AppTunnel is selected. 2. In Device Authentication Configuration, make sure the correct, valid Trusted Root Certificate is uploaded. 3. In AppTunnel Configuration, make sure you have configured the Services.
Policies & Configs > Configurations AppConnect container policy	Check the AppConnect container policy for the app. Make sure it is applied to a label belonging to the device. You do not need an AppConnect container policy if the AppConnect global policy selects Authorize for Apps without an AppConnect container policy .
Policies & Configs > Configurations AppConnect app configuration	Check the AppConnect app configuration for the app: <ol style="list-style-type: none"> 1. Make sure the AppTunnel Rules point to the intended Sentry and service. 2. For Identity Certificate, make sure you have selected the correct certificate, issued from the trusted root Certificate Authority indicated by the Trusted Root Certificate uploaded to the Sentry. 3. Make sure the certificate has not expired and that its initial validity date is in the past. 4. Make sure AppConnect app configuration is applied to a label belonging to the device.

Related topics

- [Enabling AppTunnel](#)
- [AppConnect global policy](#)
- [Configuring an AppTunnel service](#)
- [Configuring the Open With Secure Email App option](#)
- [AppConnect container policies](#)
- [AppConnect app configuration](#)
- “Standalone Sentry certificate” in the *MobileIron Sentry Guide*
- “Device and server authentication” in the *MobileIron Sentry Guide*

Status of AppConnect-related policies and configurations for an app

Information is available on the Admin Portal about the status of AppConnect-related policies and configurations for AppConnect apps on a device.

To see the information:

1. On the Admin Portal, go to **Devices & Users > Devices**.
2. Expand the device details panel of an Android device, by clicking the up arrow next to the checkbox.



3. Select the **Apps** tab.
4. Select an AppConnect app.

Information displays about the policies and configurations on the device for the AppConnect app.

Viewing certificates stored in Mobile@Work

Mobile@Work for iOS stores AppConnect-related certificates which you can view in Mobile@Work at **Settings > Secure Apps > Stored Certificates**. The following table describes the types of certificates stored.

TABLE 56. CERTIFICATES STORED IN MOBILE@WORK FOR IOS

Mobile@Work version	Types of certificates stored in Mobile@Work
9.5 through 9.8	<ul style="list-style-type: none"> • Identity certificates for which all of the following are true: <ul style="list-style-type: none"> - You configured a certificate enrollment setting on MobileIron Core that is referenced by an AppConnect app configuration, Web@Work setting, or Docs@Work setting. - Core has delivered the identity certificate to Mobile@Work.
10.0 through the most recently released version as supported by MobileIron	<ul style="list-style-type: none"> • The types of certificates stored in Mobile@Work 9.5 through 9.8 • Certificates pinned to an AppConnect app using a Client TLS configuration in the app's AppConnect app configuration, Web@Work setting, or Docs@Work.

NOTE: Mobile@Work also stores certificates from derived credentials but derived credential certificates are not delivered from Core. Mobile@Work displays derived credentials separately, if applicable. See the *MobileIron Core Derived Credentials Guide* for more information.

Procedure

1. In Mobile@Work, tap **Settings**.
2. Tap **Secure Apps**.
3. Tap **Stored Certificates**.
A screen displays a list of internal identifiers for the certificates.
4. Tap on an internal identifier.
A screen displays the following information about the certificate:
 - the subject name
 - the issuer
 - When the certificate is valid (initial date and expiration date)
 - the serial number



Related topics

- “Certificate delivery time for AppConnect-related certificates” in the *MobileIron Core Device Management Guide for iOS and macOS Devices*.
- [Certificate pinning for AppConnect apps](#)

AppTunnel diagnostics in SDK-built apps

Some AppConnect for iOS apps provide a user interface option to display or log AppTunnel diagnostic information. Apps can provide this functionality only if they are built with the AppConnect for iOS SDK.

Contact the application vendor or developer to find out whether the app can provide this information.



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.

The MobileIron UEM client app 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](#)
4. [Recovering the AppConnect passcode when forgotten](#)

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 the Mobile@Work app.
If the device user does not see **Secure Apps** in the Mobile@Work menu, 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 57. 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.

Recovering the AppConnect passcode when forgotten

When you allow self-service AppConnect passcode recovery for Android devices in the AppConnect global policy, the Secure Apps Manager menu has an option **Forgot Passcode**.

NOTE: The option is available only when the device user is logged out of AppConnect apps.

To create a new AppConnect passcode, device users:

1. Open the Secure Apps Manager.
2. Tap the menu for Secure Apps Manager (in its upper right corner).
3. Tap **Forgot Passcode**.
4. Enter their MobileIron Core registration credentials.
5. Create a new secure apps passcode, confirming it by reentering it.

NOTE: Device users who leave this flow without creating a new AppConnect passcode will have to reenter their MobileIron credentials before creating a new AppConnect passcode.

See also "Self-service AppConnect passcode recovery" in [AppConnect global policy](#).

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 the MobileIron UEM client 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.

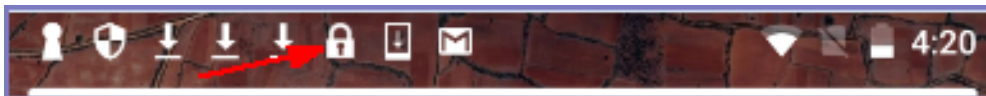
Secure apps status bar icons

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:

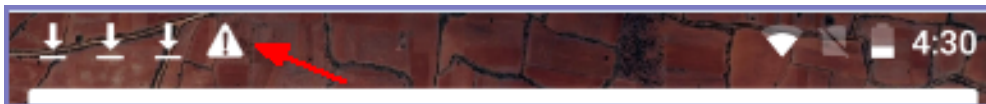


When logged out of secure apps, the icon looks like a lock that is locked, because the user is locked out of the AppConnect container. To unlock the container and access AppConnect apps, the user must enter his secure apps passcode.



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 Mobile@Work 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.

The MobileIron UEM client app also provides displays to help you troubleshoot secure apps and AppTunnel. End users typically do not use these displays. For information on these displays, see:

- [Secure apps status display in Mobile@Work](#)
- [AppTunnel configuration troubleshooting display in Mobile@Work](#)

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:

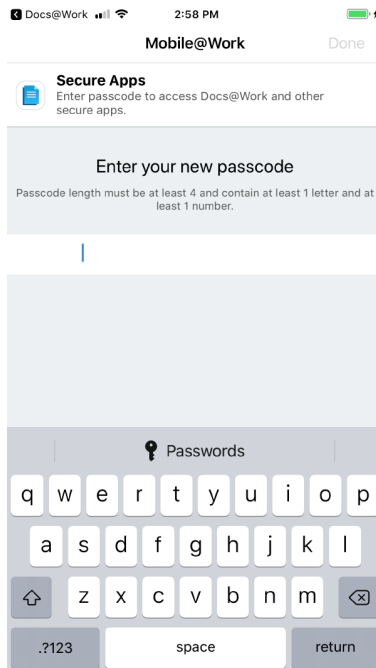
- [Creating a secure apps passcode](#)
- [Creating a more complex secure apps passcode](#)
- [Logging in with the secure apps passcode](#)
- [Logging out or resetting passcode for secure apps](#)
- [Secure apps passcode management](#)
- [Resetting the secure apps passcode - administrator initiated](#)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



Creating a secure apps passcode

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

FIGURE 12. SECURE APPS PASSWORD PROMPT

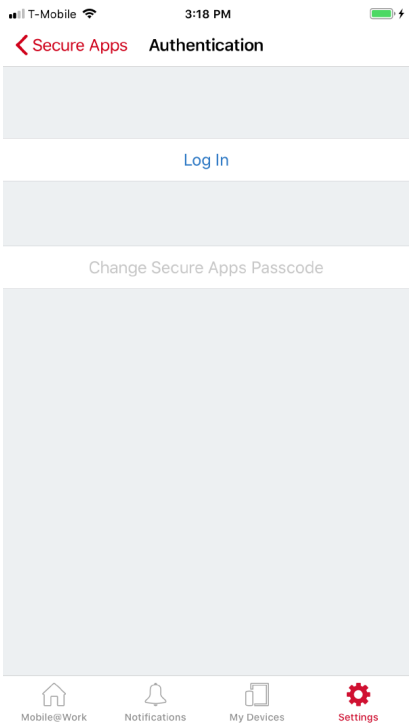


Device users can also create a secure apps password in Mobile@Work without first having to launch a secure app.

Procedure

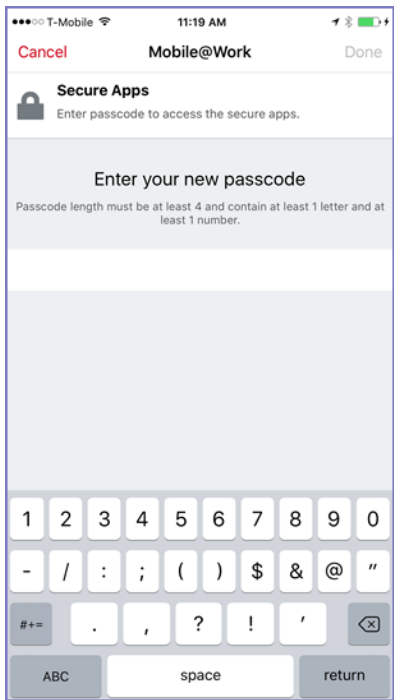
1. Launch Mobile@Work.
2. Go to Settings > Secure Apps > Authentication.

FIGURE 13. LOG IN FOR SECURE APPS PASSCODE



3. Tap Log In.

FIGURE 14. ENTER NEW PASSCODE

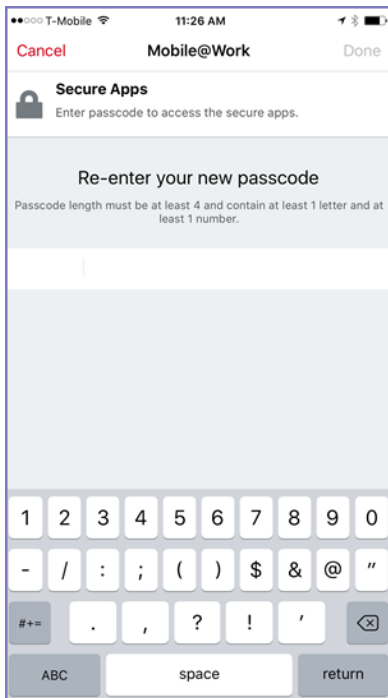


4. Enter a passcode according to the specified instructions.



5. Tap **Done**.

FIGURE 15. RE-ENTER THE NEW PASSCODE



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

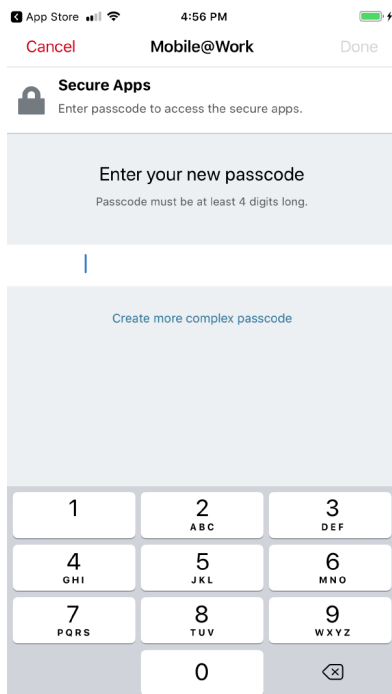
Creating a more complex secure apps passcode

Mobile@Work 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, Mobile@Work displays a numeric keypad. However, Mobile@Work 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.

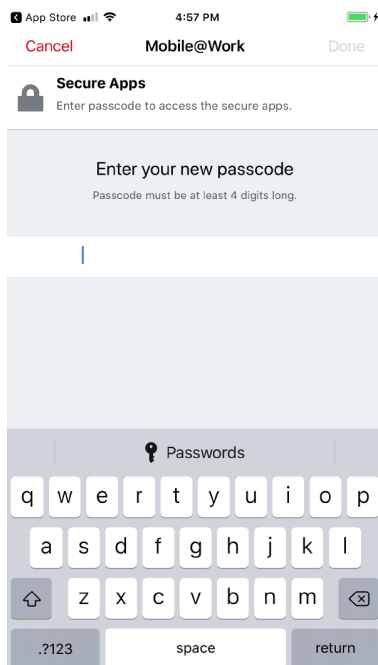
If the secure apps passcode requirements in the AppConnect global policy are 4 numeric digits, Mobile@Work displays the following:

FIGURE 16. NUMERIC PASSCODE REQUIREMENT



Mobile@Work presents a QWERTY keyboard when you tap **Create more complex passcode**.

FIGURE 17. ALPHA NUMERIC PASSCODE REQUIREMENT



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

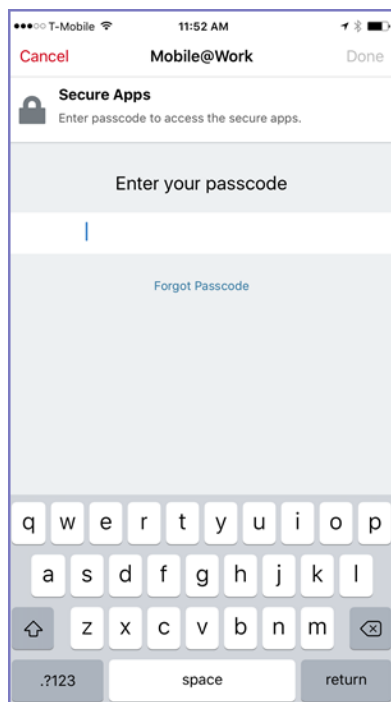
The device user has the option to create a more complex passcode when:

- 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 Core.
- After exceeding the maximum number of failed passcode attempts and reentering their user name and password for MobileIron Core.

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, Mobile@Work automatically logs the device user out of secure apps. When the user once again launches a secure app or taps **Log In** in Mobile@Work, Mobile@Work prompts the user to log in with the 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.

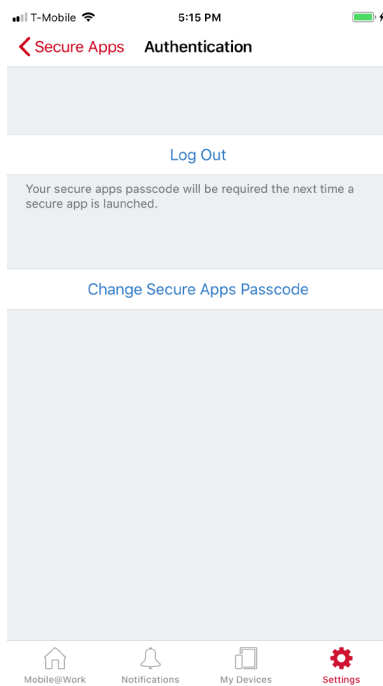
Logging out or resetting passcode for secure apps

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 Mobile@Work go to Settings > Secure Apps > Authentication.

FIGURE 18. SECURE APPS LOG OUT OR CHANGE PASSCODE

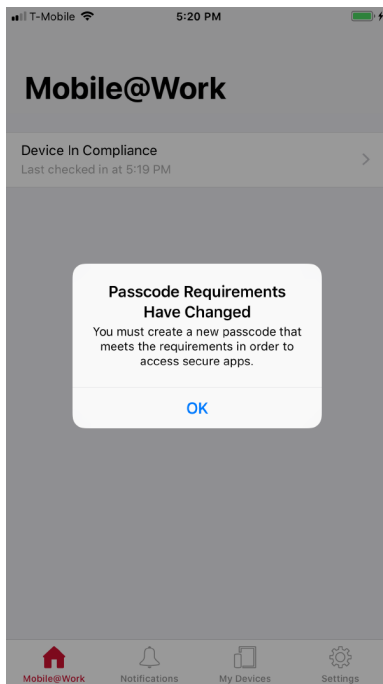


Mobile@Work prompts the device user for the secure apps passcode the next time the user launches a secure app or taps **Log In** in Mobile@Work.

Resetting the secure apps passcode - administrator initiated

You can change the secure apps passcode requirements on MobileIron Core by modifying the AppConnect global policy. When Mobile@Work checks in with Core, Mobile@Work prompts the device user as follows:

FIGURE 19. RESET PASSCODE PROMPT

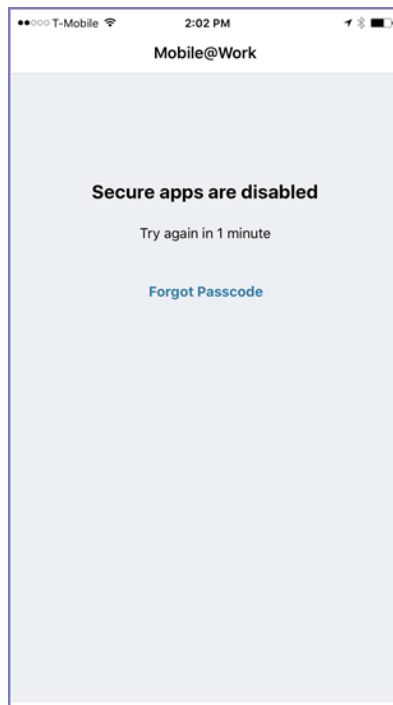


Tap **OK** and follow the prompts to reset the passcode.

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, Mobile@Work displays the following:

FIGURE 20. SECURE APPS IS DISABLED



After the maximum number of failed attempts, the device user must enter their Core 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 Mobile@Work:

- [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 are based on Mobile@Work 9.1 for iOS. Therefore, the screenshots 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:

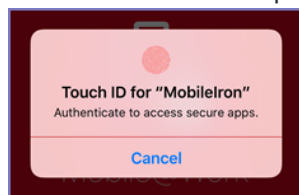
- On the AppConnect global policy, you have selected **Use Touch ID or Face ID when supported** and for **When using Touch ID or Face ID, fall back to** you have selected **Device passcode**.
- The device user has enabled the device passcode and at least one of Touch ID or Face ID.
- The device user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

Note The Following:

- Mobile@Work 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, Mobile@Work prompts the device user to enter a new secure apps passcode.
- Screenshots in this chapter are based on Mobile@Work 9.1 for iOS. Therefore, the screenshots show only Touch ID, not Face ID, but Face ID behavior is similar.

The device user chooses Touch ID/Face ID

1. Mobile@Work 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.

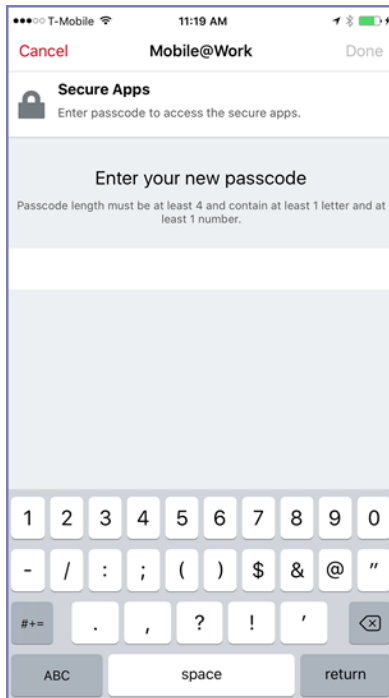


3. The device user enters the Touch ID or Face ID and is logged into secure apps.

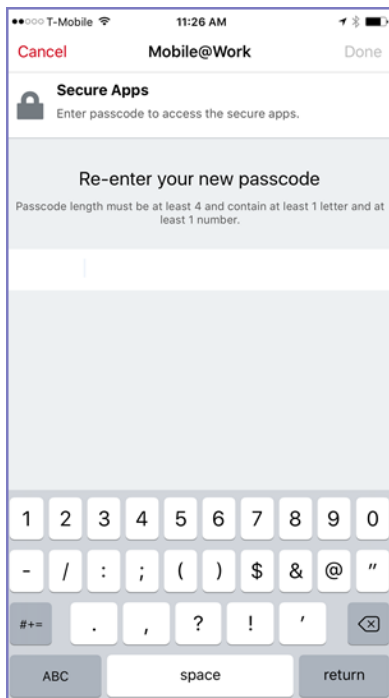
The device user will use Touch ID for all further authentications to secure apps, unless the device user changes the authentication method using **Settings > Secure Apps > Authentication** in Mobile@Work.

The device user chooses passcode

1. Mobile@Work 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.



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



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 Mobile@Work.

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

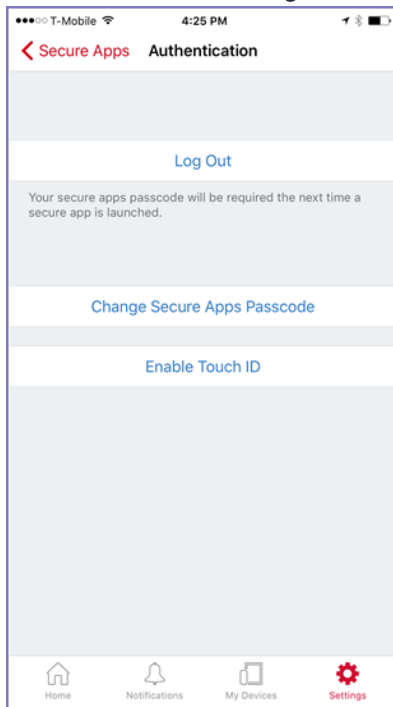
The device user can change the authentication method for accessing secure apps to Touch ID/Face ID when both of the following are true:

- 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.

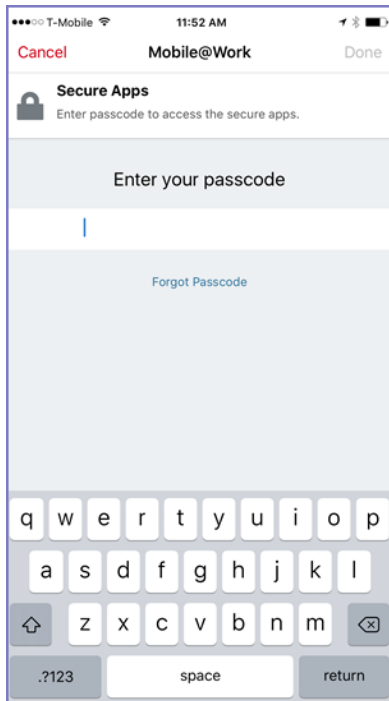
NOTE: Screenshots in this chapter are based on Mobile@Work 9.1 for iOS. Therefore, the screenshots show only Touch ID, not Face ID, but Face ID behavior is similar.

The device user does the following in Mobile@Work:

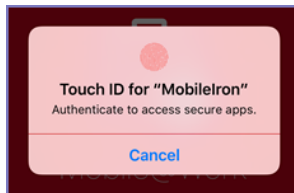
1. The device user navigates to **Settings > Secure Apps > Authentication**.



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**.



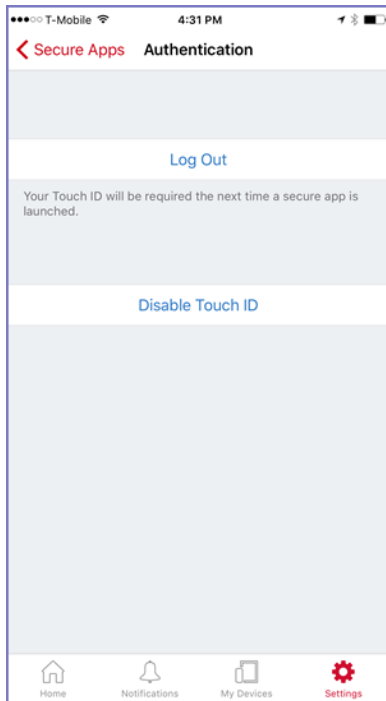
4. The device user enters the Touch ID or Face ID.

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 Mobile@Work.

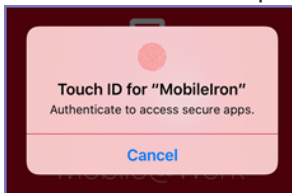
Changing from Touch ID/Face ID to secure apps passcode to access secure apps

The device user can change the authentication method for accessing secure apps to the secure apps passcode using the following steps in Mobile@Work:

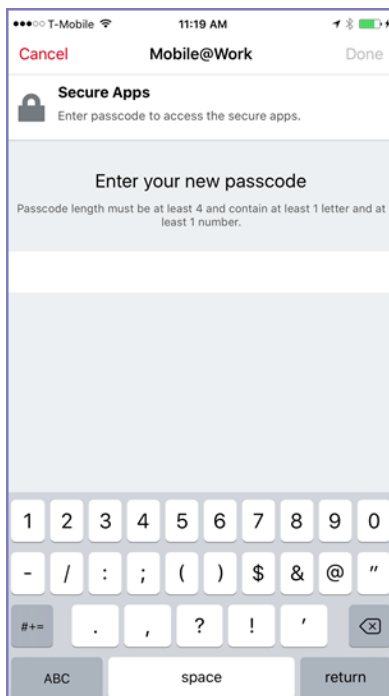
1. The device user navigates to **Settings > Secure Apps > Authentication**.



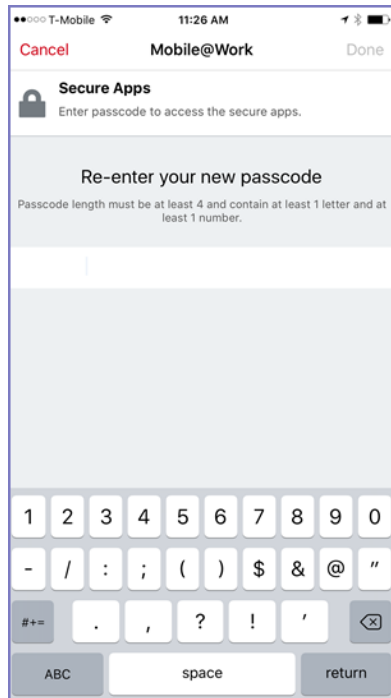
2. The device user taps **Disable Touch ID**.



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



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



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 Mobile@Work.

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 are based on Mobile@Work 9.1 for iOS. Therefore, the screenshots 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 Mobile@Work, Mobile@Work 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, Mobile@Work 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](#)

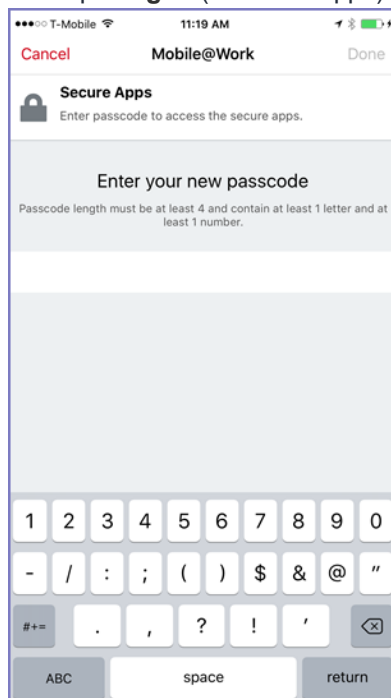
The device user can later use Mobile@Work settings to change his choice about using Touch ID/Face ID.

See also: [Touch ID or Face ID for accessing secure apps](#) for the administrative perspective.

The device user creates an AppConnect passcode

Mobile@Work 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 Mobile@Work home screen



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

NOTE: Screenshots in this chapter are based on Mobile@Work 9.1 for iOS. Therefore, the screenshots show only Touch ID, not Face ID, but Face ID behavior is similar.

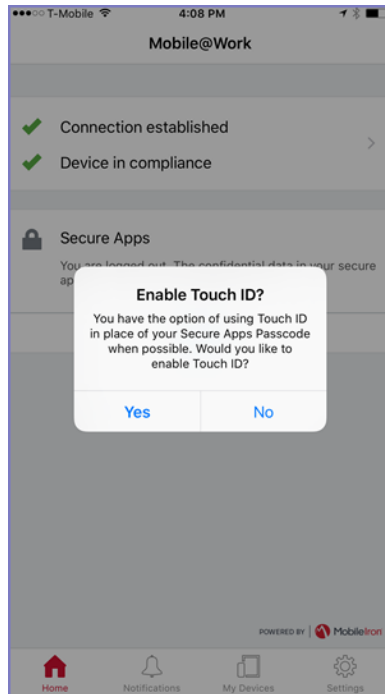
After creating the AppConnect passcode, Mobile@Work gives the device user the choice to use Touch ID or Face ID with fallback to the AppConnect passcode, or to use only the AppConnect passcode for accessing secure apps. However, Mobile@Work gives this choice *only if* the device user has already done 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, Mobile@Work displays the following:



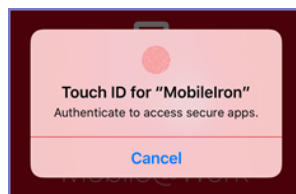
If the device user taps

- **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 Mobile@Work.
- **No**, he will use the AppConnect passcode for all further authentications to secure apps.

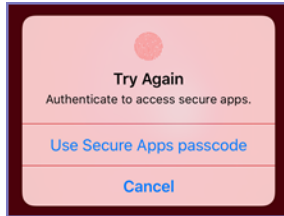
The device user uses Touch ID/Face ID when the auto-lock time expires

NOTE: Screenshots in this chapter are based on Mobile@Work 9.1 for iOS. Therefore, the screenshots 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, Mobile@Work displays the following:



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:



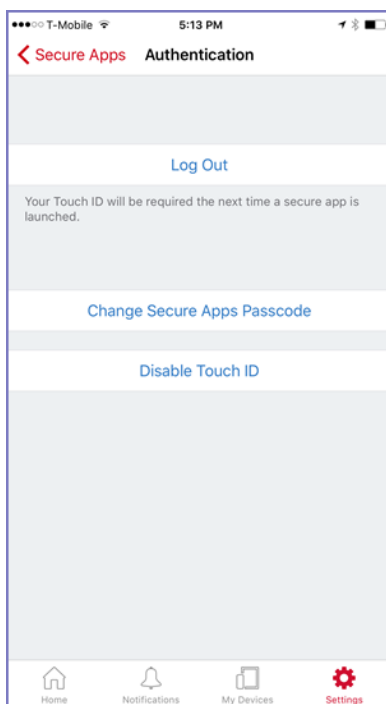
NOTE: Tapping either **Cancel** or **Use Secure Apps passcode** causes Mobile@Work to prompt the device user for the secure apps passcode.

The device user changes Touch ID/Face ID choice

NOTE: Screenshots in this chapter are based on Mobile@Work 9.1 for iOS. Therefore, the screenshots 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 Mobile@Work.

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



If the device user taps **Disable Touch ID**, Mobile@Work 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**.

