MDX Toolkit

Contents

About the legacy MDX Toolkit	3
What's new	5
Fixed issues	12
Known Issues	20
System requirements	22
Installing the MDX Toolkit	30
Non-compliant device behavior	32
Wrapping iOS mobile apps	32
Wrapping Android mobile apps	48
MDX third-party app policies at a glance	59
MDX policies for third-party apps for Android	72
Supported device policies and MDX policies for Android Enterprise	82
MDX policies for third-party apps for iOS	84
MDX developer guide	97
System requirements	104
Developing Android apps	106
Best practices for Android apps	110
API for Android	116
Developing iOS apps	123
Best practices for iOS apps	130
API for iOS	144
Policy defaults and custom policies	152
Troubleshooting	155

About the legacy MDX Toolkit

December 12, 2023

The MDX Toolkit is an app container technology that enhances the mobile device experience. The toolkit lets you prepare apps for secure deployment with Citrix Endpoint Management.

> **Note:**
>
> MDX Toolkit has reached End of Life and End of Support on July 31, 2023.

Key facts about End of Life

We've transitioned from MDX to MAM SDK for Secure Mail and Secure Web apps. Although MDX Toolkit continues operational, we recommend you to use the MAM SDK for optimal support and user experience.

In case you need assistance transitioning to MAM SDK, open a Technical Support ticket.

For more information about MDX Toolkit deprecation, see Deprecation.

For more information about MAM SDK, see MAM SDK overview.

Recent announcements

- **WKWebView Issues and Citrix SSO**

  Citrix started supporting WKWebView for mobile productivity apps as of version 20.11.x, after Apple ended support for apps using UIWebView. WKWebView is an Apple framework that replaced the previously used UIWebView framework. Due to technical limitations and the complexities of WKWebView, some tunneling issues may occur with some websites.

  Citrix may be able to provide analysis and suggest modifications to the way you render your website on a best-effort basis. Ultimately, however, if you are experiencing issues, we recommend that you use the Citrix SSO app for VPN tunneling.

  For details about Citrix SSO, see Citrix Gateway clients.

- A Mobile Application Management (MAM) SDK is available to replace areas of MDX functionality that aren't covered by iOS and Android platforms.

- The MDX wrapping technology reaches end of life (EOL) in July 2023. To continue managing your enterprise applications, you must incorporate the MAM SDK.

  For more details about the MAM SDK, see the Citrix Developer section on Mobile App Management (MAM) SDK. You can find more information as well in this Citrix blog post and in the MAM SDK overview.

The MAM SDK is available for download when you sign on to Citrix downloads.

What the MDX Toolkit does

The MDX Toolkit adds the following information to the apps:

- The code required to support mobile app management tasks such as provisioning, custom authentication, per-app revocation, data containment policies, data encryption, and per-app virtual private networking
- Signed security certificates
- Policy information and other Endpoint Management settings

The MDX Toolkit can securely wrap apps created within your organization or third-party mobile apps.

You use the Endpoint Management console to add your app to Endpoint Management. When you add the app, you can change the policy configuration, add app categories, apply workflows, and deploy apps to delivery groups.

To download Endpoint Management components, see the Download page on the Citrix website.

The MDX Toolkit version 20.10.5 is the release available on Citrix.com. For details, see What's new in the MDX Toolkit and Fixed issues.

> **Note:**
>
> The MDX Toolkit is not supported with Windows Phone.

About app wrapping

You can wrap Android or iOS apps you obtain from app vendors. With public app store distribution, you do not sign and wrap Citrix-developed apps with the MDX Toolkit. This process significantly streamlines deploying apps. Since the Endpoint Management server already supports deploying apps from the public app store, no server update is required. However, you can use the MDX Toolkit to wrap third-party or enterprise apps. For more information on public app store distribution, see Enabling public app store distribution.

Independent Software Vendors (ISVs) can wrap apps they develop and then make them available in an app store or the Citrix mobile productivity app gallery. For details, see the MDX Developer Guide

The MDX Toolkit combines app files (.ipa, .app, or .apk) with Citrix components and your keystore or signing certificate to produce a wrapped MDX app.

The MDX Toolkit supports the following third-party frameworks:

- Android and iOS apps developed on the Xamarin platform.

> **Note:**
>
> Apps developed using Xamarin.forms framework is not supported.

- Apps developed by using the PhoneGap (Apache Cordova) framework

We don't guarantee that other third-party frameworks, such as Swift, work with the MDX Toolkit, unless explicitly stated.

The MDX Toolkit and XenMobile App SDK for iOS and Android includes the following tools:

- A macOS GUI tool that can wrap both iOS and Android apps.
- A macOS command-line tool that wraps iOS apps.
- A Java command-line tool that wraps Android apps.
- XenMobile App SDK: Third-party app developers can use the XenMobile App SDK to perform actions in wrapped apps based on Endpoint Management policies. For example, if an Endpoint Management policy prevents cut and copy in a mobile productivity app, a developer can prevent text selection in the app. For details, see the MDX Developer Guide.

The Mobile Application Management (MAM) SDK

With the move away from the MDX Toolkit, the MAM SDK replaces areas of MDX functionalities not covered by the iOS and Android platforms. Rather than wrapping third-party apps using the MDX Toolkit, you instead create apps using the MAM SDK. This method of developing apps allows you to call on the APIs directly instead of relying on a wrapper. For more information about the MAM SDK APIs, see the developer documentation for Mobile Application Integration.

What's new

October 17, 2022

What's new in the current release

MDX Toolkit 21.8.5

- The MDX Toolkit reaches end of life (EOL) in July 2023 as announced previously in Deprecations and removals. As of that time, you can't wrap apps using this method anymore. To continue managing your enterprise applications, you must incorporate the MAM SDK.

  For more information about the MAM SDK, see MAM SDK overview.

  The following MAM SDKs are available for download at Citrix downloads and GitHub:

  - MAM SDK version 21.9.0 for iOS Native (Supporting iOS 15)
  - MAM SDK version 21.7.0 for Xamarin iOS [Tech preview]

  - MAM SDK version 21.7.0 for Cordova iOS [Tech preview].

- Support for iOS 15.
- Support for Android 12.

MDX Toolkit 21.6.0

- Citrix started supporting WKWebView for mobile productivity apps as of version 20.11.x, after Apple ended support for apps using UIWebView. If the iOS apps that you are wrapping using the MDX Toolkit have upgraded to WKWebView, then you must use the following Network Access policy options:
  - **Blocked**
  - **Unrestricted**
  - **Tunneled - Web SSO**

What's new in earlier releases

MDX Toolkit 21.3.0

This MDX Toolkit version contains fixes. For details, see Fixed issues.

MDX Toolkit 20.10.5

This MDX Toolkit version contains fixes. For details, see Fixed issues.

MDX Toolkit 20.8.5

This MDX Toolkit version contains fixes. For details, see Fixed issues.

MDX Toolkit versions 19.9.5 to 20.7.0

These MDX Toolkit versions contain fixes. For details, see Fixed issues.

MDX Toolkit 19.9.0

This release supports iOS 13. Note the following policy change: The Allowed Wi-Fi network MDX policy is not supported on devices running iOS 13 or later.

MDX Toolkit 19.8.0

This release supports Android Q.

MDX Toolkit 19.6.5

Encryption management. Encryption management allows you to use modern device platform security while also ensuring the device remains in a sufficient state to use platform security effectively. A set of security criteria is identified that a device must adhere to, be considered compliant for encryption management. You are then able to identify non-compliant devices, and restrict access to apps on devices that are non-compliant with these criteria.

By using encryption management, you eliminate local data encryption redundancy since file system encryption is provided by the Android and iOS platforms. Encryption management also improves performance (avoiding double encryption) and application compatibility with MDX.

> **Note:**
>
> To help get started, run a new report in Citrix Endpoint Management to list the non-compliant devices are in your organization. This report helps determine the impact of turning on compliance enforcement. To access the report, open the Endpoint Management console and navigate to **Analyze > Reports > Non-Compliant Devices** and generate the report.
> The Non-Compliant Devices report is available in environments running Citrix Endpoint Management version 19.6.0.2 or later.

Encryption type

To use the encryption management feature, in the Endpoint Management console, set the **Encryption type** MDX policy to **Platform encryption with compliance enforcement**. This enables encryption management and all the existing encrypted application data on users' devices seamlessly transition to a state that is encrypted by the device and not by MDX. During this transition, the app is paused for a one-time data migration. Upon successful migration, responsibility for encryption of locally stored data is transferred from MDX to the device platform. MDX continues to check compliance of the device upon each app launch. This feature works in both MDM + MAM and MAM-only environments.

When you set the **Encryption type** policy to **Platform encryption with compliance enforcement**, the new policy supersedes your existing MDX Encryption.

For details about the encryption management MDX policies, see the **Encryption** section in:

- MDX policies for iOS apps
- MDX policies for Android apps

Non-compliant device behavior

When a device falls below the minimum compliance requirements, the Non-compliant device behavior MDX policy allows you to select what action is taken:

- **Allow app** - Allow the app to run normally.
- **Allow app after warning** - Warn the user that an app does not meet the minimum compliance requirements and allows the app to run. This is the default value.
- **Block app** - Block the app from running.

The following criteria determine whether a device meets the minimum compliance requirements.

Devices running iOS:

- iOS 10: An app is running operation system version that is greater than or equal to the specified version.
- Debugger access: An app does not have debugging enabled.
- Jailbroken device: An app is not running on a jailbroken device.
- Device passcode: Device passcode is ON.
- Data sharing: Data sharing is not enabled for the app.

Devices running Android:

- Android SDK 24 (Android 7 Nougat): An app is running operation system version that is greater than or equal to the specified version.
- Debugger Access: An app does not have debugging enabled.
- Rooted devices: An app is not running on a rooted device.
- Device lock: Device passcode is ON.
- Device encrypted: An app is running on an encrypted device.

Support for WkWebView. This release includes support for WkWebView. WKWebView is an Apple framework that displays web content and has performance and security improvements over the previously used UIWebView framework. The iOS apps that you build using the WKWebView framework allow traffic to go through micro VPN (also referred to as SSL VPN) when using the Tunneled - Web SSO policy. To use this feature, no setup configuration is required.

> **Note:**
>
> If you are already using the Full VPN tunnel, you can either continue using the UIWebView framework, or switch to using the Secure Browse mode (recommended).

Limitations

WkWebView is not supported in the following scenarios:

- Devices running iOS 10 or earlier.
- Setups running Endpoint Management integration with EMS/Intune.
- Apps that use two instances of the WKWebView component simultaneously.
- Setups configured for Full VPN Mode.

Support for 64-bit apps for Google Play. Beginning on August 1, 2019, Google Play requires that apps support 64-bit architectures. This version of the MDX Toolkit supports the wrapping of 64-bit versions of apps. To assess if your app is prepared for 64-bit devices and for instructions on building apps with 64-bit libraries, see the Google Developers documentation on Google Play.

Updated crypto libraries. The MDX Toolkit 19.6.5 includes updated crypto libraries. These libraries are updated periodically to keep up with the latest security trends and to help fix security vulnerabilities. We also deprecated older ciphers. This update improves security because the update enforces the environment to use the latest and most secure ciphers. Not updating the ciphers, however, may result in an error when users update to apps that you wrap with the MDX Toolkit 19.6.5.

If you're a Citrix Gateway customer and plan to allow SSL VPN through your iOS and Android third-party apps, confirm that you've completed the following steps.

- On the Citrix Gateway, add the following cipher suite value in the SSL Ciphers option: - ECDHE-RSA-AES256-GCM-SHA384. For steps to add Secure Cipher, see Secure cipher alias.
- Enable elliptical curve cryptography (ECC). For details, see ECDSA cipher suites support.

Support for Chrome 74 for devices running Android. This release includes micro VPN support on devices running Android that have upgraded to Chrome version 74.

Support for Apktool version 2.4. This release includes support for apps using Apktool version 2.4.

MDX Toolkit 19.3.5

The MDX Toolkit version 19.3.5 contains fixes. For details, see Fixed issues.

MDX Toolkit 18.12.0

Network policy behavior. New policies replace the previous network policies in order to unify functionality and make the policies more intuitive. See the following table for more information about the changes.

Legacy policies	Intune Policy	New policy	Notes
Network access, Preferred VPN mode, and Permit VPN mode switching	Enable http/https redirection (with SSO), Disable mVPN full tunnel (TCP level) redirection	Network Access	
Split tunnel exclusion list	mVPN tunnel exclusion list	Exclusion list	
Online session required	mVPN session required	micro VPN session required	If Endpoint Management MAM, a corresponding grace period policy controls how long users have to reestablish a VPN session. Intune MAM does not support a grace period.

For the new Network Access and micro VPN Session Required policies, we encourage you to select a new value. By default, **Use Previous Settings** is selected, which uses the values you had set in the earlier policies. Once changed, you shouldn't revert to **Use Previous Settings**. If you are publishing a new app, do not select **Use Previous Settings**. Also note that changes to the new policies do not take effect until the user upgrades the app to 18.12.0.

For newly uploaded apps, the defaults are as follows:

- **Network access: Blocked** for all apps except Secure Mail. Because Intune does not have a blocked state, the default for Secure Mail is **Unrestricted**.
- **Exclusion list:** Empty
- **micro VPN session required: No**

> Note for customers upgrading to 18.12.0:
>
> The Network access policy defaults to your previous VPN setting and requires no action. If you change the access type, that change doesn't go into effect until you upgrade your apps to 18.12.0.

For details, see the **App Network Access** section in:

- MDX policies for iOS apps
- MDX policies for Android apps

MDX Toolkit 10.8.60

MDX supports Android P. You can now wrap apps for Android P.

MDX is now available in Polish.

MDX Toolkit 10.8.35

Changes to the Exclusion policy. On Android devices, non-MDX apps can now receive decrypted copies of files encrypted within MDX apps. When the Document Exchange (Open In) policy is set to **Restricted**, apps listed in the Restricted Open-In exception list policy can receive files that have been encrypted in MDX apps. When receiving files, the content of those files is decrypted to local storage and is deleted from local storage upon closing the file. For example, adding {package=com.microsoft.office.word} to the Document Exchange (Open In) Policy enables the Word application to receive decrypted files from an MDX application.

MDX Toolkit 10.7.20

Control Net Promoter Score survey: A new policy for Citrix Files apps, Allow NPS Citrix Files, allows you to occasionally display a Net Promoter Score survey to users for feedback. The default value is **Off**.

Do not tunnel endpoints policy: Some service endpoints that Citrix Endpoint Management SDKs and apps use for various features need to be excluded from micro VPN tunneling. MDX does this by default, but it is possible to override this list by setting a client property on the Citrix Endpoint Management server. For details about configuring client properties in the Citrix Endpoint Management console, see Client properties. For details about overriding the service endpoint list, see TUNNEL_EXCLUDE_DOMAINS. The default list of domains that are excluded from tunneling by default are as follows.

- `ssl.google-analytics.com`
- `app.launchdarkly.com`
- `mobile.launchdarkly.com`
- `events.launchdarkly.com`
- `stream.launchdarkly.com`
- `clientstream.launchdarkly.com`
- `firehose.launchdarkly.com`
- `hockeyapp.net`
- `rttf.citrix.com`
- `rttf-test.citrix.com`
- `rttf-staging.citrix.com`
- `cis.citrix.com`

- `cis-test.citrix.com`
- `cis-staging.citrix.com`
- `pushreg.xm.citrix.com`
- `crashlytics.com`
- `fabric.io`

New policies to control opening URLs from Secure Mail in the native browser: The SecureWebDomains policy controls which domains are sent to the Secure Web browser instead of the native browser. A list of comma-separated URL host domains is matched against the host name portion of any URL the application would normally send to an external handler. Typically, administrators configure this policy as a list of internal domains for Secure Web to handle.

This feature is available iOS and Android. An earlier known issue for the policy on Android devices was fixed in version 10.6.25.

The ExcludeUrlFilterForDomains policy is a comma-separated list of website domains excluded from URL filtering. URLs including any domain in the list get sent to the user's native browser instead of Secure Web. If the policy is empty, then all URLs are passed through the URL filters. This policy takes priority over the SecureWebDomains policy. The default policy value is empty.

MDX support for crash reporting to the Citrix Insight Service (CIS): If Citrix Reporting policies are **On** and a crash happens, MDX creates a report bundle with any logs and crash reports available and uploads it to CIS. It then deletes the support bundle and notifies the related Mobile productivity app of the crash so that it creates a support bundle as well.

MDX Toolkit 10.7.10

The MDX Toolkit 10.7.10 is the final release that supports the wrapping of mobile productivity apps. Users access mobile productivity apps versions 10.7.5 and later from the public app stores. For table listing the mobile productivity apps enterprise versions that you can wrap with the MDX Toolkit 10.7.10, see the Enterprise delivery of mobile productivity apps section in Mobile productivity apps administration and delivery.

The MDX Toolkit version 10.7.10 contains fixes. For details, see Fixed issues.

MDX Toolkit 10.7.5

The MDX Toolkit version 10.7.5 contains fixes. For details, see Fixed issues.

Fixed issues

September 8, 2021

MDX Toolkit 21.8.5

- In Secure Mail for Android, the following error message appears when you reboot Android Enterprise devices running Android 10 or Android 11:
  "Unsupported change to SDK Management Mode Policy" [CXM-99115]

MDX Toolkit 21.6.0

- On devices running Android, you are unable to wrap an .apk with the MDX Toolkit. This issue occurs if the app is using Apktool 2.4.1 or below and targeting Android SDK 30. [CXM-94960]
- While trying to view PDF files on certain third party apps, the DSMobile app doesn't appear in the app list. This issue occurs on Samsung devices running Android 10 or later. [CXM-95087]
- In Secure web for iOS, you are unable to access websites by clicking links unless you manually pull to refresh the page. [CXM-96493]
- When WKWebView is enabled and *login.windows.net* is added to the mVPN tunnel exclusion list, authentication fails and you are unable to configure Secure Mail for iOS. [CXM-96721]

Fixed issues in earlier versions

MDX Toolkit 21.3.0

- On devices running iOS version 14.5 and later, Secure apps and wrapped apps crash upon launching. [CXM-91992]
- On devices running iOS, certain apps crash with MDX Toolkit versions 20.8.5 and 20.10.5 [CXM-92768]
- On devices running Android in the Work Profile mode, the App Geofence policy does not work as expected. [CXM-92917]
- In MDX for Android, you can't launch certain apps through Secure Hub. You get the following error: **MDX encryption is not supported on this version of android. Please reinstall the app.** [CXM-92986]

MDX Toolkit 20.10.5

- On devices running Android, certain apps crash with MDX Toolkit version 20.7.0. [CXM-85203]
- In MDX for iOS, when you access certain intranet sites, `/vpn/index.html` is appended to the URL when a Citrix ADC session expires. [CXM-85345]
- On devices running iOS, you can't launch certain apps. You get the error **Please update Secure hub to use this app** even when there are no updates available for Secure Hub. [CXM-86669]
- On devices running Android, you can't launch certain apps. [CXM-86887]

- On devices running iOS, authentication on certain websites fails if Secure Web is using WKWeb-View. [CXM-87056]
- You are unable to configure Secure Mail for iOS on setups running Intune MAM and the following error appears: **Gateway Login Failed. Please retry with valid login credentials.** [CXM-88023]

MDX Toolkit 20.8.5

- On devices running iOS, you can't use the **Open In** option to open files in some apps. You get the following error: "Restricted by admin policy **Open In** is blocked for this app." [CXM-84436]
- On devices running Android, some apps force quit after wrapping with MDX Toolkit version 20.3.0. [CXM-84511]
- On devices running iOS, you can't open any files on Citrix Files using Secure Mail or QuickEdit, when the encryption is set to Platform encryption. [CXM-84725]
- On devices running iOS, when the encryption is set to Platform encryption, users receive random prompts to verify Touch ID for Secure Mail. [CXM-84849]

MDX Toolkit 20.7.0

- In MDX for iOS, automatic renewal of the client certificate fails on certain wrapped apps. [CXM-76040]
- On devices running iOS, certain apps crash when wrapped with MDX Toolkit version 19.3.0 and later. [CXM-76271]
- When you launch certain managed apps, the screens of unmanaged apps appear until the Secure Hub store is refreshed. [CXM-78975]
- On devices running iOS, documents from an Intune managed application fail to open with an MDX managed application. [CXM-79109]
- On devices running Android, certain wrapped apps switch from Microsoft Intune to Endpoint Management. [CXM-79279]
- On devices running iOS, you are unable to open files from certain apps to Secure Mail. [CXM-80194]
- On devices running iOS, ShareFile single sign-on authentication fails with SecureBrowse mVPN mode in MDX. [CXM-80466]
- On devices running Android, you are unable to download attachments in certain MDX wrapped apps. [CXM-80819]
- Unable to wrap certain apps with MDX version 20.3.0. You get the following error: **Could not create Citrix Mobile app. Click View Log for details**. [CXM-80953]
- Internally wrapped apps crash on devices running Android. [CXM-81244]
- When WkWebview is enabled, single sign-on fails on some internal websites[CXM-83361]
- In MDX for Android, certain apps force quit after wrapping with MDX Toolkit version 20.3.0. [CXM-83448]

MDX Toolkit 20.3.0

- Custom applications crash at launch after wrapping with the MDX Toolkit. [CXM-76084]
- Unable to connect in-house Android apps to the back end server after being wrapped with the MDX Toolkit. [CXM-76139]
- On devices running iOS, some wrapped apps may crash when you try to open attachments through applications other than Secure Mail. [CXM-76248]
- On iOS, apps wrapped with MDX Toolkit version 19.9 or 19.11 display a blank screen when opened. [CXM-77042]
- On devices running Android, certain third party apps crash, and the following error appears: **Error: Service unavailable** [CXM-77616]
- On devices running Android, certain third party apps encounter socket timeout errors while using full VPN to upload a file with chunk size greater than 16 KB. [CXM-78115]
- Wrapped Android apps crash when you send an SMS from the app. The following error message appears if you restart the app:
  Starting the secure network connection... [CXM-78795]
- SAP Fiori app crashes on the splash screen after wrapping with the MDX Toolkit. [CXM-79117]

MDX Toolkit 19.11.6

On Samsung Galaxy S10 devices running Android 10, the Clipboard policies, such as restrictions on cut, copy, paste functions, do not work with third-party apps that you wrap with the MDX Toolkit. [CXM-74863]

MDX Toolkit 19.11.5

- On devices running Android, **WebViews** created before the **Web SSO** tunnel is established do not have their traffic routed. [CXM-68445]
- On devices running Android, Citrix QuickEdit can crash when opening a file from another MDX app. The following error message appears: Starting the secure network connection. [CXM-70454]
- In MDX for Android, when you try opening wrapped apps that contain only ARMEABI native libraries, the following error appears: This device does not support encryption features required by the application. [CXM-72227]
- On devices running Android, some apps crash on Samsung devices when the Tunneled - Web SSO policy is enabled. This issue is due to a missing deprecated Apache library. [CXM-72586]
- On devices running iOS, a blank screen appears when you launch an in-house app. [CXM-73168]
- On devices running Android, when you try to open the wrapped Vocera Collaboration Suite app, the app displays a blank screen. [CXM-73715]

- On devices running iOS, some enterprise apps may crash when wrapped with MDX Toolkit version 19.9.0. [CXM-73924]
- On devices running Android, the UTV app fails to load. [CXM-74190]

MDX Toolkit 19.9.5

- When the Inbound document exchange MDX policy is set to **Restricted** in Citrix Endpoint Management, the following issue occurs: On iPhones running iOS 13, after users download a file from Secure Web, users cannot open the file in Secure Mail. Instead, the following error appears: Restricted by admin policy: Inbound document exchange is blocked for this app. [CXM-72365]

MDX Toolkit 19.9.0

The following issues are fixed in the MDX Toolkit 19.9.0:

- On devices running iOS, in-house apps wrapped with MDX can't connect to internal networks. [CXM-70156]
- On devices running Android, Citrix QuickEdit can crash when opening a file from another MDX app and the following error message appears: "Starting the secure network connection…" [CXM-70454]
- On devices running Android, certain wrapped apps are unable to support network calls. [CXM-71069]
- On Android devices, Secure Browse mode does not support a URL Connection that has passed a **Proxy.DIRECT** Object in its **openConnection()** call. [CXM-71333]

MDX Toolkit 19.8.0

The following issues are fixed in the MDX Toolkit 19.8.0:

- On devices running Android, certain wrapped apps are unable to support network calls. [CXM-68439]
- In-house wrapped apps running on Android 9 may not be able to communicate with internal servers. [CXM-68607]
- Android users whose devices are managed in Citrix Endpoint Management receive multiple user certificates from the certificate authority server. [CXM-69132]
- On Android devices, if you uninstall an app when a Full VPN tunnel is established, the VPN connection is disconnected [CXM-70392]
- You are unable to access MDX applications on some 64 bit Sony and Huawei devices, due to encryption errors. [CXM-70738]

MDX Toolkit 19.6.5

The following issues are fixed in the MDX Toolkit 19.6.5:

- On devices running iOS, third party apps may fail to launch when wrapped. [CXM-62402]
- On devices running iOS, you are unable to attach files in Secure Mail with the Copy to Secure Mail option. When you try to attach files from WeChat, NETDiSK, or any other mailbox applications, the following error appears: Unable to attach file. [CXM-63441]
- On setups running Citrix Endpoint Management integration with Microsoft Intune/EMS configured in WebSSO mode, you are unable to access your company network using Secure Mail for Android. [CXM-63568]
- On Android devices, when attempting to share a web link through Secure Mail, the web link does not get populated in the body of the email. [CXM-63650]
- On iOS devices, you are unable to upload image files onto websites using devices iPhone XS Max and iPad Pro 12.9 inches. [CXM-65343]
- On devices running Android, single sign-on fails when the Enable encryption policy is set to Off. [CXM-66659]
- On devices running iOS, you are able to connect to Wi-Fi networks outside of the allowed Wi-Fi networks defined in the Allowed Wi-Fi networks MDX policy. This issue allows you to open Secure Mail and Secure Web for iOS through networks that are not listed in the MDX policy. [CXM- 66730]
- On Android devices, when you log on to ShareFile, you cannot route connections via the Tunnel Network Access. When you try to log on, Chrome 74 settings do not work. [CXM-66758]
- On iOS devices, when you deploy the in-house app Atlas, the app crashes when you open the app and provide your credentials. [CXM-66805]
- On Android devices running Secure Web, users cannot access Internet or intranet sites after a Full VPN connection is established. [CXM-66940]
- On Android devices running in Secure Browse mode, MDX requests new Citrix Gateway certificates because the end point has unexpectedly ended the current connection. [CXM-67086]
- On devices running Android 9 and later, for apps wrapped with MDX, you get the following error: **Detected problems with API compatibility (visit g.co/dev/appcompat for more info)**. [CXM-67204]

MDX Toolkit 19.3.5

- In the MDX Toolkit for iOS, build failures occur during wrapping if you are running a version of Xcode that is earlier than Xcode 10.x. [CXM-59394]
- In wrapped Android applications, the UserAgent string gets appended multiple times, causing the header size to increase. This behavior results in an error and the page fails to load. [CXM-59869]
- You cannot launch Apache Cordova app for iOS in the background when wrapped with the MDX

Toolkit 18.12.0. [CXM-61255]

MDX Toolkit 18.12.0

- Third-party storage apps are unable to download files intermittently when wrapped with the MDX Toolkit. [CXM-58814]
- Android apps developed in-house on the Xamarin platform perform slowly when wrapped with the MDX Toolkit. [CXM-58779]

MDX Toolkit 10.8.60

- Secure Mail for iOS cannot save video files to ShareFile. [CXM-42238]
- When a proxy auto-config file with dnsResolve defined is configured in Full Tunnel mode, browsing with Secure Web is noticeably slowly. [CXM-49567]
- When wrapped with the MDX Toolkit, the Cisco Jabber app crashes on login. [CXM-51052]
- Enterprise Apps may experience connectivity issues to internal resources when Preferred VPN mode is set to SecureBrowse. [CXM-52309]
- Some third party apps wrapped with the MDX Toolkit crash on startup. [CXM-52311]
- Apps that specify android.support.multidex.MultiDexApplication or android.app.Application as their application class cannot connect to internal networks in the Secure Browse mode. [CXM-53126]
- On Android devices, multiple certificates are being generated and certificates are being revoked before their expiration date. [CXM-53428]
- On Android, Secure Mail crashes when users are signed out of Secure Hub. [CXM-53930]
- On iOS devices, Secure Web and Secure Mail 10.8.45 crash on launch. [CXM-54089]
- When users enroll a device running Secure Mail for Android with Intune company portal, Secure Mail stops working. [CXM-54178]
- On Android, when opening a PDF in Quick Edit, an error displays, saying "Error initializing PDF renderer". [CXM-54950]
- When you wrap ShareFile 7.1 for Android with the MDX Toolkit, single sign-on (SSO) fails when users try to access content through ShareFile Connectors. [CXM-55030]

MDX Toolkit 10.8.35

Android

- On devices that use the WebView API, including Secure Web for Android, you are unable to attach images from a gallery although the MDX policy Block Gallery is set to OFF. [CXM-41475]
- On Android, Secure Web is unable to open all internal/external sites. [CXM-47379]
- On Android, in-house apps wrapped with MDX Toolkit 10.7.20 and later crash on launch. [CXM-47566]

- Certain enterprise apps crash while starting and the following error message appears: "Unfortunately, the app has stopped." [CXM-49901]

iOS

- On iOS, MDX Toolkit 10.7.20 cannot wrap some iPad-only third-party apps. [CXM-44122]
- On iOS, users are unable to paste links from ShareFile into other MDX apps except Secure Mail. [CXM-44274]
- On iOS, after wrapping the app SAP Fiori, the login page does not display. [CXM-45542]
- When you configure the Export contacts MDX policy, in Secure Mail on iOS 10 devices, you cannot share contacts or sync with local contacts. Instead, a Contact Export failed message appears. [CXM-44613]
- On iOS, third-party apps crash when calling CFSocketConnectToAddress in unrestricted mode. [CXM-46592]
- Secure Web for iOS crashes while browsing certain websites in the Secure Browse mode if your network is tunneled. [CXM-47989]
- For third-party iOS Cordova apps wrapped with the MDX Toolkit version 10.7.20, after you enable the Obscure screen contents policy, a black screen appears on iOS devices instead of a PIN screen. [CXM-48471]
- Once you upgrade to iOS 11.3, when the "Paste" and "Cut and Copy" policies are restricted, you are able to perform the paste operation in unmanaged apps. [CXM-50427]

MDX Toolkit 10.8.5

Android

- On Android, when the BLOCK_DNS_FROM_UNMANAGED_APPS_IN_FULL_VPN policy is set and an MDX application requests a full tunnel VPN while a VPN is already running, network failure occurs. Users need to log off then log on to resolve the issue. [CXM-42853]
- After the micro VPN connection on Android devices is established: When users restart the device, micro VPN fails to start. [CXM-43919]
- On devices running Android, some apps do not work if connections are tunneled to the internal network. [CXM-44735]
- An error occurs when trying to wrap an .apk file with the MDX Service and wrapping fails. [CXM-47060]

iOS

- With encryption disabled, you cannot perform a copy and paste in XenMobile Apps on devices running iOS. [CXM-43920]

- On iOS, after upgrading XenMobile Apps to 10.7.30, if Log Level is set to 11 or higher, Secure Mail is extremely slow and crashes if left open. [CXM-46721]

Known Issues

September 8, 2021

MDX Toolkit 21.8.5

There are no known issues in this release.

MDX Toolkit 21.6.0

There are no known issues in this release.

Known issues in earlier versions

MDX Toolkit 21.3.0

There are no known issues in this release.

MDX Toolkit 20.10.5

There are no known issues in this release.

MDX Toolkit 20.8.5

There are no known issues in this release.

MDX Toolkit 20.7.0

There are no known issues in this release.

MDX Toolkit 19.6.5 to 20.3.0

There are no known issues in these releases.

MDX Toolkit version 18.12.0

There are no known issues in the MDX Toolkit 18.12.0 release for Android devices. For fixed issues, see Fixed issues.

MDX Toolkit version 10.8.60

There are no known issues in the MDX Toolkit 10.8.60 release for Android devices. For fixed issues, see Fixed issues.

MDX Toolkit version 10.8.35

- When the Document Exchange policy is set to restricted, Secure Web stops working after trying to download a file. [CXM-48447]

MDX Toolkit version 10.8.5

There are no known issues in the MDX Toolkit 10.8.5 release for Android devices. For fixed issues, see Fixed issues.

MDX Toolkit version 10.7.20

- When VPN mode is set to Secure Browse, configuring OAuth for Citrix Office365 might fail if the Cipher suite supported by back end server is not supported by NetScaler in the deployment. [CXM-41738]
- On Android, the Secure Web Domains policy does not open URLs in the default web browser. [CXM-43021]

MDX Toolkit version 10.7

- On iOS 11, users are able to move data from an MDX managed app to an unmanaged app using the drag and drop feature. [CXM-38106]
- On iOS 11, when the Document Exchange policy is set to Restricted, unmanaged apps still appear in the open-in list. [CXM-38705]
- On iOS 11, files from ShareFile opened in MDX wrapped applications appear corrupted. [CXM-38900]
- On iOS, managed apps do not appear in the open in list on the first attempt in managed apps. [CXM-38897]
- On iOS 10 and 11, selecting open in from MDX managed apps displays an error message. [CXM-38912]

System requirements

June 2, 2020

This article provides the system requirements for using the MDX Toolkit to wrap mobile apps. The article also provides the requirements specific to app platforms.

> **Important:**
>
> The XenMobile App SDK 10.2 now requires the following components: JavaScriptCore.framework and LocalAuthentication.framework.

- **Java Development Kit (JDK) 1.7 or 1.8:** You can download the JDK 1.8 from Java SE Development Kit Downloads on the Oracle website. For installation instructions, see the JDK 8 and JRE 8 Installation Guide on the Oracle website. Be sure to install the full JDK and set JDK 1.8 as the default.
- **macOS:** Use the most recent version. The installer for the MDX Toolkit and XenMobile App SDK must run on macOS. The installer includes macOS tools that wrap both iOS and Android apps and a Java command-line tool that wraps Android apps.
- **XenMobile App SDK:** Use the most recent version of the XenMobile iOS SDK and the Xcode; bitcode generation disabled.

Other requirements for wrapping iOS mobile apps

To obtain access to the app wrapping prerequisites for iOS, you must register for an Apple distribution account. There are three types of iOS developer accounts: Enterprise, Individual, and University. Citrix strongly recommends iOS Developer Enterprise accounts. The MDX Toolkit is also compatible with iPadOS.

- **iOS Developer Enterprise accounts:** The only type of Apple Developer account that allows you to provision, deploy, and test unlimited apps to unlimited devices, with or without app wrapping. Be sure to distribute your Developer Certificate to your developers so they can sign apps.
- **iOS Developer Individual accounts:** Limited to 100 registered devices per year and do not qualify for app wrapping and enterprise distribution with Citrix Endpoint Management.
- **iOS Developer University accounts:** Limited to 200 registered devices per year and do not qualify for app wrapping and enterprise distribution with Endpoint Management.

> **Important:**
>
> Be sure to track when the provisioning profiles for your account are due to expire and renew the profiles before they expire. When a profile used to wrap apps expires, you must renew the profile, rewrap the apps, and then reinstall the apps on user devices. To renew a provisioning profile, log

> on to your Apple Developer account, go to **Certificates, Identifiers & Profiles**, and then select **Provisioning Profiles**.

Download the Xcode command-line tools from the Xcode Apple Developer website. macOS 10.10 does not install the tools automatically. To install the tools, follow these steps:

1. In **Applications > Utilities**, click Terminal to use the Mac command line interface (CLI).
2. Type the following command:

   ```
   xcode-select --install
   <!--NeedCopy-->
   ```

   Be sure to include two hyphens before the word install in the command.
3. After the Xcode command-line tools install, run Xcode to install any prerequisites.

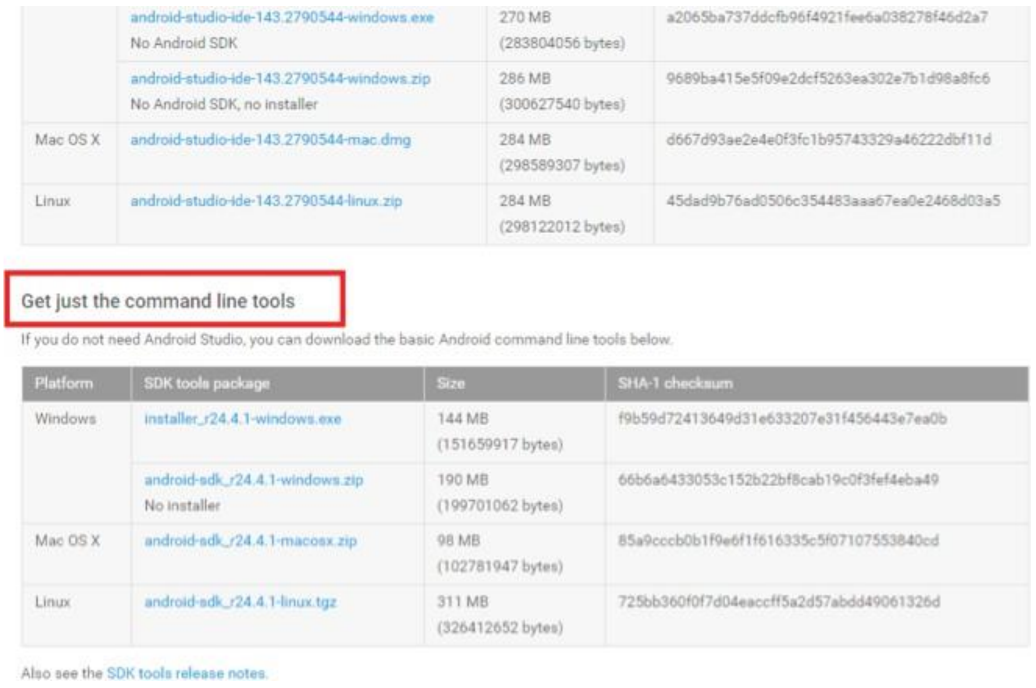
Other requirements for wrapping Android mobile apps

To wrap Android wraps, you also need a compatible Android software development kit (SDK) and a valid keystore. To download, create, and properly configure the SDK and keystore, follow these instructions:

Android software development kit

The MDX Toolkit is compatible with API Level 26 of the Android SDK.

1. Go to the Google developer website and download the Android SDK from the SDK download page. The full Android Studio is not required. You can download the command-line tools from the section near the bottom of the page.



	android-studio-ide-143.2790544-windows.exe No Android SDK	270 MB (283804056 bytes)	a2065ba737ddcfb96f4921fee6a038278f46d2a7
	android-studio-ide-143.2790544-windows.zip No Android SDK, no installer	286 MB (300627540 bytes)	9689ba415e5f09e2dcf5263ea302e7b1d98a8fc6
Mac OS X	android-studio-ide-143.2790544-mac.dmg	284 MB (298589307 bytes)	d667d93ae2e4e0f3fc1b95743329a46222dbf11d
Linux	android-studio-ide-143.2790544-linux.zip	284 MB (298122012 bytes)	45dad9b76ad0506c354483aaa67ea0e2468d03a5

Get just the command line tools

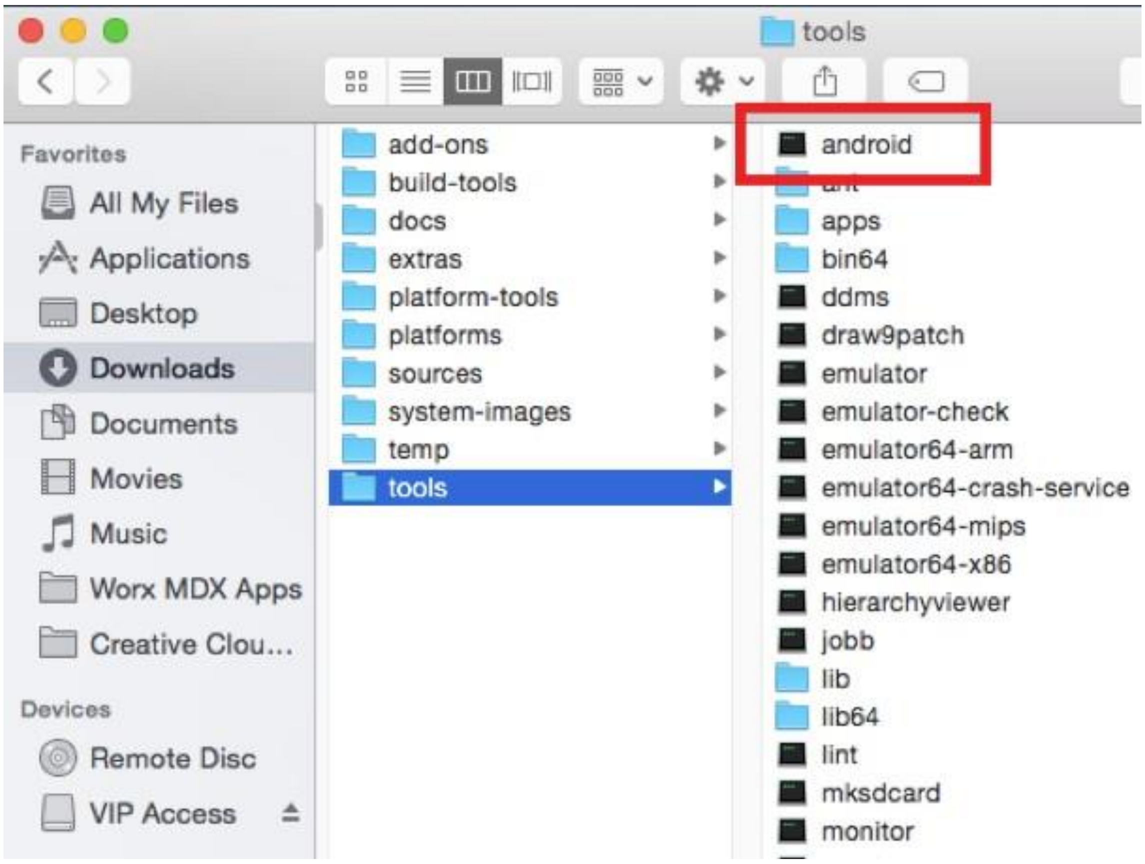
If you do not need Android Studio, you can download the basic Android command line tools below.

Platform	SDK tools package	Size	SHA-1 checksum
Windows	installer_r24.4.1-windows.exe	144 MB (151659917 bytes)	f9b59d72413649d31e633207e31f456443e7ea0b
	android-sdk_r24.4.1-windows.zip No installer	190 MB (199701062 bytes)	66b6a6433053c152b22bf8cab19c0f3fef4eba49
Mac OS X	android-sdk_r24.4.1-macosx.zip	98 MB (102781947 bytes)	85a9cccb0b1f9e6f1f616335c5f07107553840cd
Linux	android-sdk_r24.4.1-linux.tgz	311 MB (326412652 bytes)	725bb360f0f7d04eaccff5a2d57abdd49061326d

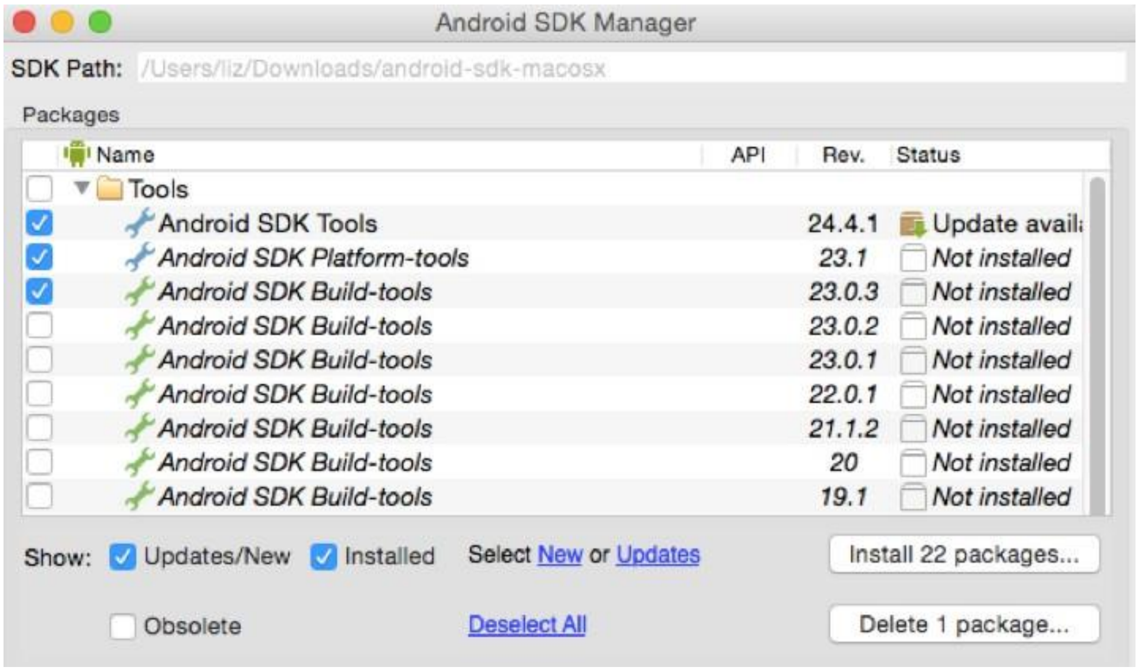
Also see the SDK tools release notes.

2. Install the latest tools, platform-tools, and build-tools. This installation requires using the Android tool in **Android SDK > tools** to start the SDK Manager:

   - Unzip the SDK file you downloaded.
   - Go to the tools folder and then click **Android** to run the SDK Manager.

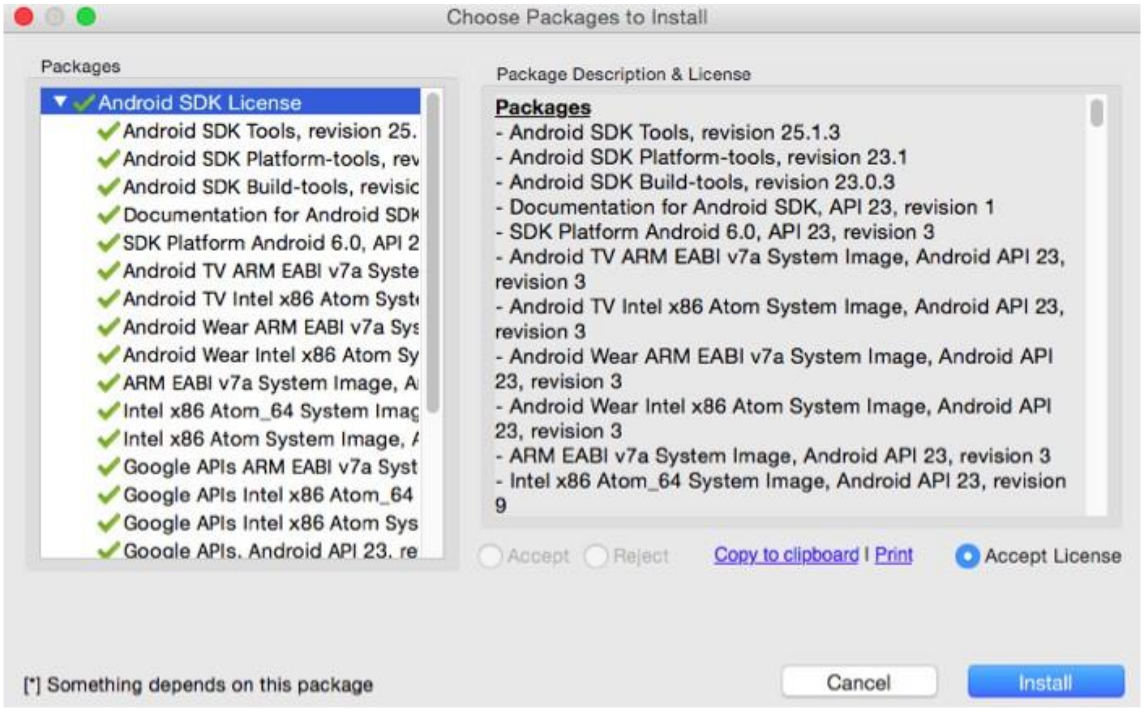


3. In the SDK Manager, select the latest versions of the following:
   - Android SDK Tools
   - Android SDK Platform
   - Android SDK Platform-tools
   - Android SDK Build-tools
4. Click **Install Packages**.


Android SDK Manager
SDK Path: /Users/liz/Downloads/android-sdk-macosx
Packages
Show: Updates/New Installed Select New or Updates Install 22 packages...
Obsolete Deselect All Delete 1 package...

	Name	API	Rev.	Status
☐	Tools			
✓	Android SDK Tools		24.4.1	Update avail
✓	Android SDK Platform-tools		23.1	Not installed
✓	Android SDK Build-tools		23.0.3	Not installed
☐	Android SDK Build-tools		23.0.2	Not installed
☐	Android SDK Build-tools		23.0.1	Not installed
☐	Android SDK Build-tools		22.0.1	Not installed
☐	Android SDK Build-tools		21.1.2	Not installed
☐	Android SDK Build-tools		20	Not installed
☐	Android SDK Build-tools		19.1	Not installed

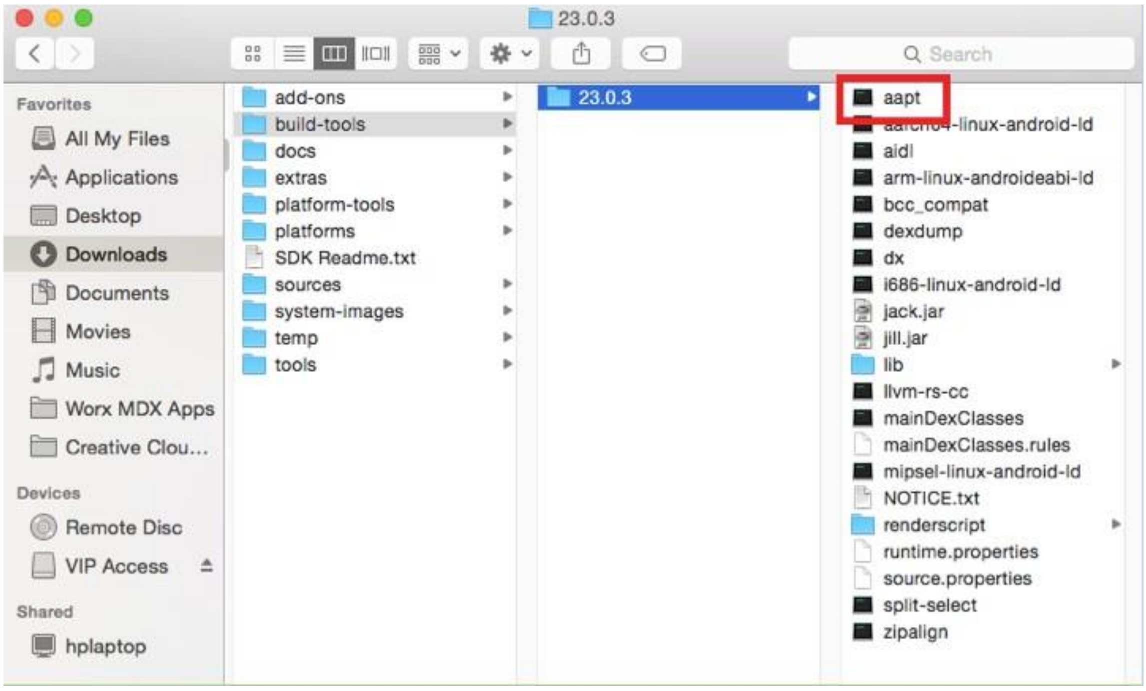
5. On the **Choose Packages to Install** screen, click **Accept License** for all the packages you are installing and then click **Install**.


Choose Packages to Install
Packages
Package Description & License
Accept
Reject
Copy to clipboard | Print
Accept License
[*] Something depends on this package
Cancel
Install

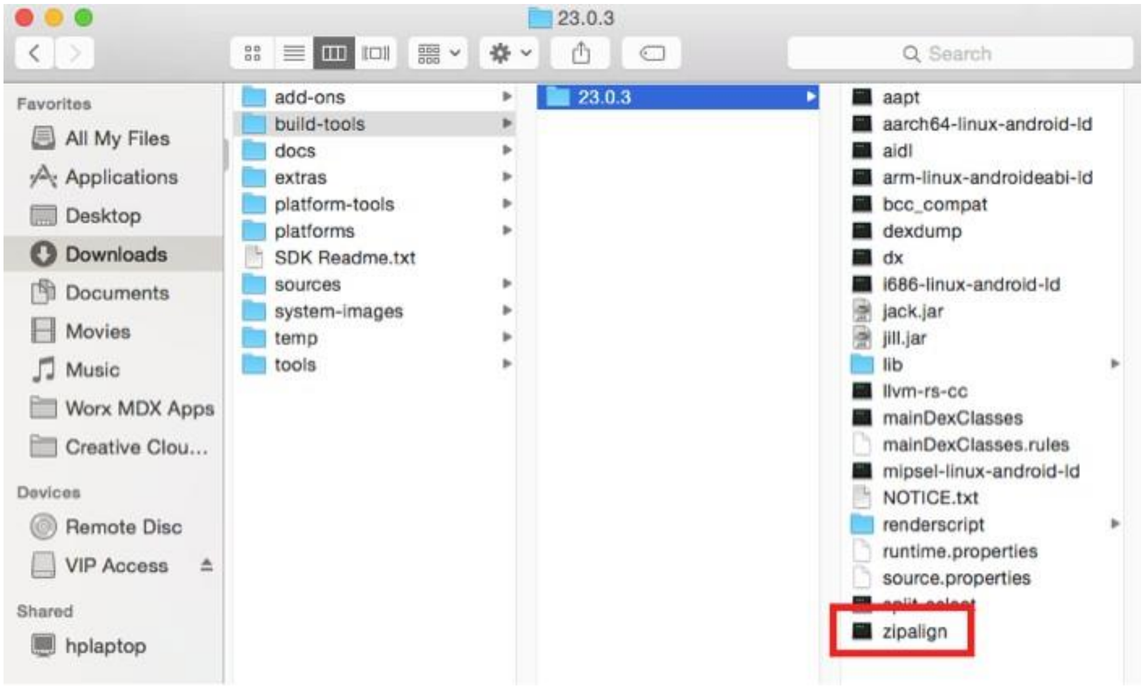
Android SDK License
- Android SDK Tools, revision 25.
- Android SDK Platform-tools, rev
- Android SDK Build-tools, revisic
- Documentation for Android SDK
- SDK Platform Android 6.0, API 2
- Android TV ARM EABI v7a Syste
- Android TV Intel x86 Atom Syst
- Android Wear ARM EABI v7a Sys
- Android Wear Intel x86 Atom Sy
- ARM EABI v7a System Image, A
- Intel x86 Atom_64 System Imag
- Intel x86 Atom System Image, A
- Google APIs ARM EABI v7a Syst
- Google APIs Intel x86 Atom_64
- Google APIs Intel x86 Atom Sys
- Google APIs, Android API 23, re

Packages
- Android SDK Tools, revision 25.1.3
- Android SDK Platform-tools, revision 23.1
- Android SDK Build-tools, revision 23.0.3
- Documentation for Android SDK, API 23, revision 1
- SDK Platform Android 6.0, API 23, revision 3
- Android TV ARM EABI v7a System Image, Android API 23, revision 3
- Android TV Intel x86 Atom System Image, Android API 23, revision 3
- Android Wear ARM EABI v7a System Image, Android API 23, revision 3
- Android Wear Intel x86 Atom System Image, Android API 23, revision 3
- ARM EABI v7a System Image, Android API 23, revision 3
- Intel x86 Atom_64 System Image, Android API 23, revision 9

6. To verify that you downloaded the appropriate SDK Tools and APIs, check that the .aapt file is in **Android SDK > build-tools > 23.0.3**.



7. When updating your SDK, you have to delete all .aaptfiles from the platform-tools folder. Ensure that the .aapt file is in build-tools only.
8. If the zipalign file is missing from build-tools, copy the file from the platform-tools folder to the build-tools folder, and then delete it from platform-tools.



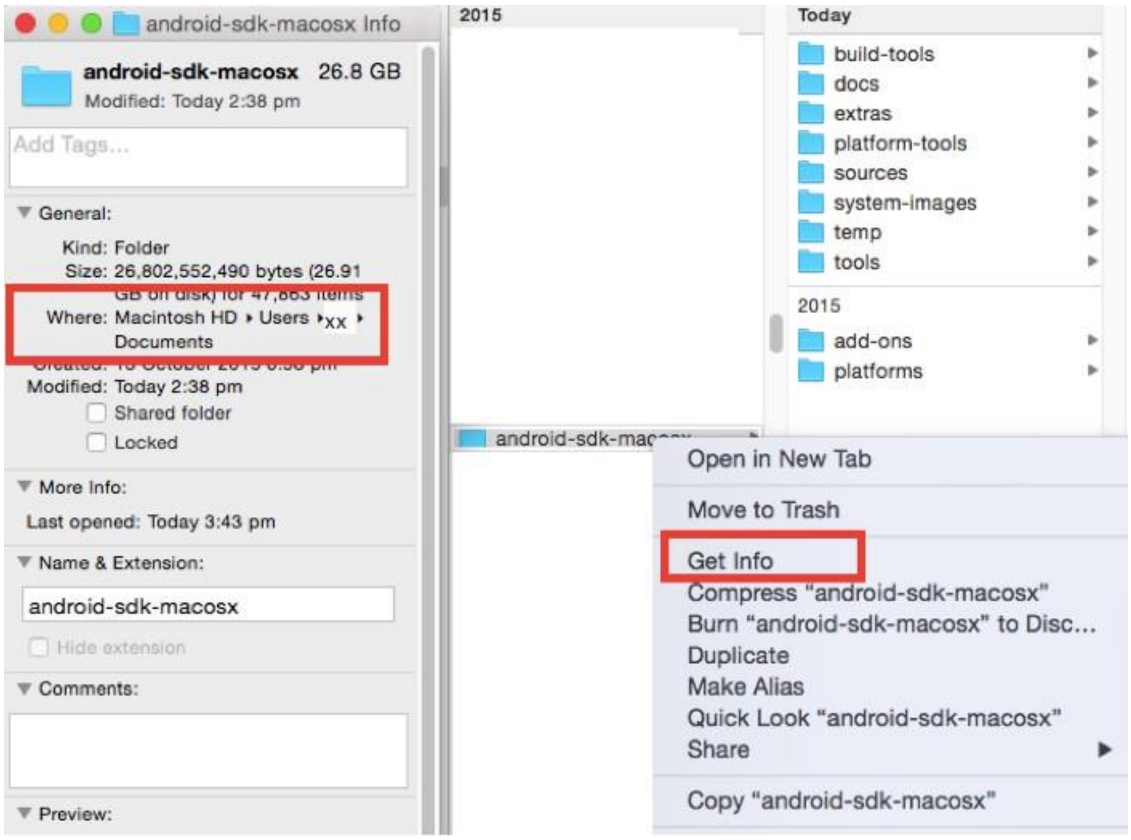
9. Add the location of the newly installed folders to the android_settings.txt file in the MDX Toolkit install folder.

10. In **Applications > Citrix > MDX Toolkit**, open the android_settings.txt file and then add the full path for the following folders:

    - **Android SDK**
    - **Android SDK > tools**
    - **Android SDK > platform-tools**
    - **Android SDK > build-tools > [version]**

    > **Note:**
    >
    > Be sure to remove the **Android SDK > apktools** path from the android_settings file, as that path is no longer required.

    To find the full path of your SDK folder, right-click on the file, select **Get Info** and then on the Info panel, review the Where information.



11. Before editing the android_settings file, make a copy of the file.

    a) Go to **Applications > Citrix > MDXToolkit > Android_settings**.

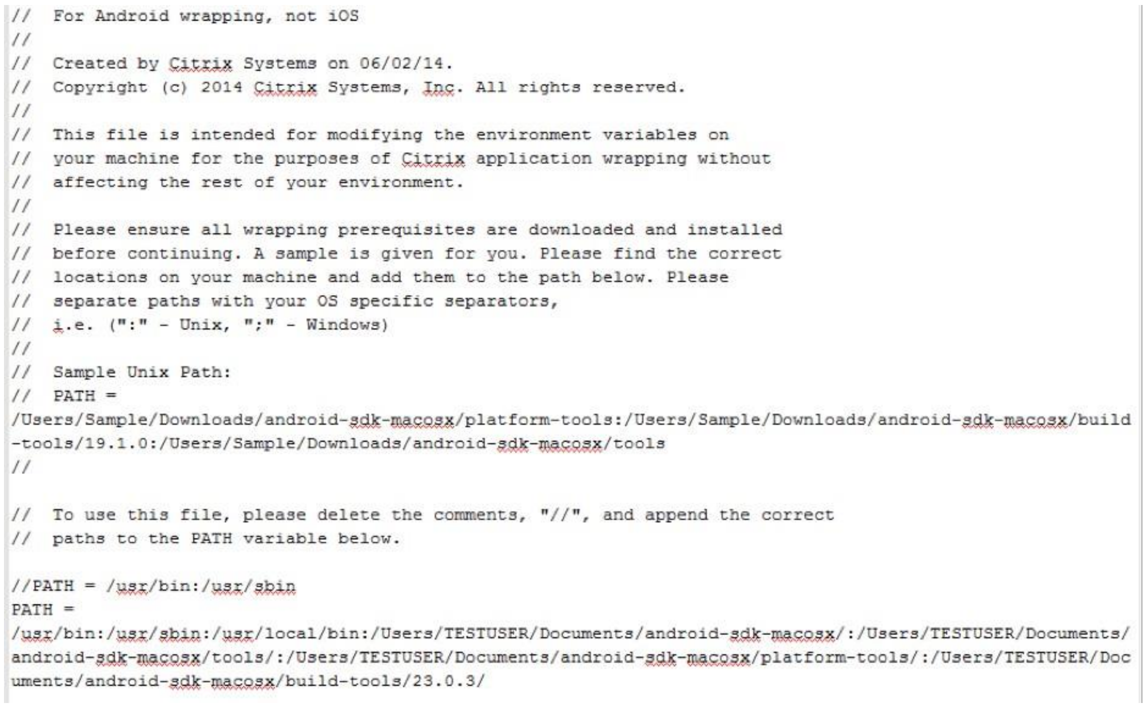
    b) Add the new paths.

    c) Save the file outside of the **Applications > Citrix > MDX Toolkit** folder.

    d) Rename the original android_settings file in the **Applications > Citrix > MDX Toolkit** folder; for example, android_settings.old.

e) Copy the new android_settings file with the added paths into the **Applications > Citrix > MDX Toolkit** folder.

The following example shows the file with the paths added:



```
//  For Android wrapping, not iOS
//
//  Created by Citrix Systems on 06/02/14.
//  Copyright (c) 2014 Citrix Systems, Inc. All rights reserved.
//
//  This file is intended for modifying the environment variables on
//  your machine for the purposes of Citrix application wrapping without
//  affecting the rest of your environment.
//
//  Please ensure all wrapping prerequisites are downloaded and installed
//  before continuing. A sample is given for you. Please find the correct
//  locations on your machine and add them to the path below. Please
//  separate paths with your OS specific separators,
//  i.e. (":" - Unix, ";" - Windows)
//
//  Sample Unix Path:
//  PATH =
/Users/Sample/Downloads/android-sdk-macosx/platform-tools:/Users/Sample/Downloads/android-sdk-macosx/build
-tools/19.1.0:/Users/Sample/Downloads/android-sdk-macosx/tools
//

//  To use this file, please delete the comments, "//", and append the correct
//  paths to the PATH variable below.

//PATH = /usr/bin:/usr/sbin
PATH =
/usr/bin:/usr/sbin:/usr/local/bin:/Users/TESTUSER/Documents/android-sdk-macosx/:/Users/TESTUSER/Documents/
android-sdk-macosx/tools/:/Users/TESTUSER/Documents/android-sdk-macosx/platform-tools/:/Users/TESTUSER/Doc
uments/android-sdk-macosx/build-tools/23.0.3/
```

Valid Keystore

A valid keystore contains digitally signed certificates that you use to sign Android apps. You create a keystore one time and retain this file for current and future wrapping. If you do not use the same keystore when wrapping new versions of apps that you've previously deployed, upgrades of those apps don't work. Instead, users must manually remove older versions before installing new versions.

A keystore can contain multiple private keys. Usually, though, the keystore has only one key.

For details about certificates, see Signing Your Applications.

Sign your apps with a key that meets the following guidelines:

- 2048-bit key size
- DSA or RSA key algorithm (-keyalg)
- Do not use MD5.

The MDX Toolkit signs apps using SHA1 to support older versions of Android. This algorithm deprecates soon in favor of SHA256. If you want to sign your app with another algorithm, use another tool.

If you don't want to use the debug keystore, create a keystore. To create a keystore, start **Terminal** and then enter the command:

keytool -genkey -keystore my-release-key.keystore -alias alias_name -keyalg RSA -key size 2048 -validity 10000

Provide the information requested, such as a password for the keystore and the domain name of your organization (example: example.com). The key is valid for 25 years.

To sign an app, use this command:

jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore *my-release-key*.keystore *my_application*.apk *alias_name*

You can now wrap Android apps. For details, see Wrapping Android apps.

Installing the MDX Toolkit

October 12, 2020

Follow these procedures to install the MDX Toolkit and XenMobile App SDK for iOS and Android.

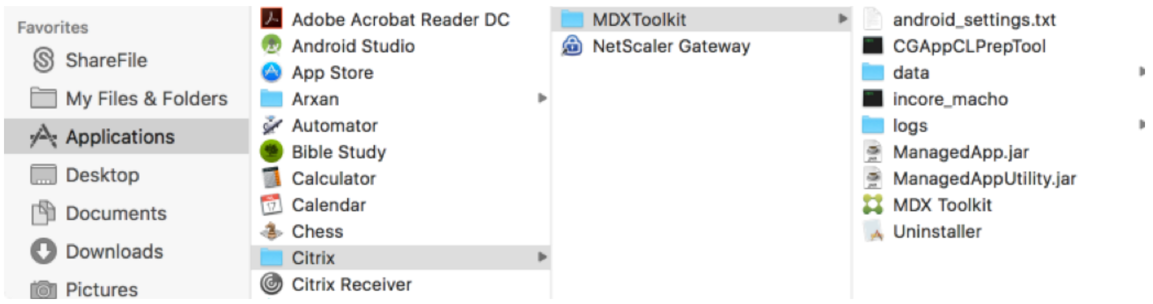
Perform the following steps on a computer running macOS. The installer includes the following tools:

- macOS tools that wrap both iOS and Android apps.
- A Java command-line tool that wraps Android apps. You can also run this tool on a Windows computer.

> **Note:**
>
> Remove the previous version of the MDX Toolkit before installing the new version. Backup Android_settings.txt before uninstalling the toolkit.

1. Go to the Citrix Endpoint Management (and Citrix XenMobile Server) page and sign in.
2. Expand **Citrix Endpoint Management Productivity Apps and MDX Toolkit**.
3. Click **MAM SDKs and Toolkit**.
4. Locate the MDX Toolkit version you want to install and then click its link to begin the download.
5. Open MDXToolkit.mpkg with the macOS Finder tool on the more recent versions of macOS and Xcode. For version requirements, see System Requirements.

   The default installation path is /Applications/Citrix/MDXToolkit.



6. If you want to run the Java command-line tool on a Windows computer, copy ManagedApp.jar and ManagedAppUtility.jar to a directory on a Windows computer that meets the Android wrapping prerequisites. For details, see System Requirements.

7. To use the GUI tool to wrap Android apps, you must update the path information in the android_settings.txt file that is installed in Applications/Citrix/MDXToolkit. If you do not complete these steps, the GUI tool will indicate that the prerequisites cannot be located.

> **Important:**
>
> When wrapping Android apps, the MDX Toolkit might fail unless the locale of the computer on which you run the MDX Toolkit is English.

   a) Copy android_settings.txt to a folder that you can write to.

   b) Edit the android_settings.txt file with any text editor. To use Vim, you can use the following command. Enter your user password when prompted. The file opens in your terminal window.

   ```
   sudo vim /Applications/Citrix/MDXToolkit/android_settings.txt
   ```

   c) Update the file with the path to the JDK and the Android SDK binaries in your environment.

   Add the following to the end of the "PATH =" line in your settings.txt file (separated by ":" on Mac/Unix, and ";" on Windows):

   ```
   PATH = /bin:/usr/bin:/usr/sbin/sbin:/<Install Location> /adt-
   bundle-mac-x86_64-20130729/sdk:/<Install Location>/adt-bundle-mac-
   x86_64-20130729/sdk/tools:<Install Location>/adt-bundle-mac-x86_64
   -20130729/sdk/platform-tools:Documents/Android SDK/apktools
   ```

   d) Save the updated file to the same name, android_settings.txt, and then copy the file to Applications/Citrix/MDXToolkit.

   You might be prompted to enter a password to copy to that folder.

The installation package includes a small utility for removing the MDX Toolkit. The utility is installed in the following location on your computer: /Applications/Citrix/CGAppPrepTool/Uninstaller.app/Contents. Double-click the utility to start the uninstaller app and then follow the prompts. When you remove the tool, you receive a message prompting you for your user name and password.

Non-compliant device behavior

December 11, 2020

When a device falls below the minimum compliance requirements, the Non-compliant device behavior policy allows you to select the action to take:

- **Allow app:** Allow the app to run normally.
- **Allow app after warning:** Warn the user that an app does not meet the minimum compliance requirements and allows the app to run. This setting is the default value.
- **Block app:** Block the app from running.

The following criteria determine whether a device meets the minimum compliance requirements.

Devices running iOS:

- **iOS 10:** An app is running an operating system version that is greater than or equal to the specified version.
- **Debugger access:** An app does not have debugging enabled.
- **Jailbroken device:** An app is not running on a jailbroken device.
- **Device passcode:** Device passcode is ON.
- **Data sharing:** Data sharing is not enabled for the app.

Devices running Android:

- **Android SDK 24 (Android 7 Nougat):** An app is running an operating system version that is greater than or equal to the specified version.
- **Debugger Access:** An app does not have debugging enabled.
- **Rooted devices:** An app is not running on a rooted device.
- **Device lock:** Device passcode is ON.
- **Device encrypted:** An app is running on an encrypted device.

Wrapping iOS mobile apps

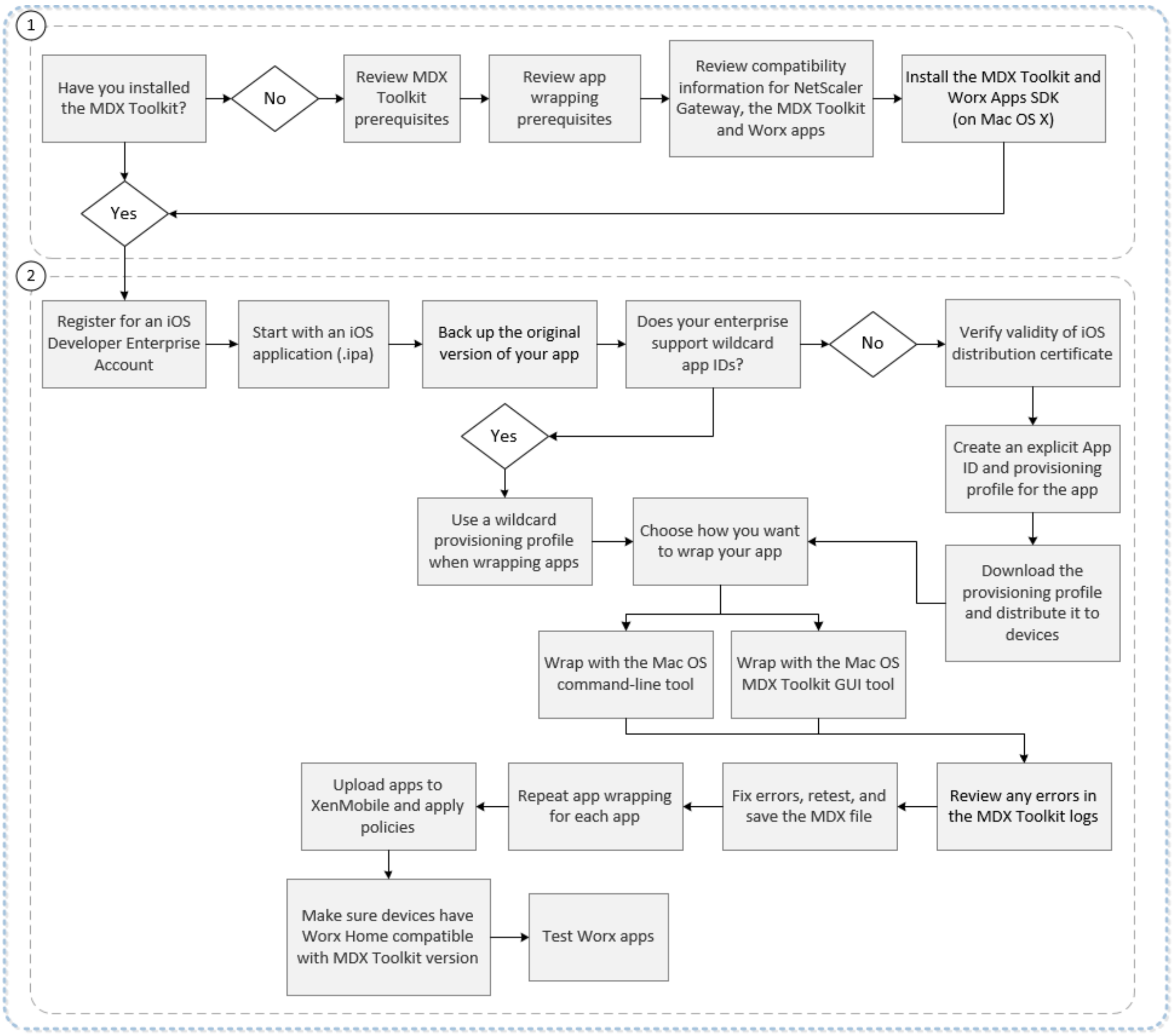
January 3, 2023

This article describes how Citrix Endpoint Management administrators wrap third-party enterprise apps and how developers wrap ISV apps. To wrap iOS mobile apps:

- Use MDX Service. For details, see MDX Service.
- Use the MDX Toolkit, which includes a macOS graphical interface tool and a macOS command-line tool. The macOS command-line tool has customization options, can be referenced from scripts that automate the app wrapping process, and lets you preset some MDX policies.

The file type for a wrapped app is .mdx. You upload the .mdx file to the Endpoint Management console where you configure specific app details and policy settings that the Endpoint Management Store enforces. When users sign on, the app appears in the store. Users can then subscribe, download, and install the app on their device.

The following figure provides an overview of the app wrapping steps, from installation of the MDX Toolkit through testing mobile productivity apps. Related topics are listed under the diagram.


1
Have you installed the MDX Toolkit?
No
Review MDX Toolkit prerequisites
Review app wrapping prerequisites
Review compatibility information for NetScaler Gateway, the MDX Toolkit and Worx apps
Install the MDX Toolkit and Worx Apps SDK (on Mac OS X)
Yes
2
Register for an iOS Developer Enterprise Account
Start with an iOS application (.ipa)
Back up the original version of your app
Does your enterprise support wildcard app IDs?
No
Verify validity of iOS distribution certificate
Yes
Create an explicit App ID and provisioning profile for the app
Use a wildcard provisioning profile when wrapping apps
Choose how you want to wrap your app
Download the provisioning profile and distribute it to devices
Wrap with the Mac OS command-line tool
Wrap with the Mac OS MDX Toolkit GUI tool
Upload apps to XenMobile and apply policies
Repeat app wrapping for each app
Fix errors, retest, and save the MDX file
Review any errors in the MDX Toolkit logs
Make sure devices have Worx Home compatible with MDX Toolkit version
Test Worx apps

For details about number one, see:

- System requirements
- Other requirements for wrapping iOS mobile apps

- Endpoint Management Compatibility
- Installing the MDX Toolkit

For details about number two, see:

- Creating Provisioning Profiles
- App Upgrades
- Policies and mobile productivity apps
- Enterprise App Wrapping Using the Graphical Interface
- Enterprise iOS App Wrapping Using the Command Line
- Command Options
- Presetting MDX Policies for iOS Apps
- Identifying iOS App Wrapping Errors
- Collecting System Logs on iOS Devices
- To add an MDX app to Citrix Endpoint Management

> Important:
>
> Make sure that user devices are updated with a version of Secure Hub that is compatible with the version of MDX Toolkit used to wrap apps. Otherwise, users see an error message about the incompatibility. For details, see Endpoint Management compatibility.

Deploying iOS devices through Apple DEP

Enroll in the Apple Deployment Program to take advantage of the Apple Device Enrollment Program (DEP). You can use Apple DEP to deploy and manage iOS and macOS devices in Citrix Endpoint Management. For more information, including how to enroll in the Apple Deployment Program, see Deploy iOS and macOS devices through Apple DEP.

Creating provisioning profiles

Apps that run on a physical iOS device, other than apps in the Apple App Store, must be signed with a provisioning profile and a corresponding distribution certificate. There are two kinds of developer programs for distribution:

- The iOS Developer Program (Ad Hoc)
- The iOS Developer Enterprise Program. To wrap apps, Citrix recommends using the Enterprise program. You can enroll in the program from the Apple website.

The Enterprise profile allows you to run an app on unlimited devices. The Ad Hoc profile allows you to run an app on up to about 100 devices.

Apple no longer supports the use of wildcard App IDs for new Enterprise accounts. If your Enterprise account does not support wildcard App IDs, you must create multiple explicit App IDs and provisioning profiles, as follows.

1. Verify that you have a valid iOS distribution certificate.
2. From the Apple Enterprise Developer portal, create an explicit App ID for each app you plan to wrap with the MDX Toolkit. An example of an acceptable App ID is: com.CompanyName.ProductName.
3. From the Apple Enterprise Developer portal, go to **Provisioning Profiles > Distribution** and create an in-house provisioning profile. Repeat this step for each App ID created in the previous step.
4. Download all provisioning profiles.

If your Apple Enterprise account supports wildcard App IDs, you can continue to use a wildcard provisioning profile to wrap apps. However, if you use Apple Push Notification service (APNs) for notifications when Secure Mail is in the background, you must use an explicit provisioning profile and App ID.

Any device on which you want to install the MDX app needs to have the provisioning profile on the device. You can distribute the profile to user devices by using an email attachment. Users can add the profile on their iOS device by clicking the attachment.

For details about provisioning profiles and distribution certificates, see the Apple Developer Account Help.

For more information about deploying the provisioning profile to iOS devices and on handling expired profiles, see the Endpoint Management article on the Provisioning profile device policy.

App upgrades

> Important:
>
> Before you upgrade apps, be aware how changes to App IDs or the use of a partial wildcard App ID provisioning profile impact app upgrades.

- Previously wrapped apps upgrade in place unless the App ID has changed. For example, if you change a bundle ID from com.citrix.mail to com.example.mail, there is no upgrade path. The user must reinstall the app. A device considers the app as a new app. The new and prior versions of the app can both reside on the device.
- If you use a partial provisioning profile, such as com.xxxx, to wrap an app with a bundle ID that includes com.citrix, we recommend the following: Remove the installed MDX-wrapped apps and install the apps wrapped with the latest MDX Toolkit. As a result of a bundle ID change from com.citrix.mail to com.example, users must reinstall the app.
- An in-place upgrade succeeds when the following is true: If an app was wrapped with a full wildcard App ID, and the new version of the app has an App ID that matches the installed app.

Policies and mobile productivity apps

> Note:
>
> The MDX 10.7.5 release is the final release that supports the wrapping of mobile productivity apps. You cannot use releases of MDX 10.7.10 and later, or the MDX Service, to wrap mobile productivity apps 10.7.5 or later. You must access mobile productivity apps from the public app stores.

Citrix provides a generic set of default policies that apply to all mobile productivity apps and a set of specific policies for some of the mobile productivity apps. Policy file names are based on the bundle ID. By default, the policy file name for a Citrix Endpoint Management app is in the form com.citrix.app_policy_metadata.xml, where app is a name such as "mail".

If you have an Apple Enterprise account that does not support wildcard App IDs, do the following: Change the company identifier in the bundle ID when you wrap an Endpoint Management app. For example, the bundle ID for Secure Mail is com.citrix.mail. Replace "citrix" in that identifier with your company identifier. If your company identifier is "example", the bundle ID is com.example.mail. When you wrap that app, the policy file name is com.example.mail_policy_metadata.xml.

To determine which policy file to apply to an app, the MDX Toolkit looks for files in the following order and uses the first file it finds:

1. A file name that matches your bundle ID, such as com.example.mail_policy_metadata.xml, as described in the preceding example.
2. A file name that matches the original bundle ID, such as com.citrix.mail_policy_metadata.xml.
3. A file name that matches the generic default policy file, policy_metadata.xml.

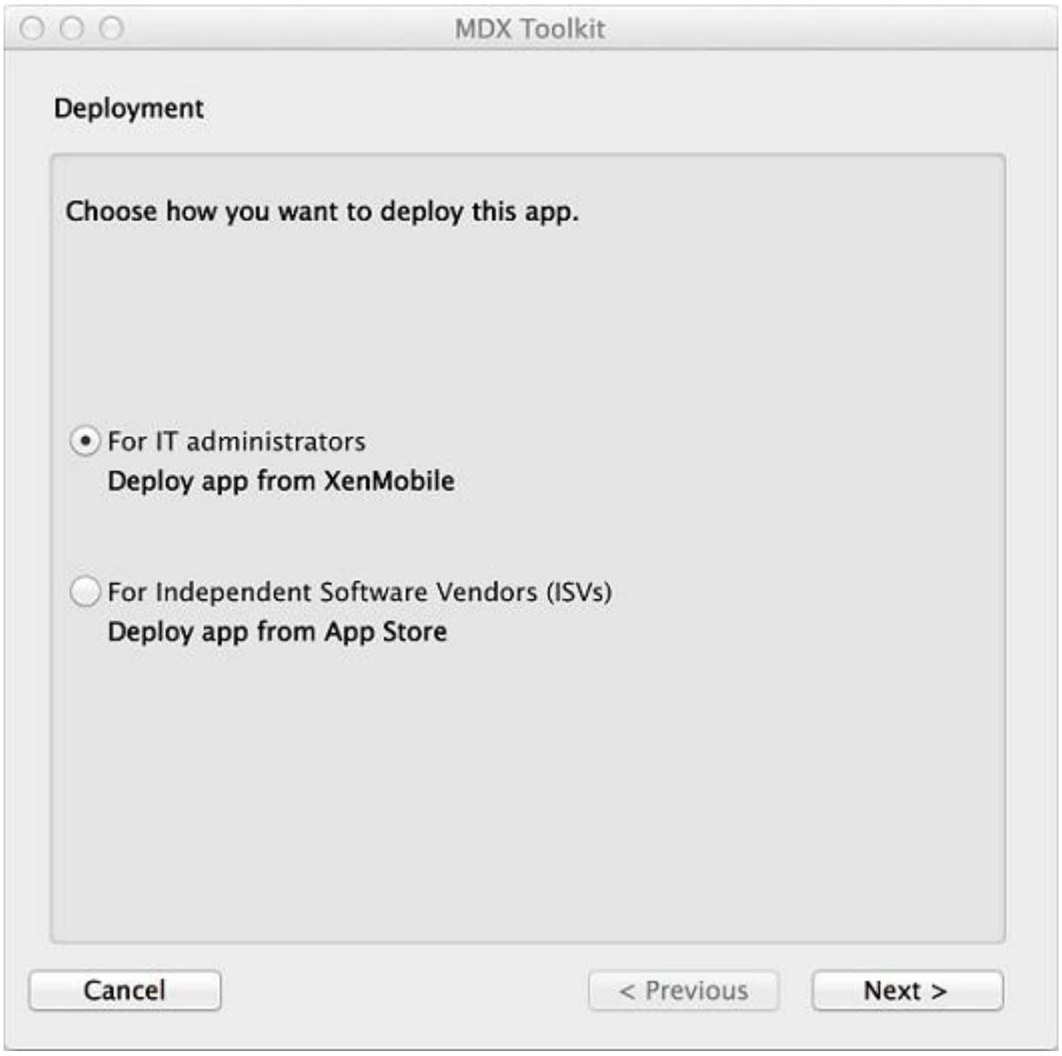
Create your own set of policy defaults for a specific Citrix Endpoint Management app by modifying the files that match your bundle ID or the original bundle ID.

Enterprise app wrapping using the graphical interface

The following steps describe the general process for wrapping an enterprise app that you deploy from Endpoint Management. The general process for ISV app wrapping is described in ISV App Wrapping Using the Graphical Interface.

> Important:
>
> Both the private key and the certificate must be installed on the Keychain Access of your Mac before using the graphical interface to wrap iOS apps. If the associated distribution certificate does not have the private key installed into Keychain Access, the graphical interface does not pre-populate the
> iOS Distribution Certificate list. For details, see "Repairing Your Keychain when the Toolkit Can't Find a Distribution Certificate," later in this article.

1. Before you use the toolkit to wrap apps, be sure to back up the original version of those apps so you can return to them if needed.
2. Start the MDX Toolkit from your iOS Applications folder, select **For IT administrators**, and then click **Next**.



MDX Toolkit

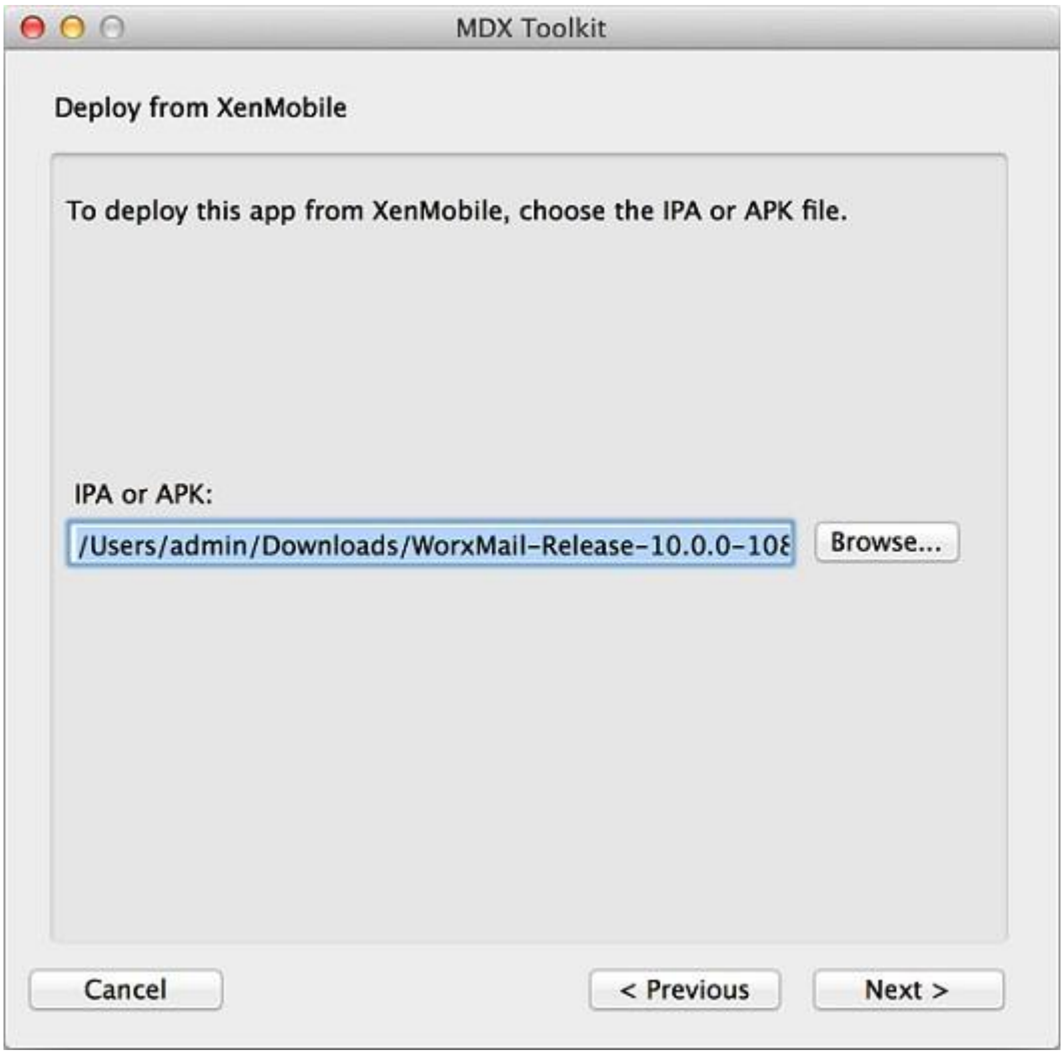
Deployment

Choose how you want to deploy this app.

(●) For IT administrators
Deploy app from XenMobile

( ) For Independent Software Vendors (ISVs)
Deploy app from App Store

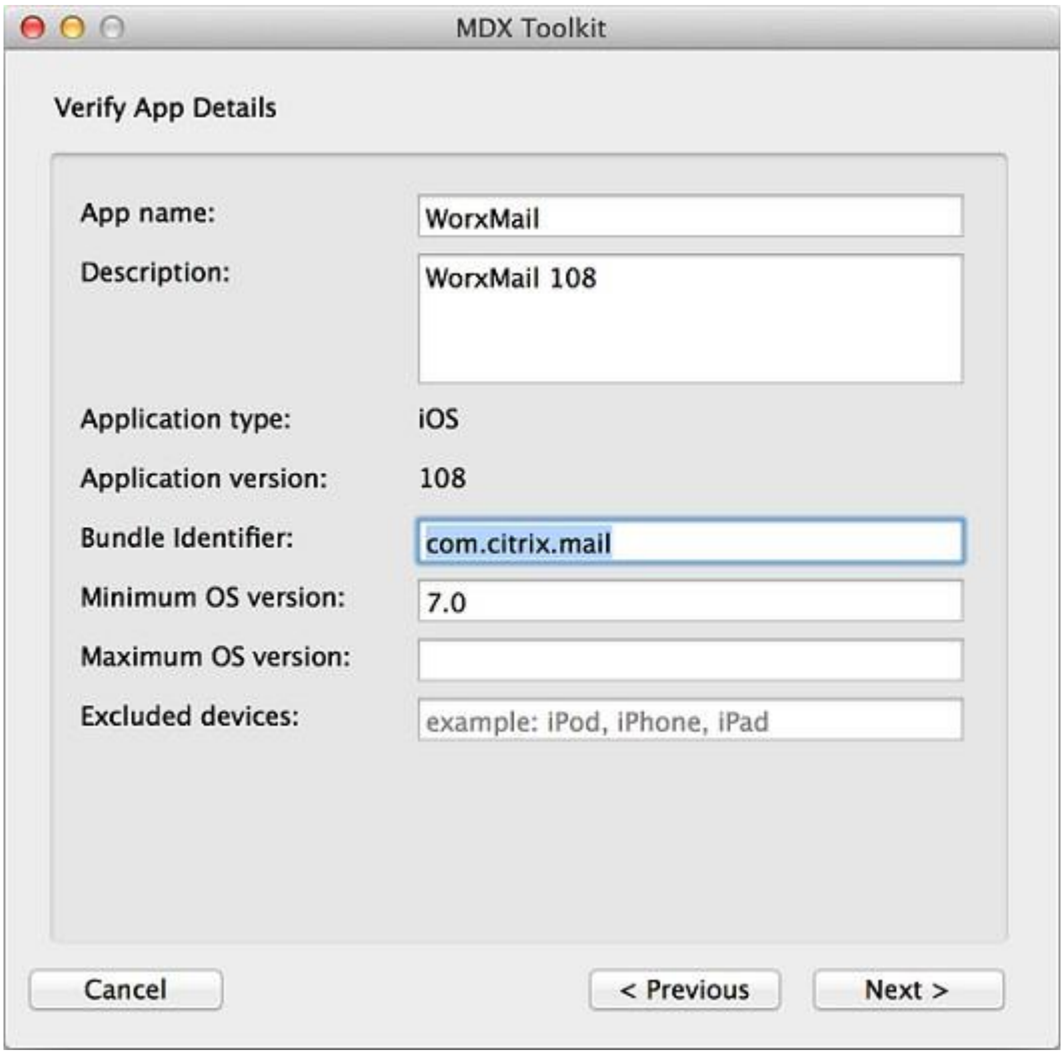
3. Click **Browse**, select the file, and then click **Next**.



To deploy this app from XenMobile, choose the IPA or APK file.

IPA or APK: /Users/admin/Downloads/WorxMail-Release-10.0.0-108

4. The **Verify App Details** screen shows information obtained from the app. As needed, change the pre-filled information. Optionally, specify a minimum and maximum OS version and list the device types on which the app is not allowed to run. You can also change the app details after uploading the app to Citrix Endpoint Management.

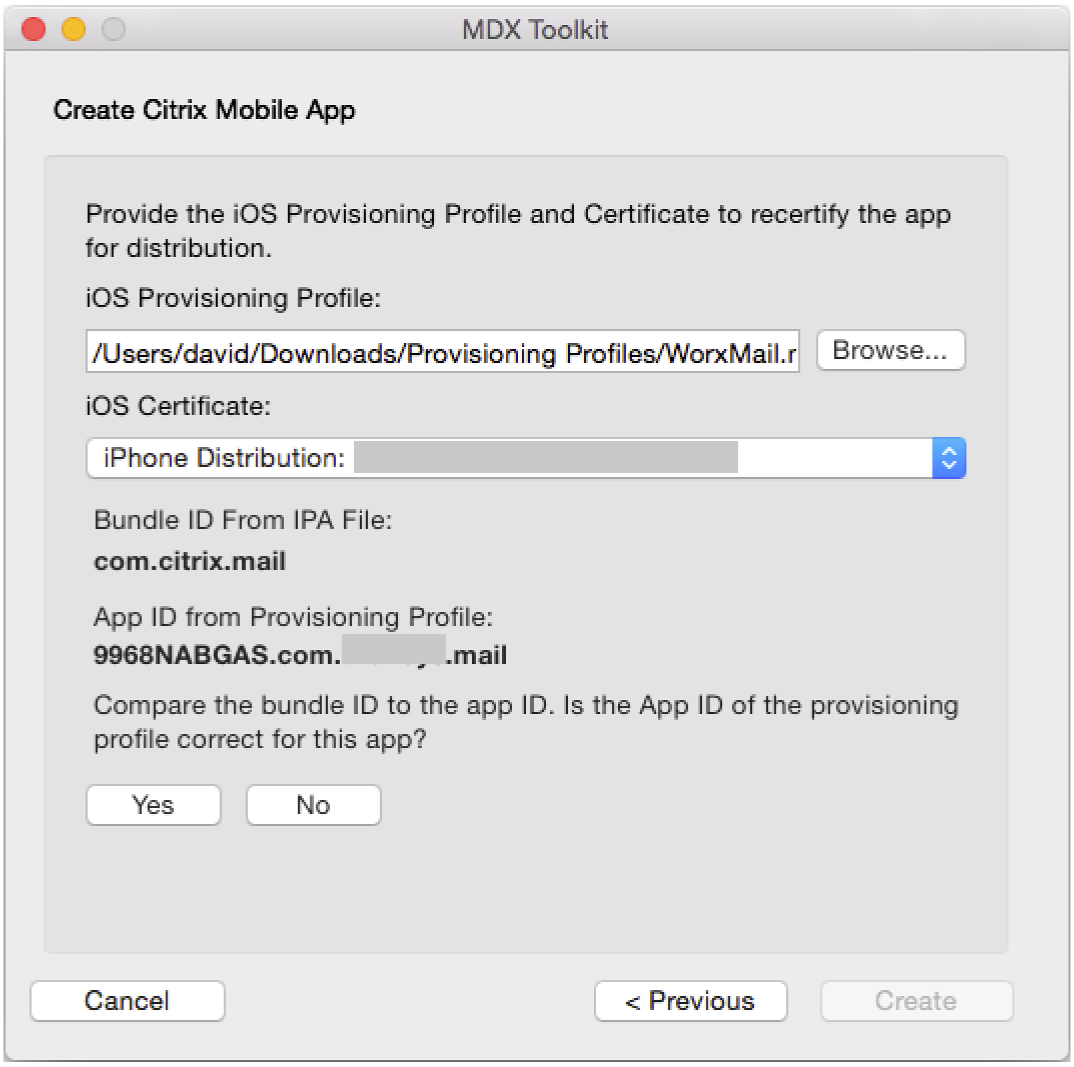


Verify App Details

App name:	WorxMail
Description:	WorxMail 108
Application type:	iOS
Application version:	108
Bundle Identifier:	com.citrix.mail
Minimum OS version:	7.0
Maximum OS version:	
Excluded devices:	example: iPod, iPhone, iPad

5. In the **Create Citrix Mobile App** screen, click **Browse**, select the provisioning profile, and select a distribution certificate. If the iOS Certificate list is empty, repair the keychain on the machine where you are running the MDX Toolkit. For details, see “Repairing Your Keychain when the Toolkit Can’t Find a Distribution Certificate,” later in this article.
6. If you selected a provisioning profile that has an explicit app ID, the tool prompts you to confirm the app ID. For example, the bundle ID for a Citrix Endpoint Management app is com.citrix.ProductName. The provisioning profile that you use must include your company identifier instead of “citrix”.

   After you click **Yes**, click **Create**.



MDX Toolkit

Create Citrix Mobile App

Provide the iOS Provisioning Profile and Certificate to recertify the app for distribution.

iOS Provisioning Profile:

/Users/david/Downloads/Provisioning Profiles/WorxMail.r

iOS Certificate:

iPhone Distribution: [redacted]

Bundle ID From IPA File:

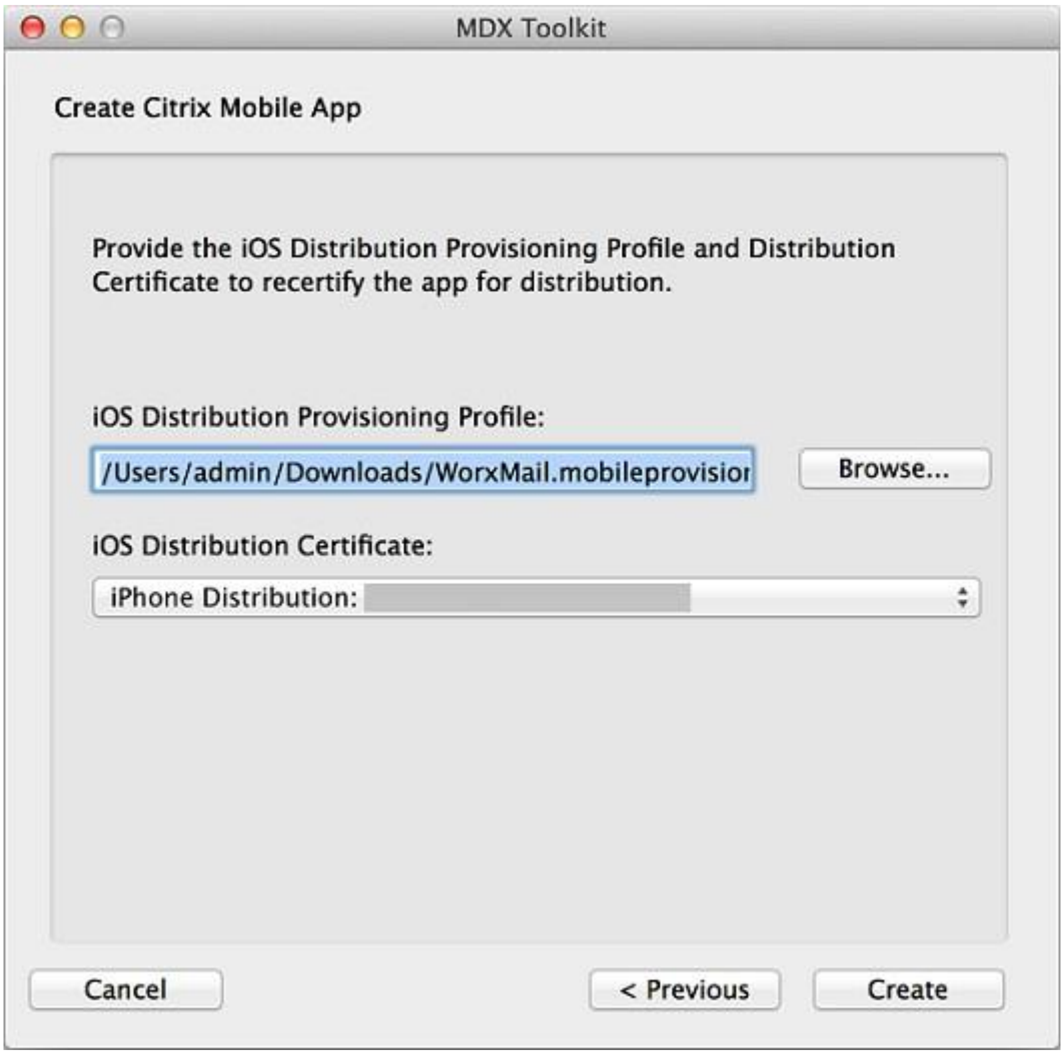
com.citrix.mail

App ID from Provisioning Profile:

9968NABGAS.com.[redacted].mail

Compare the bundle ID to the app ID. Is the App ID of the provisioning profile correct for this app?

7. If you selected a provisioning profile that has a wildcard app ID, the tool shows a list of available app IDs. If the app ID you want to use isn't listed, choose a different provisioning profile. After you choose an app ID, click **Create**.



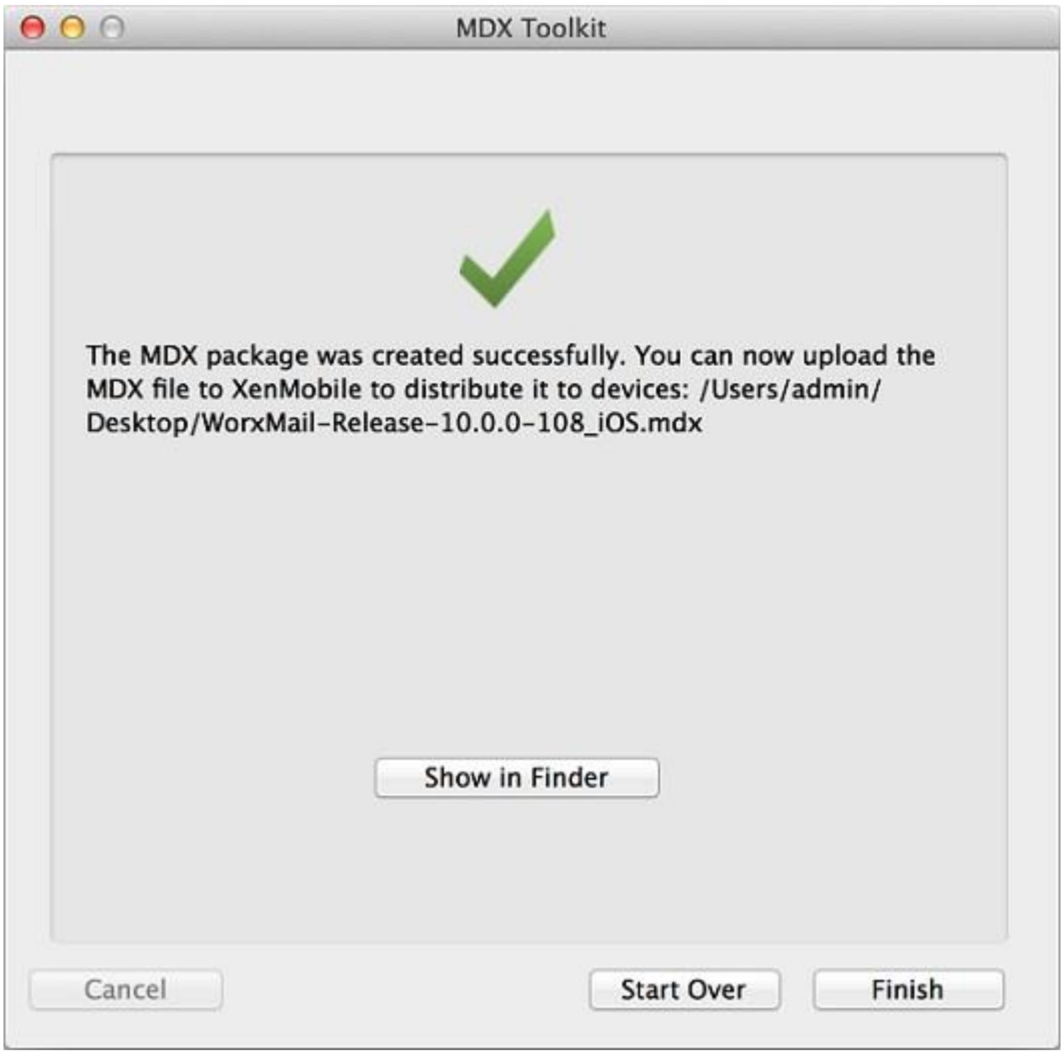
Create Citrix Mobile App

Provide the iOS Distribution Provisioning Profile and Distribution Certificate to recertify the app for distribution.

iOS Distribution Provisioning Profile: /Users/admin/Downloads/WorxMail.mobileprovisior

iOS Distribution Certificate: iPhone Distribution: [redacted]

8. The toolkit lets you know when the MDX package is created. To wrap another app, click **Start Over**.



The MDX package was created successfully. You can now upload the MDX file to XenMobile to distribute it to devices: /Users/admin/Desktop/WorxMail-Release-10.0.0-108_iOS.mdx

The toolkit appends _iOS to the end of the filename of a wrapped iOS app.

Enterprise iOS app wrapping using the command line

> Note:
>
> Be sure to obtain third-party apps directly from the app vendor. iOS apps downloaded from the Apple store are encrypted and cannot be wrapped.

Before you use the toolkit to wrap apps, be sure to back up the original version of those apps so you can return to them if needed.

The following example shows a basic app wrapping command using default settings. Modify the bold information for your specific system. The trailing backslash signifies the command continues to the next line. Remove these symbols before running the command.

To perform these commands, navigate to the /Applications/Citrix/MDXToolkit/ directory in your command line.

A basic iOS wrapping command line is as follows.

```
./CGAppCLPrepTool \
Wrap \
 – Cert CERTIFICATE \
 – Profile PROFILE \
-bundleID ID \
 – in INPUT_FILE \
 – out OUTPUT_FILE
<!--NeedCopy-->
```

The following is an example of this command-line option.

```
./CGAppCLPrepTool \
Wrap \
 – Cert “iPhone Developer: Joe Admin (12MMA4ASQB)” \
 – Profile “team_profile.mobileprovision” \
-bundleID “com.CompanyABC.Sample” \
 – in “~/Desktop/SampleApps/Sample.ipa” \
 – out “~/Desktop/SampleApps/Sample.mdx”
<!--NeedCopy-->
```

Examples of options you may add to the preceding command include:

`-appName “Wrapped Sample app”`

`-appDesc “This is my newly wrapped iOS application.”`

Both of those options default to the value read from the app, if possible.

For details about the options, see Command Options next. For inline documentation, use the -help option.

Command options

Wrap command

- **Help:** Displays Help for this command.
- **In:** Required. Path and file name of the app you are wrapping.
- **Out:** Optional. Path and file name for the resulting .mdx file. If this option is omitted, the file has the same path and file name as the input file and has an .mdx extension.
- **outBundle:** Required when generating an .ipa file for uploading to Intune. Path and file name for the resulting .ipa file.
- **Cert:** Required. Name of the certificate to use to sign the app.
- **Profile:** Required. Name of the provisioning profile to use to sign the app.

- **bundleID:** Required for Enterprise accounts that do not support wildcard App IDs. This value is your Apple bundle ID. The MDX Toolkit verifies whether the bundle ID and provisioning profile are compatible.
- **Upgrade:** This option is intended for legacy apps and will be deprecated. Used for in-place upgrades when you use a partial wildcard provisioning profile. This option ensures that the new binary is signed with the same entitlement as the prior version. If the entitlements do not match, attempts by users to install the upgrade from Secure Hub fail.
- **AppName:** Optional. App name, obtained from the app if possible.
- **AppDesc:** Optional. App description, obtained from the app if possible.
- **MinPlatform:** Optional. Minimum supported platform version. Defaults to blank.
- **MaxPlatform:** Optional. Maximum supported platform version. Defaults to blank.
- **ExcludedDevices:** Optional. List of device types on which the app is not allowed to run. Defaults to blank.
- **PolicyXML:** Optional. Replacement XML policy definition file and path. Defaults to the built-in policy definitions. Example: -policyxml /Applications/Citrix/MDXToolkit/data/policy_metadata.xml. For details, see Presetting MDX Policies for iOS Apps next.
- **useNetworkOnlylib:** This option wraps the application with the network-only lighter-weight version of the MDX dynamic library. An application wrapped using this option can only be managed by Intune or run unmanaged. It cannot be managed by MDX.
- **LogFile:** Optional. Name of the log file.
- **LogWriteLevel:** Optional. Log level, 1 through 4.
- **LogDisplayLevel:** Optional. Log level for standard output, 0 through 4.

Sign command

- **Help:** Displays Help for this command.
- **In:** Required. Path and file name of the app you are wrapping.
- **Out:** Optional. Path and file name for the resulting .mdx file. If this option is omitted, the file has the same path and file name as the input file and has an .mdx extension.
- **Cert:** Required. Name of the certificate to use to sign the app.
- **Profile:** Required. Name of the provisioning profile to use to sign the app.

setinfo command

- **Help:** Displays Help for this command.
- **In:** Required. Path and file name of the app to be modified.
- **Out:** For setinfo, the output path or file name must differ from the original.
- **AppDesc:** Optional. App description. Remains unchanged if not specified.
- **MinPlatform:** Optional. Minimum supported SDK level. Remains unchanged if not specified.
- **MaxPlatform:** Optional. Maximum supported SDK level. Remains unchanged if not specified.

- **ExcludedDevices:** Optional. List of device types on which the app is not allowed to run. Remains unchanged if not specified.
- **StoreURL:** Optional. URL of the app in the app store. Remains unchanged if not specified.
- **PolicyXML:** Optional. Replacement XML policy definition file and path. Defaults to the built-in policy definitions. Example: -policyxml /Applications/Citrix/MDXToolkit/data/policy_metadata.xml. For details, see Presetting MDX Policies for iOS Apps, next.

Presetting MDX policies for iOS apps

For apps that you wrap with the MDX Toolkit command-line tool, you can preset some MDX policies. You can also configure policies in the Citrix Endpoint Management console when you add the apps.

1. Update policy values in the policy XML file.

   The MDX Toolkit installer creates this policy file: Applications/Citrix/MDXToolkit/data/policy_metadata.xml

   > Note:
   >
   > The policies files for iOS and Android differ. To preset policies for both of those platforms, you must update their respective policy XML files.

2. When you wrap the app with the command line, include

   ```
   -policyxml /Applications/Citrix/MDXToolkit/data/policy_metadata.xml
   ```

Identifying iOS app wrapping errors

If you encounter an error when wrapping an iOS app, you can use the MDX Toolkit logs to identify the error. You must have administrator rights to view the MDX Toolkit logs.

When you run the MDX Toolkit, the tool saves a log file to the following location: **Applications > Citrix > MDXToolkit > Logs > Citrix.log**. By default, the tool saves warnings and errors in the log.

If an error occurs for an iOS app, a command line with arguments appears at the end of the log. You can copy the command line and run it in Terminal. To do that, in **Applications > Utilities**, click **Terminal**, and use the Mac command-line interface to evaluate the command. You may need to refer to the app requirements to evaluate the error.

When you use the command-line tool to run the wrapping process, you can specify the following information in the command line: The log file location, log display level, and log write level. You can also specify verbose logging level and a different log file in the command line.

Selecting the correct provisioning profile

When you wrap a mobile iOS app, you might receive a warning indicating that the app was wrapped successfully, but may contain errors. Errors can occur if the provisioning profile you chose differs from the provisioning profile the app originally used.

The MDX Toolkit can alert you about certain provisioning profile issues. For example, your app may require one or more of the following functions:

- iCloud app that enables the use of iCloud data storage for your iOS app
- Push notification that uses the Apple push notification service to deliver messages to the iOS device
- Special Keychain Access Groups Entitlement to access the keychain item for another app

The logs show the missing key and value pairs for the app. For each key and value pair, you can decide whether you want to fix the error. If you do not fix the error, the app may not function correctly. Also, depending on the key and value pair, you need to check if you can fix your provisioning profile. Occasionally, you might not be able to fix the provisioning profile and can release the app with the defect.

For details about provisioning profiles, see the Apple Developer site.

Repairing your keychain when the Toolkit can't find a distribution certificate

If the MDX Toolkit does not recognize your iOS Distribution Certificate, there might be an issue between your iCloud Keychain and the keychain on the computer running the MDX Toolkit. To repair your local keychain, follow these steps.

1. On your Mac, in System Preferences, tap **iCloud**.
2. Clear the Keychain check box.

   This step removes your locally synchronized keychain from iCloud.
3. Open **Keychain Access**, which is in the Utilities folder within the Applications folder.
4. Delete the iOS Developer Certificate used to sign your wrapped apps. This certificate is typically the "iPhone Distribution: Company Name" certificate with an associated private key.
5. From the Keychain Access menu, choose Keychain First Aid.
6. In the Keychain First Aid dialog box, tap **Repair** and then **Start**.
7. After the repair completes, tap **Verify** and then **Start**.
8. If the repair is successful, import your iOS Distribution Certificate again into the Keychain Access app.
9. Start the MDX Toolkit. The iOS Distribution Provisioning Profile and iOS Distribution Certificate fields should contain your information.

10. As needed, resync you keychain to iCloud: In System Preferences, tap iCloud and then select the Keychain check box.

Resigning apps that contain the MDX App SDK

If your app already contains the MDX App SDK built into it using Xcode, you need to resign the app with your enterprise certificate or provisioning profile. The following is a sample of the **Sign** Command.

```
$ /Applications/Citrix/MDXToolkit/CGAppCLPrepTool Sign -help

Command Line Interface for MDX Toolkit, version 10.4.1.290 (Env:Test)

2016-09-29 15:21:45.284 CGAppCLPrepTool[88453:5477658]

----------------------------------------------------------------------------

Sign Command

----------------------------------------------------------------------------

CGAppCLPrepTool Sign -in INPUTFILE -out OUTPUTFILE -Cert CERTIFICATE -Profile PROFILE

-Cert CERTIFICATE           ==> (Required)Name of the certificate to sign the app with

-Profile PROFILE            ==> (Required)Name of the provisioning profile to sign the app with

-in INPUTFILE               ==> (Required)Name of the input app file, ipa/mdx file

-out OUTPUTFILE             ==> (Optional)Name of the output app, ipa(if ipa is input)/mdx file

-upgrade                    ==> (Optional)Preserve in-place upgrade capabilty (not recommended for new apps)

--------------------EXAMPLE--------------------------

Sign -Cert "iPhone Distribution: Company Name" -Profile "distributionprovisioanl.mobileprovision" -in "/Users/user1/Archives/citrix.ipa"
<!--NeedCopy-->
```

Collecting System Logs on iOS devices

You can collect System Logs on iOS devices either by using the iPhone Configuration Utility tool or Xcode. You can then email the files to Citrix support for help troubleshoot issues with apps.

To use a Configuration Utility tool to collect System Logs on iOS devices

1. Download and install the Apple Configurator (previously the iPhone Configuration Utility) tool from Apple. You can use the tool on both the iPhone and IPad.
2. Ensure that your device meets the system requirements and supported languages.
3. Run the installer and follow the prompts to complete the wizard.
4. Open the Configurator tool.
5. Under Devices, click your device.
6. Click **Console** and then click **Clear** to clear existing logs.
7. Reproduce the issue, click **Save Console As** and then attach and email the logs to support.

To use Xcode to collect logs on iOS devices

1. On the Mac, click **Finder**, click **Go** and then click **Utilities**.
2. In the **Utilities** folder, double-click **Console**.
3. In the Console, under **Devices**, click the iOS device from which you want the console logs.
4. Reproduce the issue.
5. In the Console, do one of the following:
   - In the main window, select the recent error message.
   - On the Console **Menu** bar, click **Edit** and then click **Select All**.
6. Click **Edit** and then click **Copy**.
7. Open TextEdit and then paste the logs you copied into a new file.
8. Attach the file in your email to support.

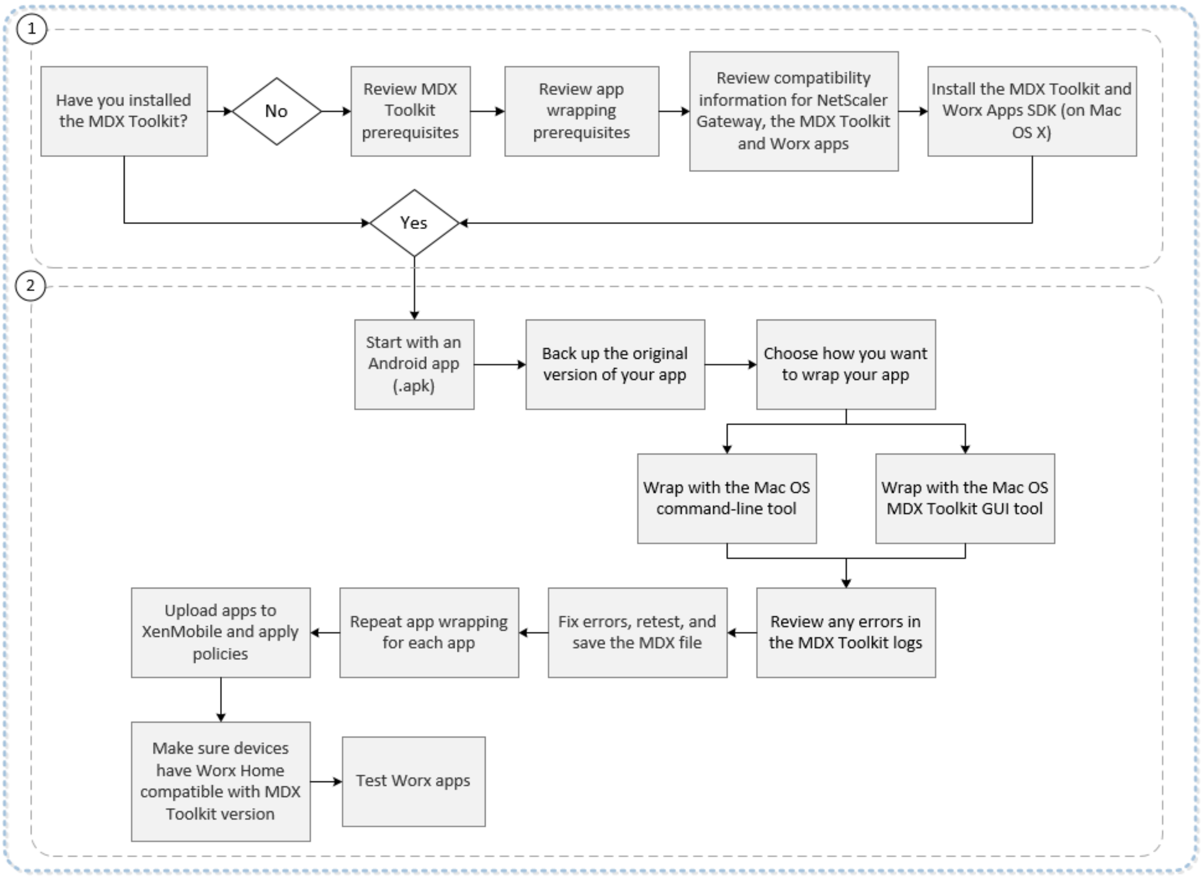
Wrapping Android mobile apps

June 2, 2020

This article describes how Citrix Endpoint Management administrators wrap third-party enterprise apps and how developers wrap ISV apps. To wrap Android mobile apps, use the MDX Toolkit, which includes a macOS graphical interface tool and a Java command-line tool. The command-line tool has customization options, can be referenced from scripts that automate the app wrapping process, and lets you preset some MDX policies.

The file type for a wrapped app is .mdx. You upload the .mdx file to the Endpoint Management console where you then configure specific app details and policy settings thatthe Endpoint Management Store enforces. When users sign on, the app appears in the app store. Users can then subscribe, download, and install the app on their device.

The following figure provides an overview of the app wrapping steps, from installation of the MDX Toolkit through testing mobile productivity apps. Related topics are listed under the diagram.


1
Have you installed the MDX Toolkit?
No
Review MDX Toolkit prerequisites
Review app wrapping prerequisites
Review compatibility information for NetScaler Gateway, the MDX Toolkit and Worx apps
Install the MDX Toolkit and Worx Apps SDK (on Mac OS X)
Yes
2
Start with an Android app (.apk)
Back up the original version of your app
Choose how you want to wrap your app
Wrap with the Mac OS command-line tool
Wrap with the Mac OS MDX Toolkit GUI tool
Review any errors in the MDX Toolkit logs
Fix errors, retest, and save the MDX file
Repeat app wrapping for each app
Upload apps to XenMobile and apply policies
Make sure devices have Worx Home compatible with MDX Toolkit version
Test Worx apps

For details on number one, see:

- System requirements
- Other requirements for wrapping Android mobile apps
- Endpoint Management compatibility
- Installing the MDX Toolkit

For details on number two, see:

- ISV Android app wrapping with the command line
- Enterprise Android app wrapping with the command line
- Command Options

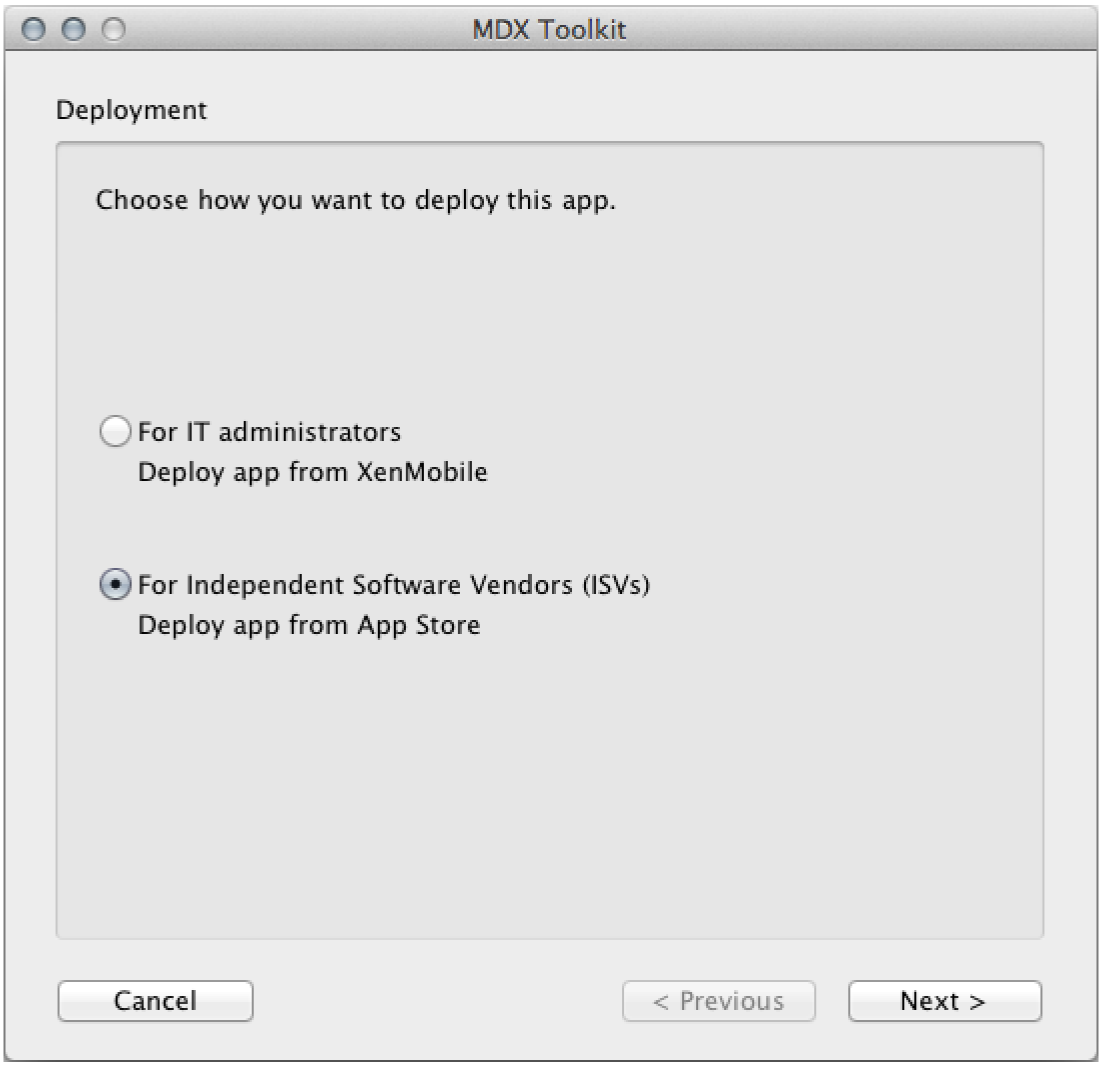
- Presetting MDX policies for Android apps
- Identifying Android app wrapping errors
- Collecting app logs from the command line
- Add an MDX app

> **Important:**
>
> Make sure that your user devices are updated with a version of Secure Hub that is compatible with the version of MDX Toolkit used to wrap apps. Otherwise, users receive an error message about the incompatibility. For details, see Endpoint Management compatibility.

ISV app wrapping with the graphical interface

The following steps describe the general process for wrapping an ISV app that you will deploy from the Google Play Store.

1. Before you use the toolkit to wrap apps, be sure to back up the original version of those apps so you can return to them if needed.
2. Start the MDX Toolkit from your iOS Applications folder, select **For Independent Software Vendors (ISVs),** and then click **Next**.


Deployment

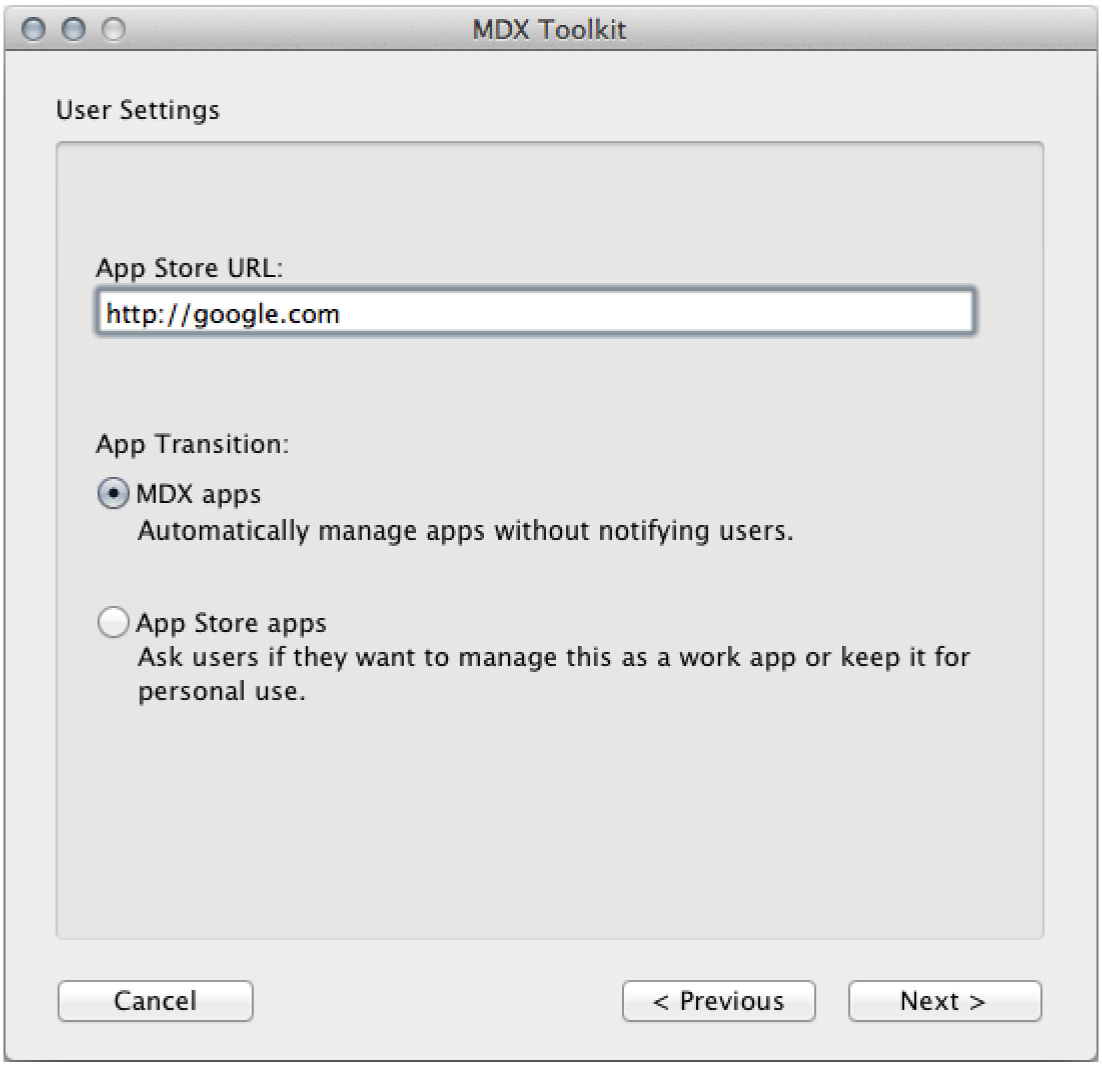
Choose how you want to deploy this app.

( ) For IT administrators
Deploy app from XenMobile

[✓ For Independent Software Vendors (ISVs)
Deploy app from App Store]

3. In the **Deploy from App Store** screen, select your app and click **Next**.
4. In the **User Settings** screen, if you already have the app store URL, enter it. If you don't have the URL, enter a placeholder such as `https://play.google.com/store/apps/details?id=com.citrix`. You can update the URL later.

   For Premium apps, select **MDX apps**. For General apps, select **App Store apps**.



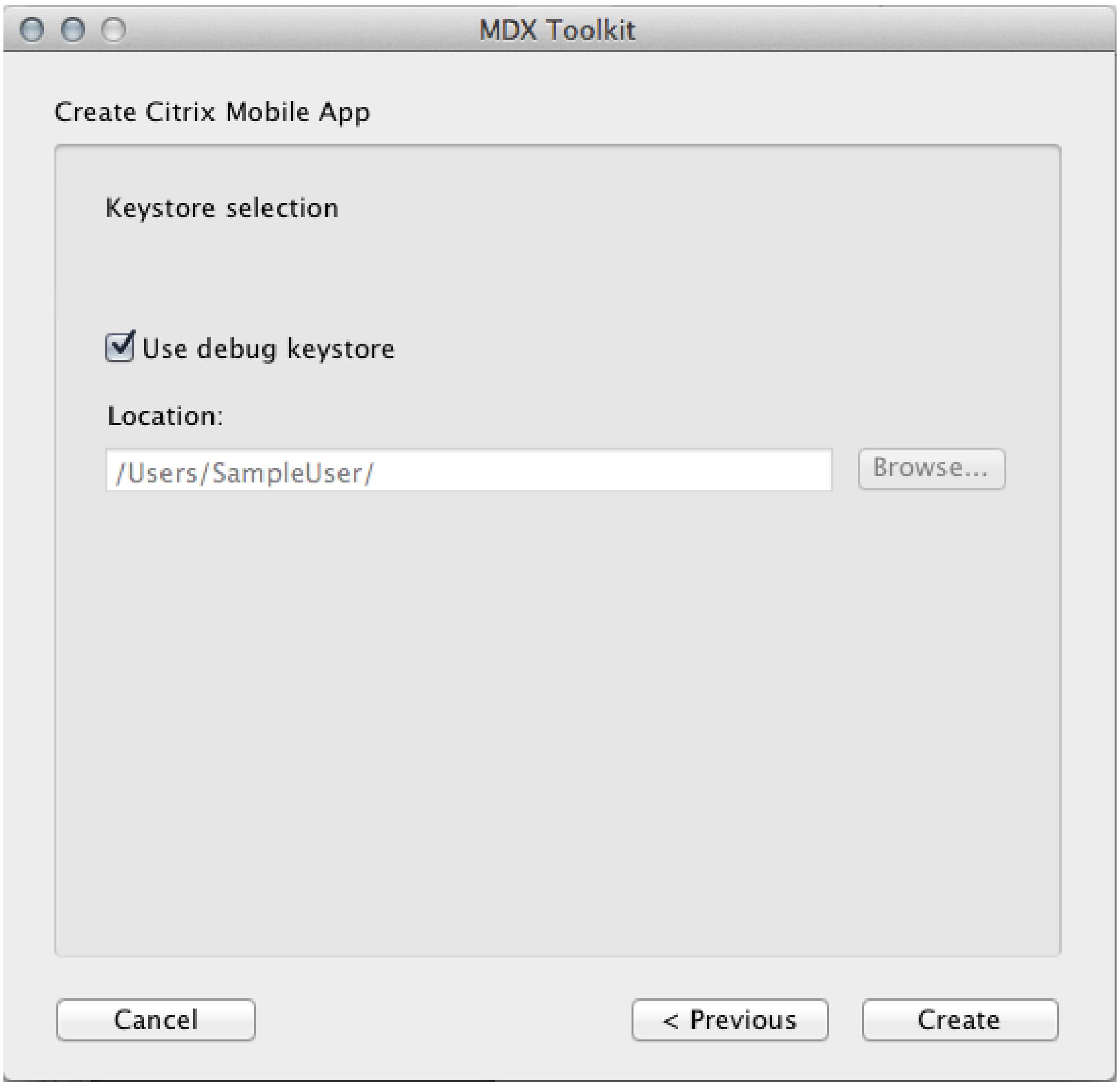
App Store URL: http://google.com

App Transition:

(● MDX apps) Automatically manage apps without notifying users.

App Store apps — Ask users if they want to manage this as a work app or keep it for personal use.

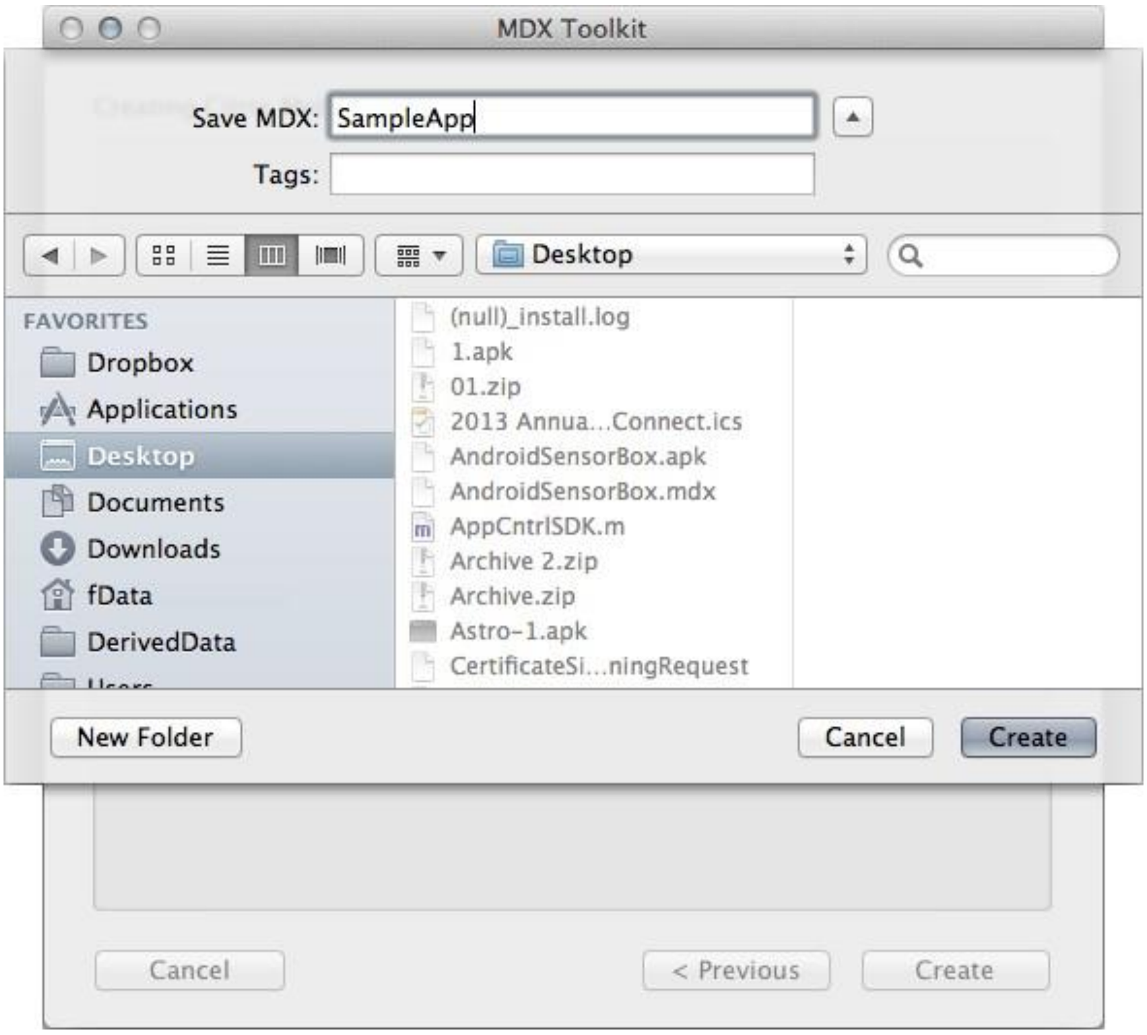
5. In the **Verify App Details** screen, update the details as needed.
6. Browse to your keystore and click **Create**.



[✓] Use debug keystore

Location: /Users/SampleUser/

7. Save your app.



When the GUI tool finishes wrapping an app, the app file name includes _andr.

Enterprise Android app wrapping with the command line

You can use enterprise app wrapping to wrap custom (in-house) apps and some third-party apps. You should acquire third-party apps directly from the app vendor. For enterprise app wrapping, begin with an Android application (.apk). Before you use the toolkit to wrap apps, back up the original version of those apps so you can return to them if needed.

The following example shows a basic app wrapping command using default settings. The app is signed with the provided keystore. A keystore is a file that contains certificates used to sign your Android app. If the keystore contains multiple private keys, you can specify the key alias. You create a keystore one time. Then, you can use the keystore to sign the apps that you wrap. If you do not use the same keystore to wrap the new version of an app you previously deployed, upgrades of that app will not work. Users will need to manually remove the old version before they can install the new one.

Modify the bold information for your specific system. The trailing backslash signifies that the command continues to the next line. Please remove these symbols before running the command.

> **Note:**
>
> Because the /Applications/ directory is restricted, you may need to run the following command in super user mode. To do this, add sudo in front of the command. You will be prompted for your computer password when running from this restricted directory.

```
java -jar /Applications/Citrix/MDXToolkit/ManagedAppUtility.jar \
wrap \
-in ~/Desktop/SampleApps/Sample.apk \
-out ~/Desktop/SampleApps/Sample.mdx \
-keystore ~/Desktop/MyCompany.keystore \
-storepass MyKeystorePassword \
-keyalias MyCompanyKeyAlias \
-keypass MyKeyAliasPassword
<!--NeedCopy-->
```

The following are examples of options you may add to the preceding command, after modifying the information in bold:

- -appName "Wrapped Sample app"
- -appDesc "This is my newly wrapped Android application."

In addition, if the release keystore is not available during development, use the following command to create a retail build of a mobile app that is signed with your key:

```
java -jar /Applications/Citrix/MDXToolkit/ManagedAppUtility.jar \
wrap \
-in ~/Desktop/SampleApps/Sample.apk \
-out ~/Desktop/SampleApps/Sample.mdx \
-keystore ~/Desktop/MyCompany.keystore \
-storepass MyKeystorePassword \
-keyalias MyCompanyKeyAlias \
-keypass MyKeyAliasPassword \
-createCert
<!--NeedCopy-->
```

For details about the options, see Command Options. For inline documentation, use the -help option.

ISV Android app wrapping with the command line

Before you use the toolkit to wrap apps, be sure to back up the original version of those apps so you can return to them if needed. To generate wrapped ISV applications for Android, start with the following basic wrapping command.

```
java -jar /Applications/Citrix/MDXToolkit/ManagedAppUtility.jar \
wrap \
-in ~/Desktop/SampleApps/Sample.apk \
-out ~/Desktop/SampleApps/Sample.mdx \
-keystore ~/Desktop/MyCompany.keystore \
-storepass MyKeystorePassword \
-keyalias MyCompanyKeyAlias \
-keypass MyKeyAliasPassword \
-createCert
<!--NeedCopy-->
```

To wrap an app as an ISV app, you must set the *-apptype* parameter as follows:

- **Premium:** To wrap an app as a Premium app, in which some Citrix policies are enforced even for unmanaged users, add the following option: -apptype Premium
- **General:** To wrap an app as a General app, which contains no Citrix policy enforcement for an unmanaged user, add the following option: -apptype General

If you need to upload the wrapped .apk file to the Google Play Store or web server and the URL is known when wrapping, add the -storeURL option. Make sure to also set the apptype parameter.

`-storeURL "https://play.google.com/store/apps/details?id=com.zenprise"`

If you do not know the URL at the time of wrapping, you can modify the .mdx file later with the following command:

```
java -jar /Applications/Citrix/MDXToolkit/ManagedAppUtility.jar \
setinfo \
-in ~/Desktop/SampleApps/Sample.mdx \
-out ~/Desktop/SampleApps/wrapped/Sample.mdx \
-storeURL \
“https://play.google.com/store/apps/details?id=com.zenprise”
<!--NeedCopy-->
```

If you customized the policy file, be sure to point to your modified file:

`-policyxml /Applications/Citrix/MDXToolkit/data/policy_metadata.xml` For details about the options, see Command Options. For inline documentation, use the **-help** option.

Command Options

wrap command

- **Help:** Displays Help for this command.
- **In:** Required. Path and file name of the app you are wrapping.

- **Out:** Optional. Path and file name for the resulting .mdx file. If this option is omitted, the file has the same path and file name as the input file and has an .mdx extension.
- **AppType:** Optional. Defaults to MDXOnly. To generate ISV apps, use either General or Premium.
- **KeyStore:** Path to the keystore file. Required if signing the .apk file.
- **StorePass:** Password for the keystore. Required if signing the .apk file.
- **KeyAlias:** Name of the specific key in the keystore. Required if signing the .apk file.
- **KeyPass:** Password for the specific key. Required if signing the .apk file.
- **SigAlg:** Optional. Algorithm to use when signing.
- **AppName:** Optional. Application name, obtained from the app if possible.
- **AppDesc:** Optional. Application description, obtained from the app if possible.
- **MinPlatform:** Optional. Minimum supported SDK level. Defaults to blank.
- **MaxPlatform:** Optional. Maximum supported SDK level. Defaults to blank.
- **ExcludedDevices:** Optional. List of device types on which the app is not allowed to run. Defaults to blank.
- **PolicyXML:** Optional. Replacement XML policy definition file and path. Defaults to the built-in policy definitions. Example:

  ```
  -policyxml/Applications/Citrix/MDXToolkit/data/policy_metadata.xml
  ```

  For details, see Presetting MDX Policies for Android Apps, next.
- **StoreURL:** For ISV apps, the URL of the app in the Google App Store. Defaults to blank.

sign command

- **Help:** Displays Help for this command.
- **In:** Required. Path and file name of the app you are wrapping.
- **Out:** Optional. Path and file name for the resulting .mdx file. If this option is omitted, the file has the same path and file name as the input file and has an .mdx extension.
- **KeyStore:** Required. Path to the keystore file.
- **StorePass:** Required. Password for the keystore.
- **KeyAlias:** Required. Name of the specific key in the keystore.
- **KeyPass:** Required. Password for the specific key.
- **SigAlg:** Optional. Algorithm to use when signing.

setinfo command

- **Help:** Displays Help for this command.

- **In:** Required. Path and file name of the app to be modified.
- **Out:** For setinfo, the output path or file name must differ from the original.
- **AppType:** Optional. Defaults to MDXOnly. To generate ISV apps, use either General or Premium.
- **KeyStore:** Path to the keystore file. Required if signing the .apk file.
- **StorePass:** Password for the keystore. Required if signing the .apk file.
- **KeyAlias:** Name of the specific key in the keystore. Required if signing the .apk file.
- **KeyPass:** Password for the specific key. Required if signing the .apk file.
- **SigAlg:** Optional. Algorithm to use when signing.
- **AppName:** Optional. Application name, obtained from the app if possible.
- **AppDes:** Optional. Application description, obtained from the app if possible.
- **MinPlatform:** Optional. Minimum supported SDK level. Defaults to blank.
- **MaxPlatform:** Optional. Maximum supported SDK level. Defaults to blank
- **ExcludedDevices:** Optional. List of device types on which the app is not allowed to run. Defaults to blank.
- **StoreURL:** For ISV apps, the URL of the app in the Google App Store. Defaults to blank.
- **PolicyXML:** Optional. Replacement XML policy definition file and path. Defaults to the built-in policy definitions. Example:

  ```
  -policyxml /Applications/Citrix/MDXToolkit/data/policy_metadata.xml
  ```

  For details, see Presetting MDX Policies for Android Apps, next.

Presetting MDX policies for Android apps

For apps that you wrap with the MDX Toolkit command-line tool, you can preset some MDX policies. You can also configure policies in the Citrix Endpoint Management console when you add the apps.

1. Update policy values in the policy XML file.

   The MDX Toolkit installer creates this policy file: Applications/Citrix/MDXToolkit/data/policy_metadata.xml

   > **Note**
   >
   > Be aware that the policies files for Android and iOS differ. To preset policies for both of those platforms, you must update their respective policy XML files.

2. When you wrap the app with the command line, include

   ```
   -policyxml /Applications/Citrix/MDXToolkit/data/policy_metadata.xml
   ```

Identifying Android app wrapping errors

If you encounter an error when wrapping an Android app, you can use the MDX Toolkit logs to identify the error. You must have administrator rights to view the MDX Toolkit logs.

When you run the MDX Toolkit, the tool saves a log file to the following location: Applications/CitrixMDXToolkit/Logs/Citrix.log. By default, the tool saves warnings and errors in the log.

Collecting app logs from the command line

1. Install the Android Debug Bridge from the Android Developer web site. For details, see Android Debug Bridge.
2. Enter the following command to clear existing logs: **"adb logcat -c"**
3. Reproduce the issue.
4. Enter the following command to capture the logs in a file: **adb logcat -d > Name_of_Log_File.txt**

MDX third-party app policies at a glance

May 29, 2020

This article describes the third-party MDX app policies for iOS and Android. The MDX Toolkit does not support Windows. The notes include restrictions and Citrix recommendations. To see which policies the Android for Work container supports, see the related section in Android for Work.

For policies for the Citrix mobile productivity apps, see MDX policies for mobile productivity apps at a glance.

> Note:
>
> Secure Hub refreshes policies during certain actions. For details, see Administering Secure Hub.

Authentication

Device passcode

- iOS: Yes
- Android: No
- Default setting: Off

App passcode

- iOS: Yes

- Android: Yes
- Default setting: On

Online session required

- iOS: Yes
- Android: No
- Default setting: Off

Maximum offline period

- iOS: Yes
- Android: Yes
- Default setting: 168 hours (7 days)

Alternate NetScaler Gateway

- iOS: Yes
- Android: Yes
- Default setting: Empty

Device security

Block jailbroken or rooted

- iOS: Yes
- Android: Yes
- Default setting: On

Require device lock

- iOS: No
- Android: Yes
- Default setting: Off

Network requirements

Require Wi-Fi

- iOS: Yes
- Android: Yes

- Default setting: Off

Allowed Wi-Fi Networks

- iOS: Yes
- Android: Yes
- Default setting: Empty

Miscellaneous access

App update grace period (hours)

- iOS: Yes
- Android: Yes
- Default setting: 168 hours (7 days)

  > Note:
  >
  > Citrix recommends using a value other than zero (0). A zero value immediately prevents users, without warning, from using a running app until they download and install the update. This setting may lead to a situation in which users are forced to exit the app and potentially lose work.

Erase app data on lock

- iOS: Yes
- Android: Yes
- Default setting: Off

Active poll period (minutes)

- iOS: Yes
- Android: Yes
- Default setting: 60 minutes (1 hour)

  > Note:
  >
  > Only set this value lower than the default for high-risk apps, or performance may be affected.

Encryption

Encryption type

- iOS: Yes
- Android: Yes
- Default setting: MDX encryption

> Caution:
>
> For newly added apps, when you change from **Platform encryption with compliance enforcement** to **MDX encryption**, you are forced to remove and reinstall the app. The default setting for newly added apps is **Platform encryption with compliance enforcement**.

Non-compliant device behavior

- iOS: Yes
- Android: Yes
- Default setting: Allow app after warning

Enable MDX encryption

- iOS: Yes
- Android: No
- Default setting: On

> Caution:
>
> If you change this policy after deploying an app, users must reinstall the app.

Encryption keys

- iOS: No
- Android: Yes
- Default setting: Offline access permitted is the only available option.

MDX Private file encryption

- iOS: No
- Android: Yes
- Default setting: SecurityGroup

Private file encryption exclusions

- iOS: No
- Android: Yes
- Default setting: Empty

Access limits for public files

- iOS: No
- Android: Yes
- Default setting: Empty

Database encryption exclusions

- iOS: Yes
- Android: No
- Default setting: Empty

MDX Public file encryption

- iOS: No
- Android: Yes
- Default setting: SecurityGroup

Public file encryption exclusions

- iOS: No
- Android: Yes
- Default setting: Empty

Public file migration

- iOS: No
- Android: Yes
- Default setting: Write(WO/RW)

File encryption exclusions

- iOS: Yes
- Android: No
- Default setting: Empty

Security group

- iOS: Yes
- Android: Yes
- Default setting: Empty

> Caution:
>
> To apply this policy to an existing app, users must delete and reinstall the app.

App interaction

Cut and copy

- iOS: Yes
- Android: Yes
- Default setting: Restricted

Paste

- iOS: Yes
- Android: Yes
- Default setting: Unrestricted

Document exchange (Open in)

- iOS: Yes
- Android: Yes
- Default setting: Restricted

Restricted Open In exception list

- iOS: Yes
- Android: Yes
- Default setting: Empty (for Android); Office 365 apps (for iOS)

> Caution:
>
> Consider the security implications of this policy. The exception list allows content to travel between unmanaged apps and the secure MDX environment.

Inbound document exchange (Open In)

- iOS: Yes
- Android: Yes
- Default setting: Unrestricted (for Android and iOS); All (for Android for Work)

App URL schemes

- iOS: Yes
- Android: No
- Default setting: All registered app URL schemes are blocked.

URL domains excluded from filtering

- iOS: Yes
- Android: Yes
- Default setting: Empty

Allowed URLs

- iOS: Yes
- Android: No
- Default setting:
  +maps.apple.com
  +itunes.apple.com
  ^http:=ctxmobilebrowser:
  ^https:=ctxmobilebrowsers:
  ^mailto:=ctxmail:
  +^citrixreceiver:
  +^telprompt:
  +^tel:
  +^col-g2m-2:
  +^col-g2w-2:
  +^maps:ios_addr
  +^mapitem:

Allowed Secure Web domains

- iOS: Yes
- Android: Yes
- Default setting: Empty

App restrictions

Block camera

- iOS: Yes
- Android: Yes
- Default setting: On

Block gallery

- iOS: No
- Android: Yes
- Default setting: Off

Block Photo Library

- iOS: Yes
- Android: No
- Default setting: On

Block mic record

- iOS: Yes
- Android: Yes
- Default setting: On

Block dictation

- iOS: Yes
- Android: No
- Default setting: On

Block location services

- iOS: Yes
- Android: Yes
- Default setting: On

Block SMS compose

- iOS: Yes

- Android: Yes
- Default setting: On

Block screen capture

- iOS: No
- Android: Yes
- Default setting: On

Block device sensor

- iOS: No
- Android: Yes
- Default setting: On

Block NFC

- iOS: No
- Android: Yes
- Default setting: On

Block iCloud

- iOS: Yes
- Android: No
- Default setting: On

Block Look Up

- iOS: Yes
- Android: No
- Default setting: On

Block file backup

- iOS: Yes
- Android: No
- Default setting: On

Block AirPrint

- iOS: Yes
- Android: No
- Default setting: On

Block printing

- iOS: No
- Android: Yes
- Default setting: On

Block AirDrop

- iOS: Yes
- Android: No
- Default setting: On

Block Facebook and Twitter APIs

- iOS: Yes
- Android: No
- Default setting: On

Obscure screen contents

- iOS: Yes
- Android: No
- Default setting: On

Block third-party keyboards

- iOS: Yes
- Android: No
- Default setting: On

Block app logs

- iOS: Yes
- Android: Yes
- Default setting: Off

App network access

Network access

- iOS: Yes
- Android: Yes
- Default setting: For newly uploaded apps, the default is **Blocked** for all apps, except Secure Mail. Because Intune does not have a blocked state, the default for Secure Mail is **Unrestricted**.

micro VPN session required

- iOS: Yes
- Android: Yes
- Default setting: No

micro VPN session required grace period (minutes)

- iOS: Yes
- Android: Yes
- Default setting: 0 (no grace period)

Certificate label

- iOS: Yes
- Android: Yes
- Default setting: Empty

Exclusion List

- iOS: Yes
- Android: Yes
- Default setting: Empty

Block localhost connections

- iOS: No
- Android: Yes
- Default setting: Off

App logs

Default log output

- iOS: Yes
- Android: Yes
- Default setting: File

Default log level

- iOS: Yes
- Android: Yes
- Default setting: 4 (informational messages)

Max log files

- iOS: Yes
- Android: Yes
- Default setting: 2

Max log file size

- iOS: Yes
- Android: Yes
- Default setting: 2 MB

Redirect system logs

- iOS: Yes
- Android: No
- Default setting: On

App geofence

Center point longitude

- iOS: Yes
- Android: Yes
- Default setting: 0

Center point latitude

- iOS: Yes
- Android: Yes
- Default setting: 0

Radius

- iOS: Yes
- Android: Yes
- Default setting: 0 (disabled)

> Note:
>
> Set the radius in meters. When set to zero, the geofence is disabled.

Analytics

Google Analytics level of detail

- iOS: Yes
- Android: Yes
- Default setting: Complete

Reporting

Citrix reporting

- iOS: Yes
- Android: No
- Default setting: Off

> Note:
>
> Citrix might also control this feature with a feature flag. Both the feature flag and this policy must be enabled for this feature to function.

Upload token

- iOS: Yes
- Android: No
- Default setting: Empty

Send reports over WiFi only

- iOS: Yes
- Android: No
- Default setting: On

Reporting file cache maximum

- iOS: Yes
- Android: No
- Default setting: 2 MB

MDX policies for third-party apps for Android

December 15, 2020

This article describes the MDX policies for Android third-party apps. You can change policy settings in the Citrix Endpoint Management console.

Authentication

App passcode

If **On**, a PIN or passcode is required to unlock the app when it starts or resumes after a period of inactivity. Default value is **On**.

To configure the inactivity timer for all apps, set the INACTIVITY_TIMER value in minutes in Client Properties on the **Settings** tab. The default inactivity timer value is 60 minutes. To disable the inactivity timer, so that a PIN or passcode prompt appears only when the app starts, set the value to zero.

> **Note:**
>
> If you select Secure offline for the Encryption keys policy, this policy is automatically enabled.

Maximum offline period (hours)

Defines the maximum period an app can run offline without a network logon for reconfirming entitlement and refreshing policies. Default value is **168 hours** (7 days). Minimum period is 1 hour.

The user is reminded to log on at 30, 15, and 5 minutes before the period expires. After expiration, the app remains locked until the user completes a successful network logon.

Alternate Citrix Gateway

> Note:
>
> This policy name in the Endpoint Management console is **Alternate NetScaler Gateway**.

Address of a specific alternate Citrix Gateway (formerly, NetScaler Gateway) that is used for authentication and for micro VPN sessions with this app. This is an optional policy when used with the Online session required policy forces apps to reauthenticate to the specific gateway. Such gateways would typically have different (higher assurance) authentication requirements and traffic management policies. If left empty, the server's default is always used. Default value is empty.

Device Security

Block jailbroken or rooted

If **On**, the app is locked when the device is jailbroken or rooted. If **Off**, the app can run even if the device is jailbroken or rooted. Default value is **On**.

Require device lock

If **Device PIN or passcode**, the app is locked if the device does not have a PIN or passcode. If **Device pattern screen lock**, the app is locked if the device does not have a pattern screen lock set. If **Off**, the app is allowed to run even if the device does not have a PIN, passcode, or pattern screen lock set. Default value is **Off**.

Device PIN or passcode requires a minimum version of Android 4.1 (Jelly Bean). Setting the policy to **Device PIN or passcode** prevents an app from running on older versions.

On Android M devices, the **Device PIN or passcode** and **Device pattern screen lock** options have the same effect: With either of those options, the app is locked if the device does nothave a PIN, passcode, or pattern screen lock set.

Network Requirements

Require Wi-Fi

If **On**, the app is locked when the device is not connected to a Wi-Fi network. If **Off**, the app can run if the device has an active connection, such as a 4G/3G, LAN, or Wi-Fi connection. Default value is **Off**.

Allowed Wi-Fi Networks

Comma-delimited list of allowed Wi-Fi networks. If the network name contains any non-alphanumeric characters (including commas), the name must be enclosed in double-quotes. App runs only if connected to one of the networks listed. If left blank, all networks are allowed. This does not affect connections to cellular networks. Default value is blank.

Miscellaneous Access

App update grace period (hours)

Defines the grace period in which an app can be used after the system discovers that an app update is available. Default value is **168 hours** (7 days).

> Note:
>
> Using a value of zero is not recommended since it immediately prevents a running app from being used until the update is downloaded and installed (without any warning to the user). This could lead to a situation where the user running the app is forced to exit the app (potentially losing work) to comply with the required update.

Erase app data on lock

Erases data and resets the app when the app is locked. If **Off**, app data is not erased when the app is locked. Default value is **Off**.

An app can be locked for any of the following reasons:

- Loss of app entitlement for the user
- App subscription removed
- Account removed
- Secure Hub uninstalled
- Too many app authentication failures
- Jailbroken device detected (per policy setting)
- Device placed in locked state by other administrative action

Active poll period (minutes)

When an app starts, the MDX framework polls Citrix Endpoint Management to determine current app and device status. Assuming the server running Endpoint Management can be reached, the framework returns information about the lock/erase status of the device and the enable/disable status of the app. Whether the server can be reached or not, a subsequent poll is scheduled based on the active poll period interval. After the period expires, a new poll is again attempted. Default value is **60 minutes** (1 hour).

> Important:
>
> Only set this value lower for high-risk apps or performance may be affected.

Non-compliant device behavior

Allows you to choose an action when a device does not adhere to the minimum compliance requirements. Select **Allow app** for the app to run normally. Select **Allow app after warning** for the app to run after the warning appears. Select **Block** to block the app from running. Default value is **Allow app after warning**.

Public file migration

This policy is enforced only when you enable the Public file encryption policy (changed from **Disabled** to **SecurityGroup** or **Application**). This policy is applicable only to existing, unencrypted public files and specifies when these files are encrypted. Default value is **Write (RO/RW)**.

Options:

- **Disabled**. Does not encrypt existing files.
- **Write (RO/RW)**. Encrypts the existing files only when they are opened for write-only or read-write access.
- **Any**. Encrypts the existing files when they are opened in any mode.

> Note:
>
> - New files or existing unencrypted files that are overwritten encrypt the replacement files in every case.
> - Encryption an existing public file makes the file unavailable to other apps that we do not have the same encryption key.

Security Group

Leave this field blank if you want all mobile apps managed by Citrix Endpoint Management to exchange information with one another. Define a security group name to manage security settings for specific sets of apps (for example, Finance or Human Resources).

> Caution:
>
> If you change this policy for an existing app, users must delete and reinstall the app to apply the policy change.

Allowed Secure Web domains

This policy is only in effect for the domains not excluded by URL filtering policy. Add a comma-separated list of fully qualified domain names (FQDN) or DNS suffixes that are redirected to the Secure Web app when Document Exchange is Restricted.

If this policy contains any entries, only those URLs with host fields matching at least one item in the list (via DNS suffix match) will be redirected to the Secure Web app when Document Exchange is Restricted.

All other URLs are sent to the default Android web browser (bypassing the Document Exchange Restricted policy). Default value is empty.

App Interaction

Cut and Copy

Blocks, permits, or restricts clipboard cut and copy operations for this app. If **Restricted**, the copied Clipboard data is placed in a private Clipboard that is only available to MDX apps. Default value is **Restricted**.

Paste

Blocks, permits, or restricts clipboard paste operations for the app. If **Restricted**, the pasted clipboard data is sourced from a private Clipboard that is only available to MDX apps. Default value is **Unrestricted**.

Document exchange (Open In)

Blocks, permits, or restricts document exchange operations for the app. If **Restricted**, documents can be exchanged only with other MDX apps and the app exceptions specified in the Restricted Open-In exception list policy. If **Unrestricted**, set the Private file encryption and Public file encryption policies to **Disabled** so that users can open documents in unwrapped apps. Default value is **Restricted**.

Restricted Open-In exception list

When the Document exchange (Open In) policy is **Restricted**, this list of Android intents is allowed to pass to unmanaged apps. A familiarity with Android intents is needed to add filters to the list. A filter can specify action, package, scheme, or any combination.

Examples

```
{
 action=android.intent.action.MAIN }

{
 package=com.sharefile.mobile }

{
 action=android.intent.action.DIAL scheme=tel }

<!--NeedCopy-->
```

> Caution
>
> Be sure to consider the security implications of this policy. The exception list allows content to travel between unmanaged apps and the MDX environment.

Inbound document exchange (Open In)

Blocks, restricts, or allows inbound document exchange operations for this app. If **Restricted**, documents can be exchanged only with other MDX apps. Default value is **Unrestricted**.

If **Blocked** or **Restricted**, you can use the Inbound document exchange whitelistpolicy to specify apps that can send documents to this app. For information about other policy interactions, see the Block Gallery policy.

Options: **Unrestricted**, **Blocked**, or **Restricted**

App Restrictions

> Important:
>
> Be sure to consider the security implications of policies that block apps from accessing or using phone features. When those policies are **Off**, content can travel between unmanaged apps and the Secure environment.

Block camera

If **On**, prevents an app from directly using the camera hardware. Default value is **On**.

Block Gallery

If **On**, prevents an app from accessing the Gallery on the device. Default value is **Off**. This policy works in conjunction with the policy Inbound document exchange (Open In).

- If Inbound document exchange (Open In) is set to **Restricted**, users working in the managed app cannot attach images from the Gallery, regardless of the Block Gallery setting.
- If Inbound document exchange (Open In) is set to **Unrestricted**, users working in the managed app experience the following:
  - Users can attach images if Block Gallery is set to **Off**.
  - Users are blocked from attaching images if Block Gallery is **On**.

Block mic record

If **On**, prevents an app from directly using the microphone hardware. Default value is **On**.

Block location services

If **On**, prevents an app from using the location services components (GPS or network). Default value is **Off** for Secure Mail.

Block SMS compose

If **On**, prevents an app from using the SMS compose feature used to send SMS/text messages from the app. Default value is **On**.

Block screen capture

If **On**, prevents users from taking screen captures while the app is running. Also, when the user switches apps, obscures the app screen. Default value is **On**.

When using the Android Near Field Communication (NFC) feature, some apps take a screenshot of itself before beaming the content. To enable that feature in a wrapped app, change the Block screen capture policy to **Off**.

Block device sensor

If **On**, prevents an app from using the device sensors (such as accelerometer, motion sensor, and gyroscope). Default value is **On**.

Block NFC

If **On**, prevents an app from using the Near Field Communications (NFC). Default value is **On**.

Block app logs

If **On**, prohibits an app from using the mobile productivity app diagnostic logging facility. If **Off**, app logs are recorded and may be collected by using the Secure Hub email support feature. Default value is **Off**.

Block printing

If **On**, prevents an app from printing data. If an app has a Share command, you must set Document Exchange (Open in) to **Restricted** or **Blocked** to block printing fully. Default value is **ON**.

App Network Access

Network access

> Note:
>
> **Tunneled - Web SSO** is the name for the **Secure Browse** in the settings. The behavior is the same.

The settings options are as follows:

- **Use Previous Settings**: Defaults to the values you had set in the earlier policies. If you change this option, you shouldn't revert to this option. Also note that changes to the new policies do not take effect until the user upgrades the app to version 18.12.0 or later.
- **Blocked**: Networking APIs used by your app will fail. Per the previous guideline, you should gracefully handle such a failure.
- **Unrestricted**: All network calls go directly and are not tunneled.
- **Tunneled - Full VPN**: All traffic from the managed app tunnels through Citrix Gateway.
- **Tunneled - Web SSO**: The HTTP/HTTPS URL is rewritten. This option allows only the tunneling of HTTP and HTTPS traffic. A significant advantage of **Tunneled - Web SSO** is single sign-on (SSO) for HTTP and HTTPS traffic and also PKINIT authentication. On Android, this option has low setup overhead and is thus the preferred option for web browsing types of operations.
- **Tunneled - Full VPN and Web SSO**: Permits switching between VPN modes automatically as needed. If a network request fails due to an authentication request which cannot be handled in a specific VPN mode, it is retried in an alternate mode.

If one of the **Tunneled modes** is selected, a per-app VPN tunnel in this initial mode is created back to the enterprise network, and Citrix Gateway split tunnel settings are used. Citrix recommends **Tunneled Full VPN** for connections that employ client certificates or end-to-end SSL to a resource in the enterprise network. Citrix recommends **Tunneled - Web SSO** for connections that require single sign-on (SSO).

micro VPN session required

If **Yes**, the user must have a connection to the enterprise network and an active session. If **No**, an active session is not required. Default value is **Use Previous Setting**. For newly uploaded apps, the default value is **No**. Whichever setting was selected prior to the upgrade to this policy remains in effect until an option other than **Use Previous Setting** is selected.

micro VPN session required grace period (minutes)

Defines the grace period that an app can continue to be used after the system has discovered that an app update is available. Default value is 168 hours (7 days).

> Note:
>
> Using a value of zero is not recommended since it immediately prevents a running app from being used until the update is downloaded and installed (without any warning to the user). This could lead to a situation where the user running the app is forced to exit the app (potentially losing work) in order to comply with the required update.

Certificate label

When used with the StoreFront certificate integration service, this label identifies the specific certificate required for this app. If no label is provided, a certificate is not made available for use with a public key infrastructure (PKI). Default value is empty (no certificate used).

Exclusion List

Comma-delimited list of FQDNs or DNS suffixes to be accessed directly instead of through a VPN connection. This only applies to the **Tunneled - Web SSO** mode when Citrix Gateway is configured with Split tunnel reverse mode.

Block localhost connections

If **On**, apps are not permitted to make localhost connections. Localhost is an address (such as 127.0.0.1 or ::1) for communications occurring locally on the device. The localhost bypasses the local network interface hardware and accesses network services running on the host. If **Off**, this policy overrides the Network Access policy, meaning that apps can connect outside the secure container if the device is running a proxy server locally. Default is **Off**.

App Logs

Default log output

Determines which output mediums are used by Citrix Endpoint Management app diagnostic logging facilities by default. Possibilities are file, console, or both. Default value is file.

Default log level

Controls default verbosity of the mobile productivity app diagnostic logging facility. Higher-level numbers include more detailed logging.

- 0 - Nothing logged
- 1 - Critical errors
- 2 - Errors
- 3 - Warnings
- 4 - Informational messages
- 5 - Detailed informational messages
- 6 through 15 - Debug levels 1 through 10

Default value is level **4** (Informational messages).

Max log files

Limits the number of log files retained by the mobile productivity app diagnostic logging facility before rolling over. Minimum is **2**. Maximum is **8**. Default value is **2**.

Max log file size

Limits the size in megabytes (MB) of the log files retained by the mobile productivity app diagnostic logging facility before rolling over. Minimum is **1** MB. Maximum is **5** MB. Default value is **2** MB.

App Geofence

Center point longitude

Longitude (X coordinate) of the center point of point/radius geofence in which the app is constrained to operate. When operated outside the configured geofence, the app remains locked. Should be expressed in signed degrees format (DDD.dddd), for example "-31.9635". West longitudes should be prefaced with a minus sign. Default value is **0**.

Center point latitude

Latitude (Y coordinate) of the center point of point/radius geofence in which the app is constrained to operate. When operated outside the configured geofence, the app remains locked.

Should be expressed in signed degrees format (DDD.dddd), for example "43.06581". Southern latitudes should be prefaced with a minus sign. Default value is **0**.

Radius

Radius of the geofence in which the app is constrained to operate. When operated outside the configured geofence, the app remains locked.
Should be expressed in meters. When set to zero, the geofence is disabled. Default is 0 (disabled).

Analytics

Google Analytics of detail

Citrix collects analytics data to improve product quality. Selecting Anonymous opts you out of including company identifiable information.

Supported device policies and MDX policies for Android Enterprise

March 9, 2021

MDX Toolkit supports the device policies as supported by Android Enterprise. For the complete list of Android Enterprise policies supported by MDX, see Configure Android Enterprise device and app policies.

For more information about device policies, see Device Policies.

Authentication policies	Supported
App passcode	X
Maximum offline period (hours)	X

Miscellaneous Access policies	Supported
App update grace period (hours)	X
Disable required upgrade	X

Miscellaneous Access policies	Supported
Erase app data on lock	X
Active poll period (minutes)	X

App Interaction policies	Supported
URL domains excluded from filtering	X
Allowed Secure Web domains	X

App Restrictions policies	Supported
Block mic record	X
Block SMS compose	X
Block NFC	X

App Network Access policies	Supported
Network access	X
micro VPN session required	X
Exclusion List	X

App Logs	Supported
Default log output	X
Default log level	X
Max log files	X
Max log file size	X
Redirect app logs	X

App Geofence policies	Supported
Center point longitude	X
Center point latitude	X
Radius	X

MDX policies for third-party apps for iOS

June 16, 2021

This article describes the MDX policies for third-party iOS apps. You can change policy settings directly in the policy XML files or in the Citrix Endpoint Management console when you add an app.

Authentication

Device passcode

If **On**, a PIN or passcode is required to unlock the device when it starts or resumes after a period of inactivity. A device passcode is required to encrypt app data using Apple file encryption. Data for all apps on the device are encrypted. Default value is **Off**.

App passcode

If **On**, a PIN or passcode is required to unlock the app when it starts or resumes after a period of inactivity. Default value is **On**.

To configure the inactivity timer for all apps, set the **INACTIVITY_TIMER** value in minutes in **Client Properties** on the **Settings** tab. The default inactivity timer value is **60** minutes. To disable the inactivity timer, so that a PIN or passcode prompt appears only when the app starts, set the value to zero.

> **Note:**
>
> If you select **Secure offline** for the Encryption keys policy, this policy is automatically enabled.

Online Session Required

Maximum offline period (hours)

Defines the maximum period an app can run without reconfirming app entitlement and refreshing policies from Citrix Endpoint Management. At expiration, log on to the server may be triggered if needed. Default value is **168** hours (7 days). Minimum period is **1** hour.

Alternate Citrix Gateway

> **Note:**
>
> This policy name in the Endpoint Management console is **Alternate NetScaler Gateway**.

Address of a specific alternate Citrix Gateway that should be used for authentication and for micro VPN sessions with this app. This is an optional policy that, when with the Online session required policy, forces apps to reauthenticate to the specific gateway. Such gateways would typically have different (higher assurance) authentication requirements and traffic management policies. If left empty, the server's default gateway is always used. Default value is empty.

Device security

Block jailbroken or rooted

If **On**, the app is locked when the device is jailbroken or rooted. If **Off**, the app can run even if the device is jailbroken or rooted. Default value is **On**.

Network requirements

Require Wi-Fi

If **On**, the app is locked when the device is not connected to a Wi-Fi network. If **Off**, the app can run if the device has an active connection, such as a 4G/3G, LAN, or Wi-Fi connection. Default value is **Off**.

Allowed Wi-Fi Networks

Comma-delimited list of Wi-Fi networks. If the network name contains any non-alphanumeric characters (including commas), the name must be enclosed in double-quotes. The app runs only if connected to one of the networks listed. If blank, all networks are allowed. This does not affect connections to cellular networks. Default value is blank.

Miscellaneous access

App update grace period (hours)

Defines the grace period that an app can continue to be used after the system has discovered that an app update is available. Default value is **168 hours (7 days)**.

> **Note:**
>
> Using a value of zero is not recommended since it immediately prevents a running app from being used until the update is downloaded and installed (without any warning to the user). This might lead to a situation where the user is forced to exit the app (potentially losing work) to comply with the required update.

Erase app data on lock

Erases data and resets the app when the app is locked. If **Off**, app data is not erased when the app is locked. Default value is **Off**.

An app can be locked for any of the following reasons:

- Loss of app entitlement for the user
- App subscription removed
- Account removed
- Secure Hub uninstalled
- Too many app authentication failures
- Jailbroken device detected (per policy setting)
- Device placed in locked state by other administrative action

Active poll period (minutes)

When an app starts, the MDX Framework polls Citrix Endpoint Management to determine current app and device status. Assuming the server running Endpoint Management can be reached, the framework returns information about the lock/erase status of the device and the enable/disable status of the app. Whether the server can be reached or not, a subsequent poll is scheduled based on the active poll period interval. After the period expires, a new poll is again attempted. Default value is **60 minutes** (1 hour).

> Important:
>
> Only set this value lower for high-risk apps or performance may be affected.

Non-compliant device behavior

Allows you to choose an action when a device does not adhere to the minimum compliance requirements. Select **Allow app** for the app to run normally. Select **Allow app after warning** for the app to run after the warning appears. Select **Block** to block the app from running. Default value is **Allow app after warning**.

App interaction

Cut and Copy

Blocks, permits, or restricts Clipboard cut and copy operations for this app. If **Restricted**, the copied Clipboard data is placed in a private Clipboard that is only available to MDX apps. Default value is **Restricted**.

Paste

Blocks, permits, or restricts Clipboard paste operations for this app. If **Restricted**, the pasted Clipboard data is sourced from a private Clipboard that is only available to MDX apps. Default value is **Unrestricted**.

Document exchange (Open In)

Blocks, permits, or restricts document exchange operations for this app. If **Restricted**, documents can be exchanged only with other MDX apps.

If **Unrestricted**, set the Enable encryption policy to **On** so that users can open documents in unwrapped apps. If the receiving app is unwrapped or has encryption disabled, Citrix Endpoint Management decrypts the document.
Default value is **Restricted**.

Restricted Open-In exception list

When the Document exchange (Open In) policy is **Restricted**, an MDX app can share documents with this comma-delimited list of unmanaged app IDs, even if the Document exchange(Open In) policy is **Restricted** and Enable encryption is **On**. The default exception list allows Office 365 apps:

com.microsoft.Office.Word,com.microsoft.Office.Excel,com.microsoft.Office.Powerpoint,
com.microsoft.onenote,com.microsoft.onenoteiPad,com.microsoft.Office.Outlook

Only Office 365 apps are supported for this policy.

> Caution:
>
> Be sure to consider the security implications of this policy. The exception list allows content to travel between unmanaged apps and the MDX environment.

Inbound document exchange (Open In)

Blocks, restricts, or allows inbound document exchange operations for this app. If **Restricted**, documents can be exchanged only with other MDX apps. Default value is **Unrestricted**.

If **Blocked** or **Restricted**, you can use the Inbound document exchange whitelistpolicy to specify apps that can send documents to this app.

> **Note:**
>
> The Inbound Document Exchange Whitelist policy only supports devices running on iOS 12.

Options: **Unrestricted**, **Blocked**, or **Restricted**

App URL schemes

iOS apps can dispatch URL requests to other apps that have been registered to handle specific schemes (such as "`http://`"). This facility provides a mechanism for an app to pass requests for help to another app. This policy serves to filter the schemes that are passed into this app for handling (that is, inbound URLs). Default value is empty, meaning that all registered app URL schemes are blocked.

The policy should be formatted as a comma-separated list of patterns in which each pattern may be preceded by a plus "+" or minus "-". Inbound URLs are compared against the patterns in the order listed until a match is found. Once matched, the action taken is dictated by the prefix.

- A minus "-" prefix blocks the URL from being passed into this app.
- A plus "+" prefix permits the URL to be passed into the app for handling.
- If neither "+" or "-" is provided with the pattern, "+" (allow) is assumed.
- If an inbound URL does not match any pattern in the list, the URL is blocked.

The following table contains examples of App URL schemes:

Scheme	App that requires the URL scheme	Purpose
ctxmobilebrowser	Secure Web-	Permit Secure Web to handle HTTP: URLs from other apps.-
ctxmobilebrowsers	Secure Web-	Permit Secure Web to handle HTTPS: URLs from other apps.
ctxmail	Secure Mail-	Permit Secure Mail to handle mailto: URLs from other apps.
COL-G2M	GoToMeeting-	Permit a wrapped GoToMeeting app to handle meeting requests.
ctxsalesforce	Citrix for Salesforce-	Permit Citrix for Salesforce to handle Salesforce requests.

Scheme	App that requires the URL scheme	Purpose
wbx	WebEx	Permit a wrapped WebEx app to handle meeting requests.

App interaction (outbound URL)

Domains excluded from URL filtering

This policy excludes outbound URLs from any "Allowed URLs" filtering. Add a comma-separated list of fully qualified domain names (FQDN) or DNS suffixes to exclude from the "Allowed URLs" filtering. If this policy is empty (the default), the defined "Allowed URLs" filtering processes are URLs. If this policy contains any entries, those URLs with host fields matching at least one item in the list (via DNS suffix match) are sent unaltered to iOS, bypassing the "Allowed URLs" filtering logic. Default value is empty.

Allowed URLs

iOS apps can dispatch URL requests to other applications that have been registered to handle specific schemes (such as `"http://"`). This facility provides a mechanism for an app to pass requests for help to another app. This policy serves to filter the URLs that are passed from this app to other apps for handling (that is, outbound URLs).

The policy should be formatted as a comma-separated list of patterns in which each pattern may be preceded by a plus "+" or minus "-". Outbound URLs are compared against the patterns in the order listed until a match is found. Once matched, the action taken is dictated by the prefix. A minus "-" prefix blocks the URL from being passed out to another app. A plus "+' prefix permits the URL to be passed out to another app for handling. If neither "+" or minus "-" is provided with the pattern, "+" (allow) is assumed. A pair of valus separated by "=" indicates a substitution where occurrences of the first string are replaced with the second. You can use regular-expression "^" prefix to search string to anchor it to the beginning of the URL. If an outbound URL does not match any pattern in the list, it will be blocked.

Default

+maps.apple.com

+itunes.apple.com

^http:=ctxmobilebrowser:

^https:=ctxmobilebrowsers:

^mailto:=ctxmail:

+^citrixreceiver:

+^telprompt:

+^tel:

+^lmi-g2m:

+^maps:ios_addr

+^mapitem:

+^sms:

+^facetime:

+^ctxnotes:

+^ctxnotesex:

+^ctxtasks:

+^facetime-audio:

+^itms-apps:

+^ctx-sf:

+^sharefile:

+^lync:

+^slack:

If the setting is blank, all URLs are blocked, except for the following:

- http:
- https:
- +citrixreceiver: +tel:

The following table contains examples of allowed URLs:

URL format	Description
^mailto:=ctxmail:	All mailto: URLs open in Secure Mail.
^http:	All HTTP URLs open in Secure Web.
^https:	All HTTPS URLs open in Secure Web.
^tel:	Allows user to make calls.
-//www.dropbox.com	Blocks Dropbox URLs dispatched from managed apps.

URL format	Description
+^COL-G2M:	Permits managed apps to open the GoToMeeting client app.
-^SMS:	Blocks the use of a messaging chat client.
-^wbx:	Blocks managed apps from opening the WebEx client app.
+^ctxsalesforce:	Permits Citrix for Salesforce to communicate with your Salesforce server.

Allowed Secure Web domains

This policy only affects "Allowed URLs" policy entries that would redirect a URL to the Secure Web app (^ http:=ctxmobilebrowser: and ^https:=ctxmobilebrowsers:). Add a comma-separated list of fully qualified domain names (FQDN) or DNS suffixes allowed to redirect to the Secure Web app. If this policy contains any entries, then only those URLs with host fields matching at least one item in the list (via DNS suffix match) redirect to the Secure Web app. All other URLs are sent unaltered to iOS, bypassing the Secure Web app. Default value is empty.

App Restrictions

> Important:
>
> Be sure to consider the security implications of policies that block apps from accessing or using phone features. When those policies are **Off**, content can travel between unmanaged apps and the Secure environment.

Block camera

If **On**, prevents an app from directly using the camera hardware. Default value is **OFF**.

Block Photo Library

If **On**, prevents an app from accessing the Photo Library on the device. Default value is **On**.

Block mic record

If **On**, prevents an app from directly using the microphone hardware. Default value is **On**.

Block dictation

If **On**, prevents an app from directly using dictation services. Default value is **On**.

Block location services

If **On**, prevents an app from using the location services components (GPS or network). Default value is **Off** for Secure Mail.

Block SMS compose

If **On**, prevents an app from using the SMS compose feature used to send SMS/text messages from the app. Default value is **On**.

Block email compose

If **On**, prevents an app from using the email compose feature used to send email messages from the app. Default value is **On**.

Block iCloud

If **On**, prevents an app from using iCloud for the storing and sharing of settings and data.

> **Note:**
>
> iCloud data file is controlled by the Block file backup policy.

Default value is **On**.

Block look up

If **On**, prevents an app from using the Look Up feature, which searches for highlighted text in the Dictionary, iTunes, the App Store, movie showtimes, nearby locations and more. Default value is **On**.

Block file backup

If **On**, prevents data files from being backed up by iCloud or iTunes. Default value is **On**.

Block AirPrint

If **On**, prevents an app from using AirPrint features for printing data to AirPrint-enabled printers. Default value is **On**.

Block AirDrop

If **On**, prevents an app from using AirDrop. Default value is **On**.

Block Facebook and Twitter APIs

If **On**, prevents an app from using the iOS Facebook and Twitter APIs. Default value is **On**.

Obscure screen contents

If **On**, when users switch apps, the screen is obscured. This policy prevents iOS from recording screen contents and displaying thumbnails. Default value is **On**.

Block 3rd party keyboards (iOS 11 and later only)

If **On**, prevents an app from using third-party keyboard extensions on iOS 8+. Default value is **On**.

Block app logs

If **On**, prohibits an app from using the mobile productivity app diagnostic logging facility. If **Off**, app logs are recorded and may be collected by using the Secure Hub email support feature. Default value is **Off**.

App network access

Network access

> **Note:**
>
> **Tunneled - Web SSO** is the name for the **Secure Browse** in the settings. The behavior is the same.

The settings options are as follows:

- **Blocked**: All network access is blocked. Networking APIs used by your app fail. Per the previous guideline, you should gracefully handle such a failure.
- **Unrestricted**: All network calls go directly and are not tunneled.
- **Tunneled - Web SSO**: The HTTP/HTTPS URL is rewritten. This option allows only the tunneling of HTTP and HTTPS traffic. A significant advantage of **Tunneled - Web SSO** is single sign-on (SSO) for HTTP and HTTPS traffic and also PKINIT authentication. On Android, this option has low setup overhead and is thus the preferred option for web browsing types of operations.

If the **Tunneled - Web SSO** option is selected, a per-app VPN tunnel in this initial mode is created back to the enterprise network, and Citrix Gateway split tunnel settings are used. Citrix recommends **Tunneled - Web SSO** for connections that require single sign-on (SSO).

micro VPN session required

If **Yes**, the user must have a connection to the enterprise network and an active session. If **No**, an active session is not required. Default value is **Use Previous Setting**. For newly uploaded apps, the default value is **No**. Whichever setting was selected prior to the upgrade to this new policy remains in effect until an option other than **Use Previous Setting** is selected.

micro VPN session required grace period (minutes)

This value determines how many minutes users can use the app before the Online Session Required policy prevents them from further use (until the online session is validated). Default value is **0** (no grace period). This policy isn't applicable for integration with Microsoft Intune/EMS.

Certificate label

When used with the StoreFront certificate integration service, this label identifies the specific certificate required for this app. If no label is provided, a certificate is not made available for use with a public key infrastructure (PKI). Default value is empty (no certificate used).

Exclusion List

Comma-delimited list of FQDNs or DNS suffixes to be accessed directly instead of through a VPN connection. This only applies to the **Tunneled - Web SSO** mode when Citrix Gateway is configured with Split tunnel reverse mode.

App logs

Default log output

Determines which output media are used by mobile productivity app diagnostic logging facilities by default. Possibilities are file, console, or both. Default value is **file**.

Default log level

Controls default verbosity of the mobile productivity app diagnostic logging facility. Each level includes levels of lesser values. Range of possible levels includes:

- 0 - Nothing logged
- 1 - Critical errors
- 2 - Errors
- 3 - Warnings
- 4 - Informational messages

- 5 - Detailed informational messages
- 6 through 15 - Debug levels 1 through 10

Default value is level 4 (Informational messages).

Max log files

Limits the number of log files retained by the mobile productivity app diagnostic logging facility before rolling over. Minimum is 2. Maximum is 8. Default value is 2.

Max log file size

Limits the size in megabytes (MB) of the log files retained by the mobile productivity app diagnostic logging facility before rolling over. Minimum is 1 MB. Maximum is 5 MB. Default value is 2 MB.

Redirect system logs

If **On**, intercepts and redirects system or console logs from an app to the Mobile Productivity Apps diagnostic facility. If **Off**, app use of system or console logs is not intercepted.

Default value is **On**.

App geofence

Center point longitude

Longitude (X coordinate) of the center point of point/radius geofence in which the app is constrained to operate. When operated outside the configured geofence, the app remains locked.

Should be expressed in signed degrees format (DDD.dddd), for example "-31.9635". West longitudes should be prefaced with a minus sign. Default value is **0**.

Center point latitude

Latitude (Y coordinate) of the center point of point/radius geofence in which the app is constrained to operate. When operated outside the configured geofence, the app remains locked.

Should be expressed in signed degreed format (DDD.dddd), for example "43.06581". Southern latitudes should be prefaced with a minus sign. Default value is **0**.

Radius

Radius of the geofence in which the app is constrained to operate. When operated outside the configured geofence, the app remains locked.

Should be expressed in meters. When set to zero, the geofence is disabled. When the Block location serviced policy is enabled, geofencing does not work properly. Default is **0** (disabled).

Analytics

Google Analytics level of detail

Citrix collects analytics data to improve product quality. Selecting **Anonymous** opts users out of including company identifiable information. Default is Complete.

Reporting

Citrix reporting

If **On**, Citrix collects crash reports and diagnostics to help troubleshoot issues. If **Off**, Citrix doesn't collect data.

> **Note:**
>
> Citrix might also control this feature with a feature flag. Both the feature flag and this policy must be enabled for this feature to function.

Default value is **Off**.

Upload token

You can obtain an upload token from your Citrix Insight Services (CIS) account. If you specify this optional token, CIS gives you access to crash reports and diagnostics uploaded from your devices. Citrix has access to that same information. Default value is empty.

Send reports over Wi-Fi only

If **On**, Citrix sends crash reports and diagnostics only when you're connected to a Wi-Fi network. Default value is **On**.

Report file cache maximum

Limits the size of the crash report and diagnostics bundles retained before clearing the cache. Minimum is 1 MB. Maximum is 5 MB. Default value is 2 MB.

MDX developer guide

January 23, 2019

Citrix Endpoint Management is an enterprise solution that lets you manage mobile devices, apps, and data. The basic premise of Endpoint Management mobile app management (MAM) is that it injects enterprise functionality into preexisting apps, which are then hosted on a company's private app store, the Apple App Store, or the Google Play Store.

To add Endpoint Management enterprise functionality to mobile apps, you wrap them with the MDX Toolkit. The MDX Toolkit is an app container technology that enhances the mobile device experience and prepares apps for secure deployment with Endpoint Management by adding MDX capabilities. The MDX capabilities include policies and settings, signed security certificates, and mobile app management code.

The MDX Toolkit includes the MDX App SDK, which delivers a complete set of MDX capabilities to your mobile apps through the Citrix MDX app container technology. APIs enable you to:

- Perform actions in wrapped apps based on Endpoint Management policies. For example, if an Endpoint Management policy prevents cut and copy in a MDX app, you can prevent text selection in your app. Your app can communicate and share policies with other MDX-enabled apps.
- Detect activities within your MDX-enabled apps. For example, you can check whether an app is wrapped or managed.
- Add custom functionality, such as security and policy enforcement.
- Develop mobile apps that will run either inside or outside a Citrix environment.

  In addition to being centrally configurable with MDX policies when used with Endpoint Management, apps that use the MDX App SDK can operate standalone outside of Citrix environments.

What's new in the MDX App SDK 10.2

The 10.2 release of MDX App SDK for iOS includes these enhancements and updates.

The Minimum data protection class policy is hidden. To make the policy visible in Citrix Endpoint Management, open the policy_metadata.xml file for the app (in Applications/Citrix/MDXToolkit/data) and, in the **MinimumDataProtectionClass** section, change the value of **PolicyHidden** to **false**. After you wrap your app, the policy appears when you add the app to Citrix Endpoint Management.

App wrapping integration with Xcode build process. Developers can now wrap and publish an iOS app as part of the Xcode build process. For details, see Integrating the SDK into your App Library.

Support for shared vault in Android apps. The MDX App SDK now includes the Android API for the MDX shared vault feature, enabling you to share managed content between apps. For example, the shared vault enables the sharing of certificates and private keys through an enrolled app so that apps can obtain a certificate from the secure vault instead of from Secure Hub. For details, see XenMobile API for Android.

> **Note:**
>
> Developers, be sure to test wrapped apps that perform background processing, such as content refreshes on a locked device or background syncs.

MAM capabilities

The enterprise functionality added by Endpoint Management is controlled through policies that administrators update on a per-app basis from the Endpoint Management console. Endpoint Management pushes policies to mobile devices on the schedule determined by administrators. Policies manage features, such as the following:

- **Authentication**. When opening a managed app, Endpoint Management can require users to enter corporate credentials or a PIN. This credential challenge can be repeated on a periodic basis.
- **App updates**. Endpoint Management notifies users when updates to managed apps are available. The administrator can make updates mandatory within a certain time period. If a user doesn't accept an update, the old version of the app will not execute after the time period elapses.
- **Remote locking and wiping**. An administrator can temporarily lock or permanently wipe apps on a per-app or per-device basis.
- **Network restrictions and VPN**. An Endpoint Management policy controls network access: Access can be either blocked, routed through a full VPN, or routed through a proxy VPN. VPN routing is through a Citrix Gateway device hosted by the enterprise.
- **Communication restrictions between apps**. An Endpoint Management policy determines whether document sharing between apps is blocked or permitted only between managed apps. Thus, the "Open In" pop-up in your app can omit unmanaged apps.
- **Feature containment**. An Endpoint Management policies can disable various device capabilities for an app. Examples include the camera, microphone, and location sensor.

Endpoint Management components

The following Endpoint Management components provide MAM functionality.

- **Citrix Endpoint Management server**

  This enterprise or cloud resident server hosts the Citrix Endpoint Management Store, the internal app store. Administrators upload mobile apps to Endpoint Management and then configure app and device policies.

- **Secure Hub**

  Enterprise users install Secure Hub for Android or iOS on their mobile device and then configure the app with a device enrollment URL and credentials. When Secure Hub opens, users select enterprise apps from the Citrix Endpoint Management Store. After the apps download and install on the device, Secure Hub serves as a hub for managing these apps, performing tasks, such as user authentication and updates of centrally administered policies.

- **MDX**

MDX is the source of the MAM functionality. The MDX Toolkit adds MDX code to your mobile app. Other than wrapping apps, you don't directly work with the MDX code.

- **MDX Toolkit and MDX App SDK**

  The MDX Toolkit adds enterprise functionality to existing mobile apps, a process called app wrapping. The MDX App SDK lets developers and system integrators MDX-enable their mobile apps. Application wrapping performs three main tasks. First, it injects Citrix code into your app that implements the app management capabilities. The output of that task is a new app file. Second, app wrapping signs the new app file with a security certificate. Finally, app wrapping creates an MDX file, which contains policy information and other settings. In some situations, the signed app file is also directly contained in the MDX file.

This developers guide focuses on app wrapping for ISVs.

Unmanaged and managed modes for ISV Apps

The MDX App SDK offers dual-mode app behavior, enabling you to deploy apps that can run with or without the MDX infrastructure. Apps that are run independently of Secure Hub are referred to as unmanaged apps. When those apps meet certain conditions, they transition to managed apps and run under the control of Secure Hub.

The dual-mode behavior is in contrast with the MDX Apps deployed from the Endpoint Management backend directly. Those apps always require the presence of Citrix MDX and authorization from an Endpoint Management Store to run. With Intune, however, those apps can be deployed and managed without Secure Hub or the Endpoint Management Store being present.

You use the Endpoint Management APIs to specify the type of dual-mode behavior needed when integrating an app with MDX. You can either develop two versions of app, one that is unmanaged and one that is managed, or a single app for both independent use and for inclusion in MDX. The MDX framework enforces the default behaviors associated with unmanaged and managed apps.

How an app transitions from unmanaged to managed depends on whether the app is wrapped as a General app or a Premium app:

- **General app:** A General app is hosted on the Apple App Store or the Google Play Store. Users who don't have Secure Hub can download and run the app normally in an unmanaged mode, just like any generic app store app. If an unmanaged user later installs Secure Hub, the ISV app transitions to managed mode if these conditions are met.

  - The user signs on to a Citrix Endpoint Management enterprise store at least once.
  - The user is in an Endpoint Management delivery group to which the app is deployed.
  - MDX subscribes the user.

  - When prompted, the user confirms that their enterprise can manage the app.

    If a user opts out of enterprise app management, they can continue to run the app for personal use.

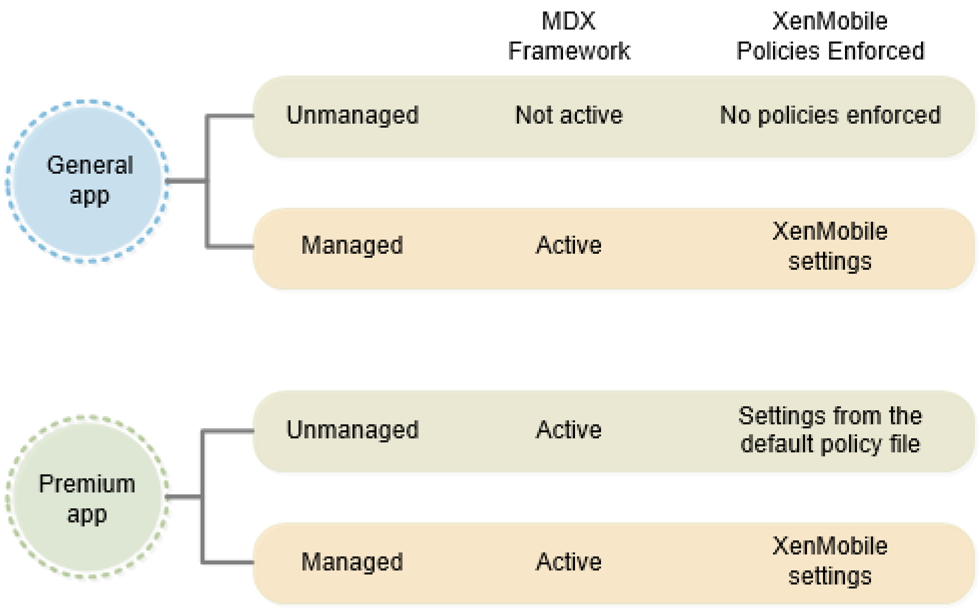
- **Premium app:** A Premium app is an app targeted to enterprise users. Citrix MDX Apps are examples of Premium apps. Although Premium apps typically run in managed mode, the embedded MDX framework allows Premium apps to run in unmanaged mode with a default set of MDX policies that you set through default policy files. Thus, you can effectively control the app behavior and use MDX capabilities even if the user is not associated with an enterprise account.

  If an unmanaged user later installs Secure Hub, the app silently transitions to managed mode if the following conditions are met.

  - The user is in a Citrix Endpoint Management delivery group to which the app is deployed.
  - The user signs on to Secure Hub if required.
  - MDX subscribes the user.

> **Note**
>
> An app cannot transition from managed mode back to unmanaged mode.

The following diagram summarizes the differences between General and Premium apps, based on whether they are managed or unmanaged.

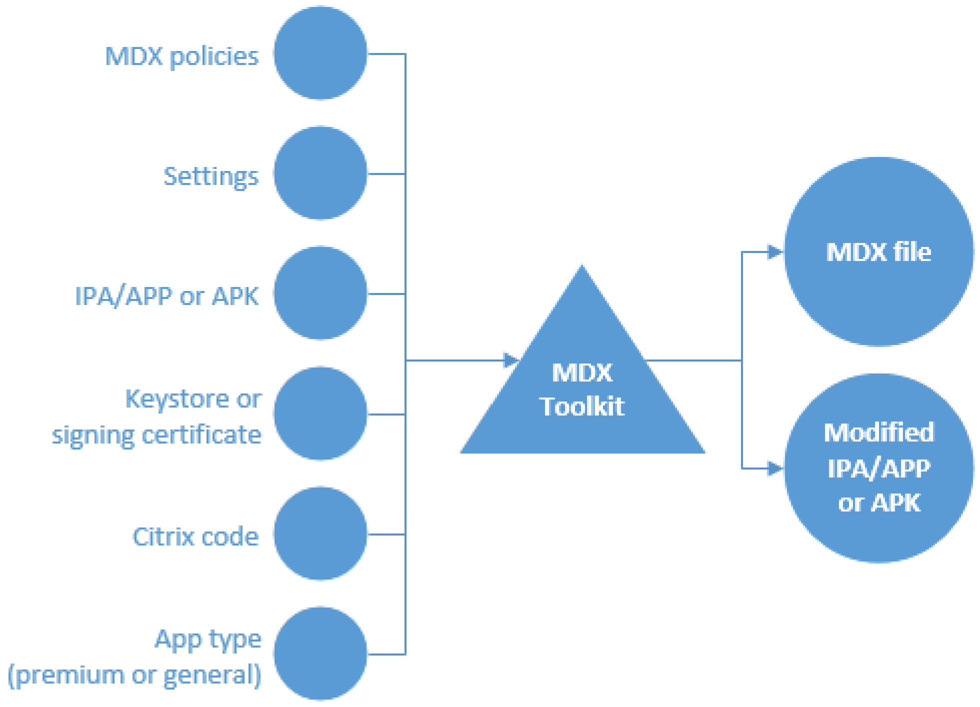


ISV app wrapping

This section provides general information about app wrapping for ISVs. App wrapping performed by enterprise administrators is discussed in About the MDX Toolkit.

When you wrap ISV apps, the MDX Toolkit creates two files: an .mdx file and the app file (.ipa, .app, or .apk). The MDX Toolkit lets you embed the app store URL into the .mdx file, which you then deliver directly to your customers or upload to the Citrix Ready Marketplace, as described in the next section. You deliver the app file through app stores, by hosting it yourself, or by distributing it to your customers.

As shown in the following diagram, the MDX Toolkit combines app files (.ipa, .app, or .apk) with Citrix components and your keystore or signing certificate to produce an .mdx file and the modified app file.



The items added by ISV app wrapping include:

- An information file containing data needed by the MDX SDK framework when the framework binds with Secure Hub. The corresponding binding information is passed to Secure Hub from the Endpoint Management server through the .mdx file added to Endpoint Management. The data includes items, such as an app ID used for self-identification and a package ID used for app update checks.
- A FIPS fingerprint on the OpenSSL FIPS Crypto Object Module embedded in the app integrated with MDX App SDK.
- For iOS only: A new URL scheme that is added to the app file and is also passed to Secure Hub through the .mdx file an administrator adds to Endpoint Management.

About the Citrix Ready Program

Citrix evaluates and certifies ISV apps through the Citrix Ready program. The evaluation largely involves Endpoint Management integration testing. The certification ensures that the apps are compatible with the Endpoint Management infrastructure, thus giving enterprises confidence in your apps.

As part of the Citrix Ready program, you can publish your certified ISV app binaries directly in the Apple App Store or the Google Play Store. That means you don't need to distribute binaries to enterprises, giving you more control over app updates. In addition, your apps are signed with your ISV certificate. You can also choose to distribute your certified apps directly to enterprises or to host them yourself.

You can also choose how to distribute the .mdx bundle for an ISV app: Either publish the bundle to the Citrix Ready Marketplace or distribute the bundle directly to your Citrix Endpoint Management customers.

For details, see Citrix Ready.

MDX app user experience

The way users interact with an app that is integrated with the MDX App SDK depends on how they install and start the app.

User starts with Secure Hub

1. The user opens Secure Hub and the Apple App Store or Google Play Store.
2. The user signs on to the Endpoint Management Store and then subscribes to the store.
3. The user downloads and installs the app from a public store.
4. Secure Hub prompts the user to sign on, if needed.
5. If the app was previously unmanaged, it silently transitions to managed.

User starts with the Apple App Store or Google Play Store

If Secure Hub is present on device already, users have the following experience for General and Premium apps.

General app

1. The user starts the app.
2. If the app detects an installation of Secure Hub and the app is entitled, the app prompts the user to confirm the transition to managed mode.
3. If the user opts to have their enterprise manage the app, Secure Hub prompts the user to sign on if needed.
4. After MDX subscribes the app to the user, the app transitions to managed mode.

   If Secure Hub isn't on the device or the app isn't entitled, the app runs in unmanaged mode, just like a regular public store app.

Premium app

1. The user starts the app.
2. If the app detects an installation of Secure Hub and the app is entitled, the app silently transitions to managed mode. If Secure Hub credentials are required, the app notifies the user about the transition to managed mode and prompts the user to sign on.

Known issues for the MDX App SDK

- Wrapping doesn't work for Android apps unless they include icons.
- Some app frameworks have compatibility issues with Citrix Endpoint Management. For details, see Mobile App Development Frameworks Support (for Android) and Third Party Library Support (for iOS).
- For other issues, see Known issues.

System requirements

April 7, 2021

This article includes the system requirements for the MDX Toolkit and the MAM SDK. For more information about the MAM SDK APIs, see the developer documentation for Mobile Application Integration.

MDX Toolkit and MDX App SDK (iOS and Android)

- Java Development Kit (JDK) 1.7 or 1.8.

  You can download the JDK 1.8 from Java SE Development Kit Downloads on the Oracle web site. For installation instructions, see the JDK 8 and JRE 8 Installation Guide on the Oracle web site. Be sure to install the full JDK; set JDK 1.8 as the default.

- macOS 10.10

  The installer for the MDX Toolkit and MDX App SDK must run on macOS. The installer includes macOS tools that wrap both iOS and Android apps, as well as a Java command-line tool that wraps Android apps.

- For the MDX App SDK: iOS 11 or later with Xcode 9, with bitcode generation disabled. (We recommend that you use the most recent version of Xcode that is available from Apple.)

  Bitcode generation is on by default. You must disable it to use Xcode 9 with the MDX App SDK.

Other requirements for wrapping iOS mobile apps

- To obtain access to the app wrapping prerequisites for iOS, you must register for an Apple distribution account. There are three types of iOS developer accounts: Enterprise, Individual, and University. Citrix strongly recommends iOS Developer Enterprise accounts.
  - iOS Developer Enterprise accounts: The only type of Apple Developer account that allows you to provision, deploy, and test unlimited apps to unlimited devices, with or without app wrapping. Be sure to distribute your Developer Certificate to your developers so they can sign apps.
  - iOS Developer Individual accounts: Limited to 100 registered devices per year and do not qualify for app wrapping and enterprise distribution with Citrix Endpoint Management.
  - iOS Developer University accounts: Limited to 200 registered devices per year and do not qualify for app wrapping and enterprise distribution with Endpoint Management.

> **Note:**
>
> Download the Xcode command-line tools from the Xcode Apple Developer web site. macOS 10.10 does not install the tools automatically. To install the tools, follow these steps:

1. In **Applications > Utilities**, click **Terminal** to use the Mac command-line interface.
2. Type the following command:

   ```
   xcode-select --install
   <!--NeedCopy-->
   ```

   Be sure to include two hyphens before the word install in the command.
3. After the Xcode command-line tools install, run Xcode to install any pre-requisites.

Other requirements for wrapping Android mobile apps

- Android Software Development Kit (SDK), API Level 21 (minimum supported version).
  - Download the Android SDK from the SDK download page on the Google developer website.
  - Install the latest Android SDK Tools, Android SDK Platform-tools, and Android SDK Build-tools.

    For details, see Installing the Android SDK on the Google developer website.
  - Edit the android_settings.txt file located in the MDX Toolkit installation folder. Set the PATH variable to include the Android SDK Build Tools to be used during wrapping. Add the path to the platform-tools and tools subdirectories of the Android SDK. See the following example:

PATH = /Users/Sample/Downloads/android-sdk-macosx/platform-tools:/Users/Sample/Downloads/an sdk-macosx/build-tools/28.0.2:/Users/Sample/Downloads/android-sdk-macosx/tools

- Valid keystore (containing digitally signed certificates used to sign your Android apps)

  You create a keystore one time and retain this file for current and future wrapping. If you do not use the same keystore when wrapping the new version of an app you previously deployed, upgrades of that app won't work. Instead, users need to manually remove the older version before installing the new version.

  A keystore can contain multiple private keys; in most cases, the keystore will only have one key.

  For details about certificates, see Signing Your Applications on the Android Developers website.

  You must sign your apps with a key that meets the following guidelines:

  - 2048-bit key size
  - DSA key algorithm (-keyalg)
  - SHA1 with DSA signing algorithm (-sigalg)

Developing Android apps

June 2, 2020

You can use the XenMobile API in your mobile apps to allow the apps to interact with Citrix Endpoint Management. This article describes how to integrate the MDX App SDK into your app library and the steps required to test, certify, and publish your apps.

How to use the MDX App SDK

Here are some examples of how you might use the APIs.

Place restrictions on apps

You can control whether your app allows access to certain features or actions based on whether API calls indicate that the app is managed or wrapped. For example, if an app is not managed or wrapped, you might allow a user access to all features and actions. If an app is wrapped but not managed, you might then restrict certain features or actions. If an app is wrapped and managed, you might put additional restrictions on the app.

Perform actions based on Citrix Endpoint Management policy settings

Suppose that you want to display a notification to users if a Citrix Endpoint Management administrator sets the Require WiFi policy to On, which means that the app is allowed to run on a wireless network. You can use the API to look up the policy setting and then base your code changes on the policy value.

- Perform actions based on custom policies

  You can use the APIs to read custom policies in your apps. For example, suppose that you want to enable Citrix Endpoint Management administrators to display a notification in the app. To do that, you can create a custom policy that is empty by default or contains a system message that is supplied by an administrator in the Citrix Endpoint Management console. When your app is managed, it can detect when the Citrix Endpoint Management administrator changes the policy value. If the policy value contains a message, your app displays the notification.

For API definitions, see API for Android.

Integrating the SDK into your app library by using Android Studio and Gradle

To add the MDX App SDK to your Android apps, you import or copy the MDX App Java libraries into your app, as described in this section. The steps are based on Android Studio and the Gradle build system. Going through the steps adds the MDX Apps SDK library to your application so that its classes and methods are accessible to the app.

1. If you haven't already installed the latest MDX Toolkit, do so now.
   a) Log on to the Citrix Endpoint Management downloads page.
   b) Expand **XenMobile Apps and MDX Toolkit**
   c) Locate the MDX Toolkit version you want to install and then click its link to begin the download.
   d) Open MDXToolkit.mpkg with the macOS Finder tool on macOS 10.9.4 or later and Xcode 5.1 or later.

   The installation path is Applications/Citrix/MDXToolkit.

   The MDX App SDK files are in Applications/Citrix/MDXToolkit/data/MDXSDK_Android.
2. After installing the MDX Toolkit, install Android Studio from the Android developers website and then do the following:
   a) In the project directory, create a folder named libs.
   b) Add the file worxsdk.aar to the libs folder.
   c) Edit the project "build.gradle" to add a rule to search the libs folder as a repository and to include worxsdk.aar from the libs folder as a dependency.

d) Build your APK file.

An example of worxsdk.aar:

```
// Top-level build file where you can add configuration options
   common to all sub-projects/modules.
buildscript {

    repositories {

        jcenter()
     }

    dependencies {

        classpath 'com.android.tools.build:gradle:1.1.0'
        // NOTE: Do not place your application dependencies here;
           they belong
        // in the individual module build.gradle files
     }

 }

allprojects {

    repositories {

        jcenter()
        flatDir {

            dirs 'libs'
         }

     }

 }

dependencies {

    compile(name:'worxsdk', ext:'aar')
```

The OpenSSL library might cause conflicts with similar libraries in Android apps. Citrix recommends that you use the Citrix versions of the libraries to avoid conflicts.

Publishing an Android app

After you add the MDX App SDK to an Android app, perform the following steps to wrap, test, certify, and publish the app. When wrapping apps by using the command-line interface, include one of the following options:

- **–appType Enterprise**
- **-appType Premium**
- **-appType General**

The default is **–appType Enterprise**. Use the following guidelines to select the appType:

- **Enterprise:** Apps require installing Secure Hub on the user device. Also, you must also publish the application in StoreFront and install the app through Secure Hub.
- **General:** ISV Apps can run without Secure Hub (initially). The application can transition to managed mode when the app detects Secure Hub on the user device and if you publish a matching app. When running as a managed app, General apps behave the same as Enterprise apps. When running as an unmanaged app, Citrix policies are not enforced.
- **Premium:** ISV apps can run without Secure Hub installed on the user device (initially). The app can transition to a managed app if it detects Secure Hub on the user device and if you publish a matching app. When the app runs unmanaged, MDX enforces some policies, such as the containment policies (allowing network access, screen capture, or blocking the camera).

If you need to upload the wrapped .apk file to an app store or web server, and you already know the URL, add the **-storeURL** option. You can also add the URL later, as indicated later in these steps.

The MDX Toolkit outputs a modified .apk file and a .mdx file. You will use those files in the following steps. Use the MDX Toolkit to wrap the .apk file for the app. For details, see Wrapping Android Mobile Apps in the MDX Toolkit documentation. That article includes all wrapping commands, including those specific to ISV apps.

> **Important:**
>
> The option to wrap ISV apps by using the MDX Toolkit user interface no longer available. You must wrap ISV apps by using the command line.

To test your app

1. Install the modified .apk file on an Android device to verify all app functions.
2. Use the Citrix Endpoint Management console to add the .mdx file to Citrix Endpoint Management and deliver it to an Android device for testing. For details, see To add an MDX app to Citrix Endpoint Management. On that device, test the MDX functionality of your app.

   If you added custom policies, be sure to verify that those policies appear in the Citrix Endpoint Management console and work as expected. If you changed default_sdk_policies.xml, test those changes. For details about adding policies and changing policy defaults, see Policy defaults and custom policies.

3. Fix any issues found in your app, regenerate the app .apk file, and wrap the app again with the MDX Toolkit.
4. Submit the original .apk file (not the one output by the MDX Toolkit) to Citrix for validation and certification.
5. After Citrix certifies your app, submit the .apk file generated by the MDX Toolkit to the Google Play Store for approval.
6. After Google approves your app, run the MDX Toolkit to update the app download URL in the .mdx file. An example command that changes the URL is as follows:

   ```
   java -jar /Applications/Citrix/MDXToolkit/ManagedAppUtility.jar
   setinfo
   -in ~/Desktop/SampleApps/Sample.mdx
   -out ~/Desktop/SampleApps/wrapped/Sample.mdx
   -storeURL
   "https://play.google.com/store/apps/details?id=com.zenprise"
   ```

Provide the final .mdx file to a Citrix Endpoint Management administrator, who will add it to Citrix Endpoint Management and publish it to users. Or, to make your app available for wider distribution, you can list your MDX verified app in the Citrix Ready Marketplace. For details, see Citrix Ready Verified Program.

Considerations for upgrading apps

The Citrix Endpoint Management software changes significantly between releases. To take advantage of the latest features and bug fixes, you must use the latest version of the MDX Toolkit to wrap your app. Be sure to wrap your original .ipa or .apk file, not the modified file that was previously generated by the MDX Toolkit.

Be sure to use the corresponding version of the MDX App SDK.

Best practices for Android apps

June 2, 2020

The best practices discussed in this article improve compatibility between Citrix Endpoint Management and mobile apps for Android devices.

MDX app SDK and wrapping

If your app uses the MDX App SDK, then you must use the matching MDX Toolkit version for wrapping. A version mismatch between these two components might cause improper operation.

To prevent such a mismatch, wrap the app with an app type of Premium or General. That lets you deliver a pre-wrapped app. As a result, your customer won't need to wrap the app, thus avoiding use of a mismatched MDX Toolkit. For details about wrapping apps, see Wrapping Android mobile apps.

Don't block the main thread

You should not use blocking code when running on the main thread. This is a Google guideline, but it is even more crucial with Citrix Endpoint Management. Some actions may take more time in a managed app or may even block further thread execution.

Blocking code includes, but is not limited, to the following:

- File or database operations
- Network operations

To be clear, all app lifecycle methods, such as onCreate, run on the main thread.

Google provides a StrictMode API which can help detect blocking code. For details, see this blog post: https://android-developers.blogspot.com/2010/12/new-gingerbread-api-strictmode.html.

Write robust code

In particular, you should check return values or catch exceptions from framework APIs. While this is just a common programming best practice, it is especially important for managed apps.

Various APIs that you'd expect to always work will fail if Citrix Endpoint Management policies block the underlying functionality. Examples would include any of the capabilities described earlier:

- Networking APIs fail as if there is no network available.
- Sensor APIs, such as GPS and camera, return null or throw an exception.
- Intents directed at a non-managed app fail.
- File and database access might fail if used from the main thread. For details, see the Ensure Data Encryption Compatibility and Encryption User Entropy sections, later in this article.

When you encounter a failure, your app should handle the issue gracefully instead of crashing.

Hooking limitations

MDX injects functionality into a binary Android app by modifying the DEX code in the APK. Several limits are present:

- Citrix Endpoint Management might not manage deprecated framework classes from the pre-4.0 Android SDK versions. Be sure to avoid those deprecated classes.
- Most functionality is injected into the Java/Android framework APIs. Native (C/C++) code is generally not managed. One exception is that even for native code, file encryption still occurs.
- Native code that uses JNI to access Java functionality must only target code in the user app. In other words, don't use JNI to directly invoke Java or Android framework methods. Instead, use the proxy design pattern to "wrap" the desired framework class in a Java class of your own. Then invoke your class from the native code.

Ensure data encryption compatibility

One of the primary features of MDX is that all persisted data is transparently encrypted. You don't need to modify your app to gain this functionality and, in fact, you can't directly avoid it. The administrator has the ability to disable encryption either selectively or entirely, but the app does not.

This is one of the more heavyweight aspects of MDX and requires an understanding of the following points:

- File encryption is present for all Java and native code that runs in managed processes.
- Some framework APIs, such as media players and printing support, actually run in separate OS processes. If you use such an API, you might encounter issues.
  - Example: Your app saves a file to disk (encrypted) and then passes a reference to the file to a media API. The media API tries to read the file but it doesn't understand the encrypted content. It fails or even crashes the app.
  - Example: You create a file handle (that starts an encrypted file) and give it to the camera API. The camera process directly writes unencrypted data into the encrypted file. When your app tries to read that data, the data is decrypted, yielding garbage.
- One method of handling separate processes is to decrypt a file before handing it to the relevant API. Or if the API writes data, then you'd let it write first and then you'd encrypt it when the API finished. A few steps are required:
  1. Designate an area that will remain unencrypted. You must document this for your customer, because a Citrix Endpoint Management administrator must create an encryption exclusion policy.
  2. To decrypt, you simply copy the file from the normal (encrypted) location to the decrypted location. Note that you must do a byte copy and not a file move operation.
  3. To encrypt, reverse the direction. Copy from unencrypted to encrypted locations.
  4. Delete the unencrypted file when no longer needed.
- Memory mapping is not supported for encrypted files. If you call an API that does memory mapping, it will fail. You should handle the error. If at all possible, avoid direct and indirect use of

memory mapping. One notable case of indirect use is the third-party SqlCipher library.

If you can't avoid memory mapping, the administrator must specify an encryption exclusion policy that omits the relevant files. You must document this policy for your customer.

- Encryption adds measurable overhead. Be sure to optimize file I/O to prevent performance degradation. As an example, if you are repeatedly reading and writing the same information, you might want to implement an app level cache.
- Databases are just files and so they are also encrypted. Performance can be an issue here too. The standard database cache size is 2000 pages or 8 megabytes. If your database is large, you might increase this size.

  SQLite WAL mode is not supported due to the memory mapping limitation.

Encryption user entropy

One Citrix Endpoint Management option for encryption requires the end user to enter a PIN before the encryption key can be generated. This option is called user entropy. It can cause a particular issue for apps.

Specifically, no file or database access can be performed until the user enters a PIN. If such an I/O operation is present in a location that runs before the PIN UI can be displayed, it will always fail. There are a few implications:

- Keep file and database operations off the main thread. For example, an attempt to read a file from the app object's onCreate() method will always fail.
- Background operations, such as services or content providers, may run even though no app activity is present. These background components can't display the PIN UI and therefore they can't perform file or database access. Note that once an activity runs in the app, the background operations are allowed to perform I/O operations.

There are several failure mechanisms if the encryption key isn't available due to user entropy:

- If the main thread accesses a database before the PIN is available, the app is killed.
- If a non-main thread accesses a database before the PIN is available, that thread is blocked until the PIN is entered.
- For non database access started before the PIN is available, the open operation will fail. At the C level, an EACCES error is returned. In Java, an exception is thrown.

To ensure that this issue isn't present in your app, test with user entropy enabled. The Citrix Endpoint Management client property, Encrypt secrets using Passcode, adds user entropy. You configure that client property, which is disabled by default, in the Citrix Endpoint Management console under **Configure > Settings > More > Client Properties**.

Networking and micro VPN

Several Citrix Endpoint Management policy options are available to administrators for networking. The Network access policy prevents, permits or redirects app network activity.

> Important:
>
> The MDX Toolkit version 18.12.0 release included new policies that combined or replaced older policies.
> The Network Access policy combines Network access, Preferred VPN mode, and Permit VPN mode switching. The Exclusion list policy replaces Split tunnel exclusion list. The micro VPN session required policy replaces Online session required. For details, see What's new in earlier releases.

The options are as follows:

- **Use Previous Settings**: Defaults to the values you had set in the earlier policies. If you change this option, you shouldn't revert to **Use Previous Settings**. Also note that changes to the new policies do not take effect until the user upgrades the app to version 18.12.0 or later.
- **Blocked**: Networking APIs used by your app will fail. Per the previous guideline, you should gracefully handle such a failure.
- **Unrestricted**: All network calls go directly and are not tunneled.
- **Tunneled - Full VPN**: All traffic from the managed app tunnels through Citrix Gateway.

Limitation: Citrix Endpoint Management doesn't support socket server. If a socket server is running inside the wrapped app, the network traffic to the socket server is not tunneled through Citrix Gateway.

Mobile app development frameworks support

Some app frameworks have compatibility issues with Citrix Endpoint Management:

- With PhoneGap, the location service is not blocked.
- SQLCipher doesn't work with encryption because it uses memory mapping. One solution is to not use SQLCipher. A second solution is to exclude the database file from encryption using an encryption exclusion policy. A Citrix Endpoint Management administrator must configure the policy in the Citrix Endpoint Management console.

Debugging tips

When debugging a wrapped app, consider these tips.

- Determine if the issue is present in an unwrapped version of the app. If the issue occurs when unwrapped, use normal debugging techniques.
- Try turning off various Citrix Endpoint Management policies.

  - This can help localize any incompatibility. Disabling a policy means that MDX no longer enforces the related restriction, thus enabling you to test those features as if the app were unwrapped.
  - If disabling a policy fixes the problem, the issue might be that the app isn't checking for errors in the associated APIs.
- If an unmodified but re-signed app doesn't run:
  1. Un-jar the contents of the APK using JAR:
     jar xvf {some.apk}
  2. Delete the META-INF folder:
     rm -rf META-INF
  3. Re-jar the contents into a new APK using JAR:
     jar cvf {/tmp/new.apk} *
  4. Sign the new APK using JARSIGNER:
     jarsigner -keystore {some.keystore} -storepass {keystorepassword} -keypass {keypass-word} {/tmp/new.apk} {keyalias}
  5. If the app still doesn't run, you cannot wrap the app using a different signing certificate than the original APK used.
- If a decompiled or recompiled .apk doesn't run:
  1. Decompile and recompile using APKTOOL:
     apktool d {some.apk} -o {some.directory}
     apktool b {some.directory} -o {new.apk}
  2. Sign the APK using JARSIGNER as described above.
  3. If the app still doesn't run, this is a third-party APKTOOL bug.
- If app wrapping doesn't work:
  1. Try removing the APKTOOL framework and rewrapping.
     - Mac/Linux: **rm -rf ~/Library/apktool/framework**
     - Windows: **del /q /s C:\Users\{username}\apktool\framework**
  2. Compare which APKTOOL is being used by the wrapper with the one you used to successfully decompile and recompile in the previous step.
     - If it is the same APKTOOL version, then there is a bug in Wrapper.
     - If it is a different APKTOOL version, then there might be a bug in the APKTOOL integrated into the MDX Toolkit utility.
       a) Un-jar the contents of ManagedAppUtility.jar.
       b) Overwrite with contents of APKTOOL.jar that you used to successfully wrapped the app in the previous step.
       c) Re-jar the contents into a new ManagedAppUtility.jar.
       d) Wrap the app to confirm the bug in the embedded APKTOOL.
- Run the wrapped app and capture log information.
  1. Use grep to investigate what is happening in the app.

To follow the app’s Activities: grep “MDX-Activity”
To follow MDX locking of the app: grep “MDX-Locked”

```
To see both logs together: egrep “MDX-Act            MDX-Loc”
```

2. If there is an Application Not Responding error, pull the ANR traces using ADB.

- If a problem occurs when interacting with multiple apps, such as when using Open in:
    1. Verify encryption policies and security group settings are the same between the apps.
    2. Try a different app. It might be a bug in one of the apps being tested.
    3. Capture logs from all apps involved. Note that Secure Hub can bundle logs and email logs from individual apps. From the My Apps screen, swipe right to the Support screen. Then click the Need Help button at the bottom of the screen.

In addition to the tools mentioned above, the following might also help:

- AAPT to dump information about the app.

  aapt dump badging {some.apk}

- DUMPSYS command on device.

  adb shell dumpsys 2>&1 | tee {dumpsys.out}

- DEX2JAR to recompile classes into pseudo-Java.

  dex2jar {some.apk}

  Convert classes from Dual-Dex wrapped apps:

  apktool d {some.apk} -o {some.dir}

  dex2jar {some.dir}/assets/secondary-1.dex

- JD-GUI to view pseudo-Java code.
- BAKSMALI to decompile app classes from Dual-Dex wrapped apps.
    - Decompile the wrapped APK:

      apktool d {some.apk} -o {some.dir}

    - Decompile the app’s classes that do not get decompiled from above call:

      baksmali {some.dir}/assets/secondary-1.dex -o {some.dir}/smali

API for Android

January 10, 2019

The API for Android is based on Java. This article summarizes the Citrix Endpoint Management APIs by feature and provides the API definitions.

App management:

- isManaged
- isWrapped

MDX policies:

- getPoliciesXML
- getPolicyValue
- setPolicyChangeMessenger

Shared vault:

- MDXDictionary

User data:

- getUserName

Class com.citrix.worx.sdk.MDXApplication

Methods

- **isManaged**

  public static boolean isManaged (Context context)

  Checks if the app is currently managed by MDX, which means that the Citrix Secure Hub app is installed on the device and Citrix Endpoint Management policies are enforced on your app. The Endpoint Management backend infrastructure (key vaults) are queried for data encryption partial keys (secrets) which MDX will use to encrypt application file data. Returns true if the app is managed.

  Unmanaged Premium apps use the Endpoint Management policy defaults specified in Applications/Citrix/MDXToolkit/data/MDXSDK_Android/default_sdk_policies.xml. Policies are not enforced for unmanaged General apps.

  Parameters

  context - The Android context that is making this call.

  Example

  ```
  boolean bIsManaged = MDXApplication.isManaged(context);
  ```

- **isWrapped**

  public static boolean isWrapped (Context context)

Returns true if the app is wrapped with the MDX Toolkit.

Parameters

context - The Android context that is making this call.

Example

```
boolean bIsWrapped = MDXApplication.isWrapped(context);
```

- **getUserName**

  public static String getUserName (Context context)

  Returns a string containing the user name of an enrolled user running an MDX-managed app, regardless of the user sign-on status. Returns nil if the user isn't enrolled, the app isn't managed, or the app isn't wrapped.

  Parameters

  context - The Android context that is making this call.

  Example

  ```
  String userName = MDXApplication.getUserName(context);
  ```

Class com.citrix.worx.sdk.MDXPolicies

Methods

- **getPoliciesXML**

  public static String getPoliciesXML (Context context)

  Returns the contents of default_sdk_policies.xml, as one line per policy, prefixed with (match) to indicate that the value in the XML file matches the value returned by **MDXPolicies.getPolicyValue()**. Returns an empty string on failure.

  Parameters

  context - The Android context that is making this call.

  Example

  ```
  String policiesXML = MDXPolicies.getPoliciesXML(context);
  ```

- **getPolicyValue**

  public static String getPolicyValue (Context context, String policyName)

  Returns a **String** which contains current value of the named policy. Returns **null** if no value is found.

  Parameters

context - The Android context that is making this call.

policyName - The name of the policy to search for. A policy name is the value of the *PolicyName* element in a policy XML file.

Example

```
String value = MDXPolicies.getPolicyValue(context"DisableCamera");
```

- **setPolicyChangeMessenger**

  public static String setPolicyChangeMessenger (Context context, String policyName. Messenger messenger)

  Registers a Messenger to receive a message when the given policy's value changes. When MDX detects that a policy value changed in the Citrix Endpoint Management console, MDX notifies this messenger. You can then use the other APIs to re-read the policy values and change your app. Returns **null**.

  Parameters

  context - The Android context that is making this call.

  policyName - The policy name to monitor. A policy name is the value of the *PolicyName* element in a policy XML file.

  messenger - The messenger that will receive messages when the policy value changes.

  Example

  ```
  MDXPolicies.setPolicyChangeMessenger(context, "DisableCamera", messenger
  );
  ```

Class com.citrix.mdx.common.MDXDictionary

MDXDictionary is a container for reading and storing encrypted Android bundles of key-value pairs. Mobile productivity apps in the same MDX security group share a dictionary. Use the shared vault API to share managed content between apps that have the same MDX dictionary. For example, you can share certificates and private keys through an enrolled app so that apps can obtain a certificate from the secure vault instead of from Secure Hub.

Dictionaries are stored encrypted regardless of the Private file encryption policy and Public file encryption policy settings. Developers must unlock the vault before retrieving dictionaries.

Constructors

- **public MDXDictionary(MDXDictionary source)**

  Constructs a copy of an existing MDXDictionary.

Parameters

source - The MDXDictionary that should be copied.

- **public MDXDictionary(String name, Bundle bundle, long sequence)**

  Constructs an MDXDictionary from a name, bundle, and sequence number. If you do not know the sequence number, use the **create() factory** method.

  Parameters

  name - The name of the dictionary.

  bundle – The Android bundle.

  sequence - A sequence number.

Methods

- **public static MDXDictionary create(Context context, String name)**

  Creates a dictionary by first checking if a dictionary with the same name already exists. If the dictionary does not exist, then a new dictionary is returned. Otherwise, the existing dictionary is returned. This method never returns **null**.

  Parameters

  context - The Android context that is making this call.

  name - The name of the dictionary.

  Example

  // Creates a instance of a dictionary.

```
MDXDictionary dict = MDXDictionary.create(getContext(), "app-settings")
;
```

- **public static boolean delete(Context context, String name)**

  Deletes a dictionary by name. Returns **true** on success; returns **false** on failure.

  Parameters

  context - The Android context that is making this call.

  name - The name of the dictionary.

  Example

  // Creates a instance of a dictionary.

```
MDXDictionary.delete(getContext(), "app-settings");
```

- **public static MDXDictionary find(Context context, String name)**

  Finds an existing dictionary. Returns an existing dictionary; returns **null** if no dictionary is found.

  Parameters

  context - The Android context that is making this call.

  name - The name of the dictionary.

  Example

  ```
  MDXDictionary dict = MDXDictionary.find(getContext(),"app-settings");
  ```

  ```
  if( dict != null )
      {

          // Use dictionary
       }
  ```

- **public boolean isNew( )**

  Checks whether this is a new dictionary or an existing dictionary. Returns **true** if a dictionary does not already exist.

  Example

  ```
  MDXDictionary dict = MDXDictionary.create(getContext(), "app-settings")
  ;
  ```

  ```
  if (dict.isNew())
      {

          // Dictionary was not found.
       }

  else
      {

          // Existing dictionary was found.
       }
  ```

- **public boolean save(Context context)**

  Stores an encrypted dictionary. If a dictionary with the same name exists, it will be overwritten. Returns **true** on success; returns **false** on failure.

  Parameters

context - The Android context that is making this call.

Example

```
MDXDictionary dict = MDXDictionary.find(getContext(), "app-settings");
```

```
if( dict != null )
    {

        String certificate = getCertificate();
        dict.bundle.putString( &quot;secret-certificate&quot;,
            certificate );
        // Update bundle by overwriting the existing bundle.
        dict.save( getContext() );
     }
```

- **public boolean append(Context context)**

  Appends an encrypted dictionary to an existing dictionary. If no dictionary exists, the specified dictionary is stored. Returns **true** on success; returns **false** on failure.

  Parameters

  context - The Android context that is making this call.

  Example

  ```
  MDXDictionary dict = MDXDictionary.find(getContext(), ";app-settings");
  ```

  ```
  if( dict != null )
      {

          String certificate = getCertificate();
          Bundle bundle = new Bundle();
          bundle.putString( &quot;secret-certificate&quot;,
              certificate );
          dict.bundle = bundle;
          dict.append( getContext() );
          // Note that dict.bundle may not match the state of the
          // bundle that was stored. The stored bundle could be
          // larger.
       }
  ```

- **public boolean delete(Context context)**

  Deletes the dictionary. Returns **true** on success; returns **false** on failure.

  Parameters

  context - The Android context that is making this call.

Example

```
MDXDictionary dict = MDXDictionary.find(getContext(), "app-settings");
```

```
if( dict != null )
    {

        dict.delete( getContext() );
     }
```

Notes and considerations

- Constructors will throw an **IllegalArgumentException** when bad parameters are passed in.
- The **create()** operation will never return null. If the encryption policy is enabled, the user is responsible for ensuring it is unlocked before **create()** is called.
- The **append()** operation can fail if a stored object that can be parsed or serialized is not a known Java or Android datatype. Secure Hub is unable to unmarshal the dictionary because the class is not known internally to Secure Hub.
- The **append()** operation will append its bundle to an existing dictionary bundle. If the stored bundle is different than the bundle in the dictionary, the local bundle will not reflect the state of the bundle stored. A **find()** operation or a **create()** operation is necessary to query the state of previously stored bundle.

Developing iOS apps

March 12, 2019

You can use the MDX API to enable your mobile apps for Citrix Endpoint Management. This article describes how to integrate the MDX App SDK into your app library and the steps required to test, certify, and publish your apps.

How to use the MDX App SDK

Here are some examples of how you might use the APIs.

- Place restrictions on apps

  You can control when your app allows access to certain features or actions based on whether API calls indicate that the app is managed or wrapped. For example, if an app isn't managed or wrapped, you might allow a user access to all features and actions. If an app is wrapped but not managed, you might then restrict certain features or actions. If an app is wrapped and managed, you might put extra restrictions on the app.

- Perform actions based on Citrix Endpoint Management policy settings

  Suppose that you want to display a notification to users if a Citrix Endpoint Management administrator sets the Require Wi-Fi policy to
  On. That means that the app is allowed to run only from inside the network of your organization. You can use the API to look up the policy setting and then base your code changes on the policy value.

- Perform actions based on custom policies

  You can use the APIs to read custom policies in your apps. For example, suppose that you want to enable Citrix Endpoint Management administrators to display a notification in the app. To do that, create a custom policy that is empty or contains a system message that is supplied by an administrator in the Citrix Endpoint Management console. If your app is managed, it can detect when the Citrix Endpoint Management administrator changes the policy value. If the policy value contains a message, your app displays the notification.

For API definitions, see API for iOS.

Integrating the SDK into your app library

To add the MDX App SDK to your iOS apps, link the SDK framework into your app as described in this section. The MDX App SDK for iOS, based on Objective-C, is a collection of header files and a static library.

1. If you haven't already installed the latest MDX Toolkit, do so now.

   a) Log on to the Citrix Endpoint Management downloads page.

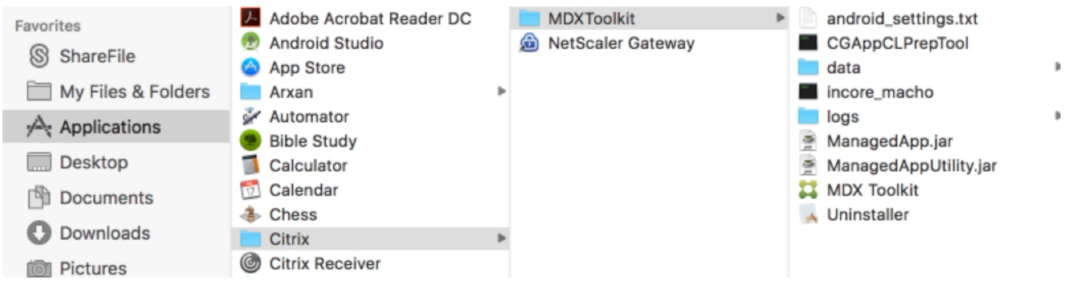
   b) Expand **XenMobile Apps and MDX Toolkit**.

   c) Locate the MDX Toolkit version you want to install and click the link to begin the download.

   d) Open MDXToolkit.mpkg with the macOS Finder tool on macOS 10.9.4 or later and Xcode 7 or later.

   For Xcode 8 and above, a known issue exists where the project file must be cleaned before pushing the app to the device.

   The installation path is Applications/Citrix/MDXToolkit.



The MDX App SDK files are in Applications/Citrix/MDXToolkit/data/MDXSDK.

After installing the MDX Toolkit on your computer, integrate MDX Framework into your Xcode project.

2. Add the data/MDXSDK folder to the Apple Xcode project. To do so, you can drag that folder to the Xcode project.
3. Revise a line of code in the pre-compiled header file in the app project to import MDX.h from MDX.framework as shown in the following example.

```
#ifdef__OBJC__
_
//import MDX extensions
#import <AVFoundation/AVFoundation.h>
#import <SystemConfiguration/SCNetworkReachability.h>
#import <MDX/MDX.h>
#endif
```

If you only include the Network Only version of the MDX framework then you should replace

```
#import <MDX/MDX.h.>
```

With

```
#import <MDXNetworkOnly/MDXNetworkOnly.h>
```

If wrapping an application that explicitly issues a call to an API exposed by the MDX SDK framework, the MDX.h and MDXNetworkOnly.h lines are optional.

If an application explicitly issues an MDX SDK API call, it must be linked to and embed the MDX.framework or MDXNetworkOnly.framework binary when the application is built.

A prebuilt 3rd party enterprise application that is wrapped by the MDX Toolkit requires no build modifications since it makes no explicit MDX SDK API calls.

After installing the MDX Toolkit on your computer, integrate MDX Framework into your Xcode project.

4. Drag the data/MDXSDK/MDX.framework (or the data/MDXSDK/MDXNetworkOnly.framework) to the Embedded Binaries section of the General properties panel for the application workspace. By doing so, you add that dynamic framework to the frameworks included in the application bundle installed with the application. In addition, the framework is automatically added to the list of frameworks that are linked to the application.

You should only add one MDX framework.

1. Drag the data/MDXSDK/CitrixLogger.framework to the Embedded Binaries section of the General properties panel for the application workspace.

2. Add a run script to remove architectures from embedded frameworks that do not appear in the list of Xcode valid architectures. This addresses the Apple requirement that embedded frameworks can not contain iOS Simulator architectures for Apple Store application builds. This script will automatically handle all build targets both non-Apple Store and Apple Store type builds.

```
echo "Strip unnecessary archs from Embedded Frameworks"
cd "${
 BUILT_PRODUCTS_DIR }
 /${
 FRAMEWORKS_FOLDER_PATH }
 "
for file in $(find . -type f -perm +111);
do
    if ! [[ "$(file "$file")" == *"dynamically linked shared library"* ]];
    then
        continue
    fi
    # Get architectures for current file
    archs="$(lipo -info "${
 file }
 " | rev | cut -d ':' -f1 | rev)"
    # Strip any archs from frameworks not valid for current app build
    for arch in $archs;
    do
        if ! [[ "${
 VALID_ARCHS }
 " == *"$arch"* ]];
        then
            lipo -remove "$arch" -output "$file" "$file" || exit 1
        fi
    done
done
```

3. Add a run script to add the SDKPrep command line.

   - Select your project in Xcode, and then select the **Build Phases** tab. Click the plus (+) icon in the upper-left corner, and then click **New Run Script Phase**.
   - Open the new Run Script, and then type the following text into the **Script** field. Be sure to change the PACKAGEID, APPTYPE, STOREURL, and POLICYFILE variables to values that are applicable to your app. The PACKAGEID is a unique identifier for your app, typically a

UUID. This is not mandatory, as the MDX Toolkit generates a unique packageID whenever the application is built. If you supply a packageID, ensure that it is unique for every new app version you wrap using this command.

- If this app is an Enterprise app, use the parameter -*Apptype Enterprise* which is the default value. For ISV apps, you can use the Premium or General values.

> Note:
>
> The supported keywords for APPTYPE are Enterprise, Premium, and General.

```
    export PACKAGEID="your-project-PackageID"
    export APPTYPE="keyword"
    export STOREURL="http://your-store-URL"
    export DATE=`date +%Y-%m-%d_%H-%M-%S`
    export POLICYFILE=${
SRCROOT }
/${
EXECUTABLE_NAME }
/${
EXECUTABLE_NAME }
_policy_metadata.xml
    /Applications/Citrix/MDXToolkit/CGAppCLPrepTool SdkPrep -in "$
        {
CODESIGNING_FOLDER_PATH }
" -out "/Users/<UserName>/Downloads/${
EXECUTABLE_NAME }
_${
DATE }
.mdx" -storeUrl "${
STOREURL }
" -appIdPrefix "ABCDEFGH" -packageId "${
PACKAGEID }
" -policyXML "${
POLICYFILE }
" -appType "${
APPTYPE }
" -entitlements "${
CODE_SIGN_ENTITLEMENTS }
"
```

Example:

```
    export PACKAGEID="a96d6ed5-6632-4739-b9b6-9ad9d5600732"
    export APPTYPE="Enterprise
    export STOREURL="http://example.com/12345"
    export DATE=`date +%Y-%m-%d_%H-%M-%S`
    export POLICYFILE=${
 SRCROOT }
 /${
 EXECUTABLE_NAME }
 /${
 EXECUTABLE_NAME }
 _policy_metadata.xml
    /Applications/Citrix/MDXToolkit/CGAppCLPrepTool SdkPrep -in "$
        {
 CODESIGNING_FOLDER_PATH }
 " -out "/Users/<UserName>/Downloads/${
 EXECUTABLE_NAME }
 _${
 DATE }
 .mdx" -storeUrl "${
 STOREURL }
 " -appIdPrefix "ABCDEFGH" -packageId "${
 PACKAGEID }
 " -policyXML "${
 POLICYFILE }
 " -appType "${
 APPTYPE }
 " -entitlements "${
 CODE_SIGN_ENTITLEMENTS }
 "
```

Parameters	Description
-in *file name*	Path to the .app file generated by Xcode. The MDX Toolkit embeds MDX-specific resources into this file.
-out *file name*	Destination path for the .mdx file. Use this file to publish the app on the Citrix Endpoint Management server.
-storeURl *URL*	App store URL for the app, embedded into the .mdx file. Cannot use this parameter with -StoreURL.
-appType *keyword*	Keywords are “Enterprise,” “Premium,” and “General.”

Parameters	Description
-packageId *UUID*	The unique package ID for this app, typically a UUID. Not mandatory, as the MDX Toolkit generates a unique packageID whenever the application is built. If you supply a packageId, ensure that it is unique for every new app version you wrap using this command. A unique ID is associated with every provisioning profile. If you open the provisioning profile (.mobileprovision) in a text editor, you see the below XML tag with the UUID. `<key>UUID</key> <string>4e38fb18-88b0-4806-acfa-e08bf38ec48d</string>`
-policyXML *file name*	Path to the MDX policy template file for your app.
-entitlements *file name*	Mandatory (introduced in version 10.3.10). Path to the entitlements file for the app. The MDX Toolkit adds a keychain access group entry for com.citrix.mdx to this file. It is needed for your app to share secrets with other MDX apps signed with the same certificate, using the iOS keychain.
-appIdPrefix *prefix*	Application Identifier Prefix - the Team ID associated with your Apple Developer account.

4. Compile your project and generate the app binaries.

   - Build your app in Xcode, verifying that it builds correctly.
   - Archive your app by selecting **Product > Archive**.
   - The Xcode Organizer opens automatically after your app is archived.
   - Select your archived build in Organizer and then click Export.
   - Select the applicable export method and then click Next.

   Follow the prompts to export your app to an IPA file.

5. Compile and archive the project to generate the app bundle that contains the embedded MDX Framework which is the .ipa package. Xcode generates a corresponding MDX file that you upload to the Citrix Endpoint Management server. After the Xcode build and archive steps creates the IPA bundle, run the SetInfo command on the MDX file. Also run the command -embedBundle

option to insert the final IPA file into the MDX file. After that, you can upload the app to Citrix Endpoint Management.

```
/Applications/Citrix/MDXToolkit/CGAppCLPrepTool SdkPrep -in "${
 CODESIGNING_FOLDER_PATH }
 " -out "/Users/<UserName>/Downloads/${
 EXECUTABLE_NAME }
 _${
 DATE }
 .mdx" " -embedBundle "/Users/deva/Desktop/{
 EXECUTABLE_NAME }
 .ipa"
<!--NeedCopy-->
```

6. If you have configured the app for distribution via the iTunes Connect website, you can also submit directly to the app store or TestFlight.

Considerations for upgrading apps

The Citrix Endpoint Management software can change significantly between releases. To take advantage of the latest features and bug fixes, you must use the latest version of the MDX Toolkit to wrap your app. Be sure to wrap your original .ipa or .apk file, not the modified file previously generated by the MDX Toolkit.

Be sure to use the corresponding version of the MDX App SDK.

Best practices for iOS apps

June 2, 2020

When developing iOS apps, use these best practices to improve compatibility between Citrix Endpoint Management and mobile apps for iOS devices.

MDX App SDK Framework and wrapping

If your app uses the MDX App SDK Framework, then you must use the matching MDX Toolkit version for wrapping. A version mismatch between these two components might cause improper operation.

To prevent such a mismatch, wrap the app as an ISV app and specify an app mode of Premium or General. That lets you deliver a pre-wrapped app. As a result, your customer doesn’t need to wrap the app, thus avoiding use of a mismatched MDX Toolkit. For details about ISV wrapping, see Wrapping iOS mobile apps.

Use explicit app IDs

If your iOS Developer Enterprise account does not support wildcard App IDs, be sure to create an explicit App ID for each app you plan to wrap with the MDX Toolkit. Also, create a provisioning profile for each App ID.

Don't block the main thread

Don't use blocking code when running on the main thread. This is an Apple guideline, but it is even more crucial with Citrix Endpoint Management. Some actions can take more time in a managed app or can block further thread execution. File, database, and network operations are examples of operations that might block the currently running thread and should be avoided on the main thread.

Write robust code

In particular, you should write apps following the best practices as documented in the Apple programming guides, such as the Apple Application Programming Guide.

Use only Apple published interfaces

Check return values from all API calls and handle any exceptions that may occur as a side effect of an API call. This effort ensures a graceful error recovery or graceful termination of the app. While this is a common programming best practice, it is especially important for managed apps.

Various APIs that you'd expect to work fail if the underlying functionality has been blocked due to Citrix Endpoint Management policies. Examples would include any of the capabilities described earlier:

- Networking APIs fail as if there is no network available.
- Sensor APIs, such as GPS and camera, return null or throw an exception.

The following Objective-C runtime selectors return nil if the underlying functionality has been blocked due to Citrix Endpoint Management policies and so should be handled accordingly.

Object class: AVCaptureDevice

- **Selector name:** devicesWithMediaType:

Object class: MFMailComposeViewController

- **Selector name:** init:

Object class: MFMessageComposeViewController

- **Selector name:** initWithNibName:bundle:

Object class: NSFileManager

- **Selector name:** URLForUbiquityContainerIdentifier:

Object class: NSUbiquitousKeyValueStore

- **Selector name:** defaultStore:

Object class: PHPhotoLibrary

- **Selector name:** sharedPhotoLibrary:

Object class: UIImagePickerController

- **Selector name:** availableCaptureModesForCameraDevice:

Object class: UIPasteboard

- **Selector name:**

  dataForPasteboardType:

  valueForPasteboardType:

  items:

  dataForPasteboardType:inItemSet:

  valuesForPasteboardType:inItemSet:

Object class: UIPopoverController

- **Selector name:** initWithContentViewController:

Object class: UINavigationController

- **Selector name:**

  ctxInitWithRootViewController:

  ctxPopToViewController:animated:

Redirect runtime interfaces

Citrix Endpoint Management provides UI Pin Prompt interaction so you don't have to do it in your app.

To ensure Citrix Endpoint Management readiness, we suggest that you don't redirect or substitute Objective-C runtime selectors. The reason is that Citrix Endpoint Management swizzles the underlying methods of several object class selectors to control or modify the runtime behavior of an app. The following table lists the Objective-C class selectors that Citrix Endpoint Management redirects:

Object Class Name: NSURLProtectionSpace

- **Selector name:** serverTrust

Object Class Name: NSURLAuthenticationChallenge

- **Selector name:** sender

Object Class Name: NSURLConnection

- **Selector name:**

  sendSynchronousRequest:returningResponse:error:

  initWithRequest:delegate:startImmediately:

  initWithRequest:delegate:

  connectionWithRequest:delegate:

Object Class Name: NSURLConnectionDelegate

- **Selector name:**

  connection:canAuthenticateAgainstProtectionSpace:

  connection:didReceiveAuthenticationChallenge:

  connection:willSendRequestForAuthenticationChallenge:

Object Class Name: NSURLSessionConfiguration

- **Selector name:**

  defaultSessionConfiguration

  ephemeralSessionConfiguration

Object Class Name: ALAssetsLibrary

- **Selector name:** authorizationStatus

Object Class Name: AVAudioRecorder

- **Selector name:**

  record

  prepareToRecord

  recordForDuration:

  recordAtTime:

  recordAtTime:ForDuration:

Object Class Name: AVAudioSession

- **Selector name:** recordPermission

Object Class Name: AVCaptureDevice

- **Selector name:**

  devices

  devicesWithMediaType:

Object Class Name: AVAsset

- **Selector name:** assetWithURL:

Object Class Name: AVURLAsset

- **Selector name:**

  initWithURL:options:

  URLAssetWithURL:options:

Object Class Name: AVPlayerItem

- **Selector name:**

  playerItemWithAsset:

  initWithURL:

  playerItemWithURL:

Object Class Name: AVPlayer

- **Selector name:**

  playerWithPlayerItem:

  initWithPlayerItem:

  initWithURL:

Object Class Name: CLLocationManager

- **Selector name:** startUpdatingLocation

Object Class Name: UIScrollView

- **Selector name:** setContentOffset:

Object Class Name: MFMailComposeViewController

- **Selector name:**

  canSendMail

  init

Object Class Name: MFMessageComposeViewController

- **Selector name:**

  canSendText

  initWithNibName:bundle:

Object Class Name: NSFileManager

- **Selector name:** URLForUbiquityContainerIdentifier:

Object Class Name: NSUbiquitousKeyValueStore

- **Selector name:** defaultStore

Object Class Name: PHPhotoLibrary

- **Selector name:** authorizationStatus

Object Class Name: QLPreviewController

- **Selector name:**

  setDataSource:

  canPreviewItem:

Object Class Name: QLPreviewControllerDataSource

- **Selector name:**

  numberOfPreviewItemsInPreviewController:

  previewController:previewItemAtIndex:

Object Class Name: SLComposeViewController

- **Selector name:** isAvailableForServiceType:

Object Class Name: UIActivityViewController

- **Selector name:**

  initWithActivityItems:applicationActivities:

  setExcludedActivityTypes:

Object Class Name: UIApplication

- **Selector name:**

  openURL:

  canOpenURL:

  setApplicationIconBadgeNumber:

Object Class Name: UIDocument

- **Selector name:**

  closeWithCompletionHandler:

  contentsForType:error:

Object Class Name: UIDocumentInteractionController

- **Selector name:**

  interactionControllerWithURL:

  setURL:

  setDelegate:

  presentPreviewAnimated:

  presentOpenInMenuFromBarButtonItem:animated:

  presentOpenInMenuFromRect:inView:animated:

  presentOptionsMenuFromBarButtonItem:animated:

  presentOptionsMenuFromRect:inView:animated:

Object Class Name: UIDocumentMenuViewController

- **Selector name:** initWithDocumentTypes:inMode:

Object Class Name: UIImage

- **Selector name:** imageNamed:

Object Class Name: UIImagePickerController

- **Selector name:** setSourceType:

  takePicture

  startVideoCapture

  isSourceTypeAvailable:

  isCameraDeviceAvailable:

  isFlashAvailableForCameraDevice:

  availableCaptureModesForCameraDevice:

  setMediaTypes

Object Class Name: UINavigationController

- **Selector name:**

  ctxInitWithRootViewController:

ctxPushViewController:animated:

ctxPopToViewController:animated:

Object Class Name: UIPasteboard

- **Selector name:**

generalPasteboard

pasteboardWithName:create:

pasteboardWithUniqueName

setValue:forPasteboardType:

setData:forPasteboardType:

setItems:

addItems:

dataForPasteboardType:

valueForPasteboardType:

numberOfItems

pasteboardTypes

pasteboardTypesForItemSet:

containsPasteboardTypes:

containsPasteboardTypes:inItemSet:

items

itemSetWithPasteboardTypes:

dataForPasteboardType:inItemSet:

valuesForPasteboardType:inItemSet:

string

strings

URL

URLs

image

images

color

colors

Object Class Name: UIPopoverController

- **Selector name:** initWithContentViewController

Object Class Name: UIPrintInteractionController

- **Selector name:**

  isPrintingAvailable

  presentAnimated:completionHandler:

  presentFromBarButtonItem:animated:completionHandler:

  presentFromRect:inView:animated:completionHandler:

Object Class Name: UIViewController

- **Selector name:** presentViewController:animated:completion:

Object Class Name: UIWebView

- **Selector name:**

  loadRequest:

  setDelegate:

  UIWebViewDelegate

  webView:shouldStartLoadWithRequest:navigationType:

  webViewDidStartLoad:

  webViewDidFinishLoad:

  webView:didFailLoadWithError:

Object Class Name: UIWindow

- **Selector name:** makeKeyAndVisible

Object Class Name: UIApplicationDelegate

- **Selector name:**

  applicationDidFinishLaunching:

  application:didFinishLaunchingWithOptions:

  application:willFinishLaunchingWithOptions:

  applicationWillResignActive:

  applicationDidEnterBackground:

  applicationWillEnterBackground:

applicationDidBecomeActive:

applicationWillTerminate:

application:openURL:sourceApplication:annotation:

application:handleOpenURL:

applicationProtectedDataWillBecomeUnavailable:

applicationProtectedDataDidBecomeAvailable:

application:performFetchWithCompletionHandler:

application:handleEventsForBackgroundURLSession:completionHandler:

application:didReceiveLocalNotification:

application:didReceiveRemoteNotification:

application:didReceiveRemoteNotification:fetchCompletionHandler:

application:didRegisterForRemoteNotificationsWithDeviceToken:

application:didFailToRegisterForRemoteNotificationsWithError:

applicationSignificantTimeChange:

application:shouldAllowExtensionPointIdentifier:

Object Class Name: QLPreviewController

- **Selector name:** allocWithZone:

Ensure data encryption compatibility

One of the primary features of MDX is that all persisted data is transparently encrypted. You don't need to modify your app to gain this functionality and, in fact, you can't directly avoid it. The Citrix Endpoint Management admin can disable encryption either selectively or entirely, but not the app.

This is one of the more heavyweight aspects of MDX and requires an understanding of the following points:

- File encryption is present for all native code that runs in managed processes.

  The file data encryption implementation supports all native code and not just code for apps using the Apple frameworks and the Apple Objective-C runtime. Any file data encryption implemented within and solely for the Objective-C runtime can be easily subverted.

- Some framework APIs, such as AVPlayer class, UIWebView class, and QLPreviewController, are implemented by iOS service processes in a different execution context than the user's managed app process.

These service processes cannot decrypt MDX encrypted file data. Therefore, the managed app must provide the service process with a temporary unencrypted copy of the data. The copy is deleted by the managed app after 5 seconds. It is important that you are aware of the limitation when using these classes. The reason is that we lose containment control of the data provided to these classes due to Apple implementation of the specific classes.

- Memory mapping is problematic for Citrix Endpoint Management encryption since it relies on app calling file I/O system call interfaces.

  After a file is memory mapped, the I/O requests for the file are managed outside of the context of the user app bypassing Citrix Endpoint Management encryption. All POSIX mmap(2) calls by a managed app are mapped as MAP_PRIVATE and MAP_ANON and not associated with any file description. An attempt is made to read in all the mapped data during the mmap call if a file description is specified to fault in all the data since any subsequent paging in of data by the operating system results in reading encrypted data without it being decrypted by Citrix Endpoint Management. This technique has been successful in all apps tested with Citrix Endpoint Management since the amount of data that is memory mapped is small with no memory page reclaims happening within the app.
- Encryption adds measurable overhead. Developers should optimize disk I/O to prevent performance degradation. For example, if you're repeatedly reading and writing the same information, you might want to implement an app level cache.
- Citrix Endpoint Management only encrypts instances of the Apple libsqlite.dylib . If the application directly links with and/or embeds an private version of the libsqlite.dylib , the databases instances of that private library are not encrypted by Citrix Endpoint Management.
- Apple SQLite databases are encrypted by Citrix Endpoint Management using the SQLite Virtual File System layer.

  Performance can be an issue. The standard database cache size is 2000 pages or 8 megabytes. If your database is large, a developer may need to specify SQLite pragma to increase the database cache size. In Objective-C Core Data Framework, the SQLite pragma can be added as an option dictionary when adding the Persistent Store object to the Persistent Store Controller object.
- SQLite WAL mode is not supported since the library is relinked to file I/O interfaces and internally uses memory mapping extensively.
- NSURLCache DiskCache is implemented by iOS using a SQLite database. Citrix Endpoint Management disables the disk cache associated since this database is referenced by unmanaged iOS service processes.
- The following is a list of the hardcoded excluded file path name patterns:
  - .plist: Excluded due to access by iOS system processes outside process context.

  - .app: Legacy substring in the Application Bundle name. This substring is deprecated because an explicit Application Bundle path is now excluded.
  - .db: A file with this suffix is not encrypted if the file is not a SQLite database.
  - /System/Library: File paths that exist within the app bundle sandbox directory and file paths outside the app data sandbox cannot be encrypted. On iOS, the installed app is read only and is in a different directory than the app data files that the app produces and stores when it is run.
  - Library/Preferences: Files are accessed by iOS directly. Normally only .plist files are present in this directory path.
  - /com.apple.opengl/: iOS accesses the files directly.
  - csdk.db: Legacy Citrix SSLSDK SQLite database
  - /Library/csdk.sql: Citrix SSLSDK SQLite database
  - CtxLog_: Citrix log file name prefix
  - CitrixMAM.config: MDX internal file name
  - CitrixMAM.traceLog: Legacy MDX internal file name
  - CtxMAM.log: MDX internal file name
  - data.999: MDX internal file name
  - CTXWrapperPersistentData: MDX internal file name
  - /Documents/CitrixLogs: MDX log directory
  - /Document/CitrixLogs.zip: Compressed MDX log directory name
  - Any file in the app Bundle directory path: Read-only directory of app files

- Citrix Endpoint Management substitutes an instance of the private Citrix Endpoint Management SecureViewController class for instances of the Apple Objective-C QLPreviewController object class at runtime. Citrix Endpoint Management SecureViewController class is derived from the Apple Objective-C UIWebView object class. The QLPreviewController object class natively supports a few file formats which the UIWebView object class doesn't natively support, such as the audio and PDF types.

- For best performance, file I/O requests should be issued to file offsets which are a multiple of 4096 bytes and should be issued for a length which is also a multiple of 4096 bytes.

- The O_NONBLOCK file mode flag is not supported by Citrix Endpoint Management encryption. This file mode flag is removed from the list of modes when processed by Citrix Endpoint Management.

Encryption user entropy

One Citrix Endpoint Management option for encryption requires the end user to enter a PIN before the encryption key can be generated. This option is called user entropy. It can cause a particular issue for apps.

Specifically, no file or database access can be performed until the user enters a PIN. If such an I/O operation is present in a location that runs before the PIN UI can be displayed, it will always fail.

To ensure that this issue isn't present in your app, test with user entropy enabled. The Citrix Endpoint Management client property, Encrypt secrets using Passcode, adds user entropy. You configure that client property, which is disabled by default, in the Citrix Endpoint Management console under **Configure > Settings > More > Client Properties**.

Data containment compatibility

- Any remote view controllers will not have security containment (for example, data encryption; copy, cut, and paste policy blocking; and so on) because a remote view controller runs in a different process context than the MDX-managed app.
- The Copy action is only action supported from UIResponder. Other actions, such as Cut and Delete, are not supported.
- AirDrop is only intercepted at the UI level, not at a lower level.
- MFI and Bluetooth are not intercepted.

Icon file support

MDX wrapping requires the presence of at least one icon that can be used as the home screen icon or app icon. App developers can add their icons to the Asset Catalog, or use the CFBundleIcons or CFBundleIconFiles keys in Info.plist.

The MDX Toolkit picks the first one from the list of known plist locations in Info.plist:

- CFBundleIcons
- CFBundlePrimaryIcon
- CFBundleIconFiles
- UINewsstandIcon
- CFBundleDocumentTypes

If none of those keys is found in Info.plist, the MDX Toolkit will identify one of the following icons in the root folder of the app bundle:

- Icon.png
- Icon-60@2x.png
- Icon-72.png
- Icon-76.png

Networking and micro VPN

MDX currently manages only those networking calls issued directly by an app. Some DNS queries are issued directly by the Apple framework and so are not managed by MDX.

Several Citrix Endpoint Management policy options are available to administrators for networking.

The Network access policy prevents, permits or redirects app network activity.

> Important:
>
> The MDX Toolkit version 18.12.0 release included new policies that combined or replaced older policies.
> The Network Access policy combines Network access, Preferred VPN mode, and Permit VPN mode switching. The Exclusion list policy replaces Split tunnel exclusion list. The micro VPN session required policy replaces Online session required. For details, see What's new in earlier releases.
>
> Tunneled - Web SSO is the name for Secure Browse in the settings. The behavior is the same.

The options are as follows:

- **Use Previous Settings**: Defaults to the values you had set in the earlier policies. If you change this option, you shouldn't revert to **Use Previous Settings**. Also note that changes to the new policies do not take effect until the user upgrades the app to version 18.12.0 or later.
- **Blocked**: Networking APIs used by your app will fail. Per the previous guideline, you should gracefully handle such a failure.
- **Unrestricted**: All network calls go directly and are not tunneled.
- **Tunneled - Full VPN**: All traffic from the managed app tunnels through Citrix Gateway.
- **Tunneled - Web SSO**: The HTTP/HTTPS URL is rewritten. This option allows only the tunneling of HTTP and HTTPS traffic. A significant advantage of **Tunneled - Web SSO** is single sign-on (SSO) for HTTP and HTTPS traffic and also PKINIT authentication. On Android, this option has low setup overhead and is thus the preferred option for web browsing types of operations.
- **Tunneled - Full VPN and Web SSO**: Permits switching between VPN modes automatically as needed. If a network request fails due to an authentication request which cannot be handled in a specific VPN mode, it is retried in an alternate mode.

Limitations

- Users cannot play videos hosted on internal websites in iOS wrapped MDX apps because the videos play in a media player process on the device that MDX does not intercept.
- NSURLSession background download (NSURLSessionConfiguration backgroundSessionConfigurationWithIdentifier) is not supported.
- We block UDP traffic if the Network access policy is set to **Blocked**. We don't tunnel UDP traffic if the Network access policy is set to **Tunneled - Full VPN**.

- MDX-wrapped apps cannot instantiate a socket server that listens for inbound connections. However, MDX-wrapped apps can use a client socket to connect to a server.

Third-party library support

Some app frameworks have compatibility issues with Citrix Endpoint Management:

- Apps developed with Xamarin cross platform development environment are supported. Citrix does not officially declare support for other cross platform development environments due to insufficient examples of use and testing.
- SQLCipher doesn't work with encryption because it uses memory mapping. One solution is to not use SQLCipher. A second solution is to exclude the database file from encryption using an encryption exclusion policy. A Citrix Endpoint Management administrator must configure the policy in the Citrix Endpoint Management console.
- App and third-party libraries which link directly to OpenSSL libcrypto.a and libssl.a libraries can result in a link error due to missing symbols and link errors due to multiple symbol definition.
- Apps requiring support for Apple Push Notification Service needs to follow specific steps which Apple requires.
- Citrix Endpoint Management explicitly sets the SQLite database version to 1 in order to disable Write Ahead Logging (WAL) file and memory mapped file support within SQLite databases. Any attempt to directly access SQLite interfaces in SQLite version 2 or version 3 fails.

API for iOS

March 29, 2021

The XenMobile API for iOS is based on Objective-C. This article summarizes the Citrix Endpoint Management APIs by feature and provides the API definitions.

App management

- isAppManaged

Interaction with Secure Hub

- isMDXAccessManagerInstalled
- logonMdxWithFlag
- isAppLaunchedByWorxHome

MDX policies

- getValueOfPolicy

Shared vault

- getVaultDataFromVault
- saveVaultData
- updateAndSynchronizeVaultItem
- updateAndSynchronizeVaultItems
- deleteVault
- deleteVaultWithError

User data

- managedUserInformation

Class MdxManager

Methods

- **getValueOfPolicy**

  ```
  +(NSString*)getValueOfPolicy:(NSString*)policyName error:(NSError **)
  error;
  ```

  For managed apps, returns the policy value set by Citrix Endpoint Management administrators. For unmanaged Premium apps, returns the policy value set in Applications/Citrix/MDXToolkit/-data/MDXSDK/default_policies.xml. For unmanaged General apps, returns **nil**.

  Parameters:

  policyName - The name of the policy to search for in default_policies.xml.

  Example:

  ```
  +(NSString*)getValueOfPolicy:(NSString*)DisableCamera  error:(NSError
  **)error;
  ```

- **isMDXAccessManagerInstalled**

  ```
  +(BOOL)isMDXAccessManagerInstalled: (NSError **)error;
  ```

  Checks if Secure Hub is installed, which means that MDX control of the app is enabled even if the app isn't managed. Returns **true** if Secure Hub is installed.

- **isAppManaged**

  ```
  +(BOOL)isAppManaged;
  ```

  Checks if the app is currently managed by MDX, which means that the MDX policy bundle is embedded in the app as an XML file. The Citrix Endpoint Management backend infrastructure (key vaults) are queried for data encryption partial keys (secrets), which MDX will use to encrypt app database data (iOS 9 and later). Returns **true** if the app is managed.

- **logonMdxWithFlag**

  `+(BOOL)logonMdxWithFlag:(BOOL)force error:(NSError**)error;`

  Initiates an MDX Logon request with Secure Hub.

- **isAppLaunchedByWorxHome**

  `+(BOOL)isAppLaunchedByWorxHome;`

  Checks whether an inter-application URL request is from Secure Hub or some other app on the device, which is necessary if an app needs to be aware of MDX control communication. On iOS, apps can register for specific URL schemes. A URL scheme is the first part of a URL, up to but not including the colon. If a URL starts with `http://`, the scheme is http.

  MDX-enabled apps and Secure Hub communicate using custom URL schemes. For example, to handle mailto: URLs from other apps, Secure Mail requires the URL scheme ctxmail. To handle http or https URLs from other apps, Secure Web requires the URL scheme ctxmobilebrowser or ctxmobilebrowsers, respectively. For details about the MDX App URL schemes policy and Allowed URLs policy, see MDX policies for iOS apps.

  Returns accurate results when queried anytime or anywhere during or after the following UIApplication delegate event calls:

  - When the app loads from springboard or an **openURL** call:

    ```
    application:willFinishLaunchingWithOptions:

    application:didFinishLaunchingWithOptions:

    applicationDidFinishLaunching:
    ```

  - When the app is activated or re-activated by users from the device springboard

    ```
    applicationDidBecomeActive:
    ```

  > **Important:**
  >
  > You must not query during applicationWillEnterForeground:.

  - When the app is activated or re-activated by an **openURL** call:

    ```
    application:openURL:sourceApplication:annotation:

    application:handleOpenURL:
    ```

- **managedUserInformation**

```
extern __attribute__((visibility ("default")))NSString *const kXenMobileUsername
; +(NSDictionary*)managedUserInformation;
```

Returns a string containing the UserName of an enrolled user running an MDX-managed app, regardless of the user sign-on status. Returns an empty string if the user isn't enrolled, the app isn't managed, or the app isn't wrapped.

Class XenMobileSharedKeychainVault

Methods

- **initWithVaultName**

  ```
  - (instancetype)initWithVaultName:(NSString*)vaultName accessGroup:(
  NSString*)accessGroup;
  ```

  Initializes a Citrix Endpoint Management shared vault.

  Use the shared vault API to share managed content between apps that have the same keychain access group. For example, you can share user certificates through an enrolled app so that apps can obtain a certificate from the secure vault instead of from Secure Hub.

  Parameters:

  vaultName - The name of the Citrix Endpoint Management shared vault.

  accessGroup - The name of the keychain access group. This can be the default MDX access group, named TEAMID_A.appOriginalBundleID, or a keychain access group you will use to share data between apps.

- **Vault Data Type Properties**

  ```
  @property(nonatomic,readonly) BOOL exists;

  @property(nonatomic,readonly) BOOL isAccessible;

  @property(nonatomic,strong) NSMutableDictionary\* vaultData
  <!--NeedCopy-->
  ```

  After you initialize a vault, these vault data type properties are returned:

  exists - Indicates whether the vault with the specified *vaultName* was found.

  isAccessible - Indicates whether the vault is in the specified *accessGroup* and can be accessed.

  vaultData - Is the contents of the shared vault. When you first initialize the vault, *vaultData* is a nil dictionary.

- **getVaultDataFromVault**

  `+ (NSDictionary*)getVaultDataFromVault:(NSString*)vaultName accessGroup:(NSString*)accessGroup error:(NSError *__autoreleasing *)error;`

  Reads data from the Citrix Endpoint Management shared vault. This is one of three ways to read vault data, as follows:

  - Directly use getVaultDataFromVault:accessGroup:error.
  - Create the XenMobileSharedKeychainVault instance and then read the vaultData property.
  - Create the XenMobileSharedKeychainVault instance and then reload vault data using `-(BOOL)loadDataWithError:(NSError *_autoreleasing *)error;` and reading the vaultData property.

  For example code, see the Shared Vault Example in this article.

  Parameters:

  vaultName - The name of the Citrix Endpoint Management shared vault.

  accessGroup - The name of the keychain access group. This can be the default MDX access group, named TEAMID_A.appOriginalBundleID, or a keychain access group you will use to share data between apps.

- **saveVaultData**

  `+ (BOOL)saveVaultData:(NSDictionary*)vaultData toVault:(NSString*)vaultName accessGroup:(NSString*)accessGroup error:(NSError *__autoreleasing *)error;`

  Saves data in the Citrix Endpoint Management shared vault. This is one of three ways to save vault data, as follows:

  - Directly use **saveVaultData:toVault:accessGroup:error:**.
  - Use **updateAndSynchronizeVaultItem:** or **updateAndSynchronizeVaultItems** (described next in this table).
  - Use - **(BOOL)synchronizeWithError:(NSError *_autoreleasing *)error;** by creating the **XenMobileSharedKeychainVault** instance, loading the vault data, modifying the vault data, and then synchronizing the data.

  For example code, see Shared Vault Example in this article.

  Parameters:

  vaultData - The data to save to the Citrix Endpoint Management shared vault. Data stored in the share vault is a dictionary of key/value pairs, such as @{@"username":@";andreo"}.

  vaultName - The name of the Citrix Endpoint Management shared vault.

accessGroup - The name of the keychain access group. This can be the default MDX access group, named TEAMID_A.appOriginalBundleID, or a keychain access group you will use to share data between apps.

- **updateAndSynchronizeVaultItem**

  updateAndSynchronizeVaultItems

  `- (BOOL)updateAndSynchronizeVaultItem:(NSString*)vaultItem withValue:(id)itemValue error:(NSError *__autoreleasing *)error;`

  `- (BOOL)updateAndSynchronizeVaultItems:(NSDictionary*)vaultItems error:(NSError *__autoreleasing *)error;`

  Updates data in the Citrix Endpoint Management shared vault. To use this method, create the **XenMobileSharedKeychainVault** instance and then synchronize it by adding or updating vault data items. For example, if the existing vault entry has {a:123, b:234, c:305} and we use this API with data to update {c:345, d:456}, this API will update the vault data to {a:123, b:234, c:345, d:456}. For example code, see Shared Vault Example in this article.

  See **saveVaultData**, above, for two other ways to save vault data.

  Parameters:

  vaultItem - A single key/value pair, in the form `@{ @";username::@";andreo"}`.

  vaultItems - A list of key/value pairs.

- **deleteVault**

  `+ (BOOL)deleteVault:(NSString*)vaultName accessGroup:(NSString*)accessGroup error:(NSError *__autoreleasing *)error;`

  Deletes the specified shared vault.

  Parameters:

  vaultName - The name of the Citrix Endpoint Management shared vault.

  accessGroup - The name of the keychain access group used by the vault you want to delete.

- deleteVaultWithError

  `-(BOOL)deleteVaultWithError:(NSError *__autoreleasing *)error;`

  Deletes the shared vault returned by the **XenMobileSharedKeychainVault** instance. You must free the object after deleting it with **deleteVaultWithError**.

Shared Vault Example

```
#import "XenMobileSharedKeychainVault.h"

@interface ClassA ()
...
@property(nonatomic,strong) XenMobileSharedKeychainVault*
    XenMobileSharedKeychainVault;
...
@end

@implementation ClassA
...
@synthesize XenMobileSharedKeychainVault =
    _XenMobileSharedKeychainVault;


...
#ifdef USE_CLASS_INSTANCE_METHODS
-(XenMobileSharedKeychainVault*)XenMobileSharedKeychainVault
{

if(_XenMobileSharedKeychainVault==nil) {

_XenMobileSharedKeychainVault = [[XenMobileSharedKeychainVault alloc]
      initWithVaultName:<VAULT_NAME>
      accessGroup:kXenMobileKeychainAccessGroup];
 }

return _XenMobileSharedKeychainVault;
 }

#endif

-(void)read
{

NSError* error=nil;
#ifdef USE_CLASS_INSTANCE_METHODS
NSDictionary* vaultDictionary = nil;
if([self.XenMobileSharedKeychainVault loadDataWithError:&error]) {

vaultDictionary = [self.XenMobileSharedKeychainVault vaultData];
 }

#else
```

```
NSDictionary* vaultDictionary = [XenMobileSharedKeychainVault
      getVaultDataFromVault:<VAULT_NAME>
      accessGroup:kXenMobileKeychainAccessGroup error:&error];
#endif

 }


-(void)save
{

NSError* error=nil;
/// check error handling here...

NSDictionary* dictToSave = @{
 <VAULT_DATA_DICTIONARY_OBJECTS> }
 ;
#ifdef USE_CLASS_INSTANCE_METHODS
#ifdef USE_CLASS_INSTANCE_METHODS_TO_UPDATE
BOOL result = [self.XenMobileSharedKeychainVault
      updateAndSynchronizeVaultItems:dictToSave error:&error];
#else
self.XenMobileSharedKeychainVault.vaultData = [NSMutableDictionary
      dictionaryWithDictionary:dictToSave];
BOOL result = [self.XenMobileSharedKeychainVault synchronizeWithError:&
   error];
#endif
#else
BOOL result = [XenMobileSharedKeychainVault
      saveVaultData:dictToSave toVault:<VAULT_NAME>
      accessGroup:kXenMobileKeychainAccessGroup error:&error];
#endif

 }


-(void)delete
{

NSError* error=nil;
#ifdef USE_CLASS_INSTANCE_METHODS
BOOL result = [self.XenMobileSharedKeychainVault deleteVaultWithError:&
   error];
#else
BOOL result = [XenMobileSharedKeychainVault deleteVault:<VAULT_NAME>
```

```
        accessGroup:kXenMobileKeychainAccessGroup error:&error];
#endif

 }


...

@end
<!--NeedCopy-->
```

Policy defaults and custom policies

January 10, 2019

This article discusses the ways you can work with policies in your wrapped ISV apps.

Change policy defaults for unmanaged premium apps

The MDX App SDK includes the following policy files that specify policy defaults for unmanaged Premium apps only.

- Android: Applications/Citrix/MDXToolkit/data/MDXSDK_Android/ default_sdk_policies.xml
- iOS: Applications/Citrix/MDXToolkit/data/MDXSDK/default_policies.xml

All of the policies in those files are disabled. Any policies not in the file are ignored for unmanaged Premium apps.

You can change the default settings as follows.

1. Make a backup of any default policy files you plan to change, in case you need them later.
2. To change a policy default for ISV apps, use the policy values specified in the MDX Toolkit documentation, in MDX policies for Android apps and MDX policies for iOS apps.
3. Include the default policy file with your app resources when you build the Premium app.

Create custom policies

The policy files in the MDX Toolkit provide full definitions of the policies, including the policy label and help text displayed in the Citrix Endpoint Management console. When you wrap an app, these policies are included with the generated .mdx file. You can add custom policies to these files, which are located in the MDX Toolkit installation folder in Applications/Citrix/MDXToolkit/data.

1. Make a backup of any policy files you plan to change, in case you need them later.

2. To add policies to the policy XML files, use the formats provided in “Policy Formats,” next.
3. When you wrap your app, specify the location of your modified policy XML file by including the -policyxml option with the wrapping command line:

   -policyxml /Applications/Citrix/MDXToolkit/data/policy_metadata.xml

   For details about using the command line to wrap ISV apps, see Enterprise iOS app wrapping using the command line and ISV Android app wrapping with the command line
4. To verify the policy names, descriptions, and values in the Citrix Endpoint Management console, upload your app to Endpoint Management.

Guidelines for adding policies

- Change only the items shown in bold.
- The value of the PolicyName element is the name called from your app.
- The value of the PolicyCategory element is the category name under which the policy will be listed in the Citrix Endpoint Management console. To look up category names, see the CategoryId values in the **<Category>** section of the MDX policy files.
- The value of the PolicyDefault element is the default setting of your policy.
- The POLICY_ID in **<Title res_id="POLICY_ID">** is a unique ID used for the policy. The ID must start with a letter, cannot include spaces, and includes only letters, numbers, or the underscore character.
- The value of the Title element is the policy label thatappears in the Citrix Endpoint Management console.
- The POLICY_DESC_ID in **<Description res_id="POLICY_DESC_ID">** is a unique ID for the policy description. The ID must start with a letter, cannot include spaces, and includes only letters, numbers, or the underscore character.
- The value of the Description element is the policy description thatappears in the Citrix Endpoint Management console.

String

```
<Policy>
   <PolicyName>PolicyName</PolicyName>
   <PolicyType>string</PolicyType>
   <PolicyCategory>Category_ID</PolicyCategory>
   <PolicyDefault>Value</PolicyDefault>
   <PolicyStrings>
      <Title res_id="POLICY_ID">Sample String Policy</Title>
      <Description res_id="POLICY_DESC_ID">
          Please enter the policy value.
```

```
        </Description>
    </PolicyStrings>
</Policy>
<!--NeedCopy-->
```

Boolean

```
<Policy>
    <PolicyName>PolicyName</PolicyName>
    <PolicyType>string</PolicyType>
    <PolicyCategory>Category_ID</PolicyCategory>
    <PolicyDefault>false</PolicyDefault>
    <PolicyStrings>
           <Title res_id="POLICY_ID" >Sample Boolean Policy</Title>
        <BooleanTrueLabel res_id="POLICY_ON">On</BooleanTrueLabel>
        <BooleanFalseLabel res_id="POLICY_OFF">Off</BooleanFalseLabel>
        <Description res_id="POLICY_DESC_ID">
            If On, the app does something.
            If Off, the app does something else.

              Default value is Off.
        </Description>
    </PolicyStrings>
</Policy>
<!--NeedCopy-->
```

Enum

```
<Policy>
    <PolicyName>PolicyName</PolicyName>
    <PolicyType>enum</PolicyType>
          <PolicyEnumValues>
           <PolicyEnumValue>
                   <PolicyEnumValueId>Value1</PolicyEnumValueId>
               <PolicyEnumValueString res_id="ID_1">Yes</
                   PolicyEnumValueString>
               </PolicyEnumValue>
               <PolicyEnumValue>
                   <PolicyEnumValueId>Value2</PolicyEnumValueId>
               <PolicyEnumValueString res_id="ID_2">No</
                   PolicyEnumValueString>
           </PolicyEnumValue>
```

```
            <PolicyEnumValue>
               <PolicyEnumValueId>Value3</PolicyEnumValueId>
               <PolicyEnumValueString res_id="ID_3">Maybe</
                  PolicyEnumValueString>
               </PolicyEnumValue>
           </PolicyEnumValues>
     <PolicyCategory>Category_ID</PolicyCategory>
     <PolicyDefault>Value1</PolicyDefault>
     <PolicyStrings>
           <Title res_id="POLICY_ID" >Sample Enum Policy</Title>
        <Description res_id="POLICY_DESC_ID">
            Sample policy description.

               Default value is Yes.
        </Description>
     </PolicyStrings>
</Policy>
<!--NeedCopy-->
```

Troubleshooting

January 10, 2019

To troubleshoot issues that occur when your apps are running in a Citrix Endpoint Management environment, you must first determine whether the issue occurs when your app is unwrapped or wrapped. If the issue occurs when your app is unwrapped, it is specific to your app. Follow your usual troubleshooting procedures.

If the issue occurs when your app is wrapped

- Review the known issues. See Known issues for the MDX App SDK and Known issues.
- Verify that the version of the MDX App SDK you are using is from the same MDX Toolkit your are using to wrap the app. For iOS, make sure that the correct line is added to your project. This will ensure the framework has been added and the APIs will work. For Android, make sure that the libs for all the devices you are using have been added to the project and the worxsdk.aar is added to the project dependencies. If you experience additional problems in integrating the SDK with your project, please contact Citrix Ready or Citrix Support.
- Determine if the issue is an app wrapping error. Review the MDX Toolkit logs in Applications/Citrix/MDXToolkit/logs.

The log files contain the information and progress from wrapping. Check these logs for error messages and warnings. For details, see Identifying iOS App Wrapping Errors and Identifying Android App Wrapping Errors.

Collect logs from Secure Hub: In Secure Hub, tap Support, tap Need help?, and then tap your app name. Secure Mail then opens to a new message that has a log from the selected app attached. You can add more information about the problem in the message. Please list the steps needed to reproduce the issue and include the log from the failed wrap attempt, and any additional information about the issue.

Other logs from the device might be useful. See Collecting System Logs on iOS Devices and Collecting App Logs from the Command Line.

If you are unable to install a wrapped app on a device

- Verify that you are using a valid keystore for Android apps or a valid provisioning profile and certificate pair for iOS. Be aware of the special considerations for provisioning profiles and certificates in Wrapping iOS mobile apps.

If there is an issue with your Apple certificate key

- Request to reissue the certificate from the the Apple Keychain Access app. This will generate a new private key. Then download the certificate and provisioning profile from the Apple developer website.