MDX Toolkit Documentation
For XenMobile 9.0.2

Revision

Date	Version	Description
09/25/2014	1.7	Update for XM 9.0.2. Adding WorxSDK integration steps and documentation. Minor fixes. Adding clarification for existing iOS cert use. iOS8 info.

Contents

Introduction 4

Enterprise Application Wrapping 5

ISV Application Wrapping 6

Worx SDK 8

The MDX Toolkit 9

Downloading the Toolkit 9

Installing the MDX Toolkit 10

Prerequisites for Application Wrapping 13

Android Prerequisites 13

iOS Prerequisites 17

Wrapping Enterprise Applications 19

Command Line for Android 19

Command Line for iOS 23

Wrapping with the GUI Tool 26

Wrapping ISV Applications 33

Command Line for Android 33

Command Line for iOS 34

Wrapping with the GUI Tool 35

MDX Policies 38

Android Policies 38

Default Policies 38

iOS Policies 39

Default Policies 39

Worx SDK 40

Worx SDK for Android 41

Adding the SDK to your Application 41

APIs definitions 42

isManaged 42

isWrapped 42

getPoliciesXML 42

getPolicyValue 42

setPolicyChangeMessenger 43

Libraries 43

Publishing Steps 44

Worx SDK for iOS 45

Publishing Steps 45

Public APIs 47

Libraries 48

Creating Custom Policies 49

String 49

Boolean 49

Enum 49

FAQ 51

Troubleshooting 52

Collecting wrapping logs 52

Common issues and steps to take if they occur 52

Known Issues/Limitations 53

Appendix 54

Importing Certificate and Profile onto new machine 54

Export Existing Cert 54

Import Existing Certificate 56

Creating an Apple Enterprise Provisioning Profile 59

Creating an Apple Developer Provisioning Profile 65

Resources 72

Introduction

The MDX Toolkit is used to prepare iOS and Android applications for deployment with Citrix XenMobile. This preparation process is called application wrapping. The toolkit also surfaces the Worx App SDK for third-party developers to consume and leverage the existing Citrix application management framework within their respective app. This introductory section explains how both processes work. This document details how to install and use the MDX Toolkit as well as how to consume and use the Worx App SDK.

The MDX technology used by Citrix XenMobile adds many enterprise features to existing in-house and third-party mobile apps. Examples of these features include provisioning, custom authentication requirements, per-application revocation, data containment policies, data encryption, and per-application virtual private networking.

Application wrapping performs three main tasks. First, Citrix code is injected into the existing application. This code implements the features described above, and this task outputs a new application file. Next, the new application file is signed with a security certificate. Finally, an MDX file is created. This file contains policy information and other settings. In some situations, the signed application file is also directly contained in the MDX file.

Application wrapping can be done by two distinct sets of users:

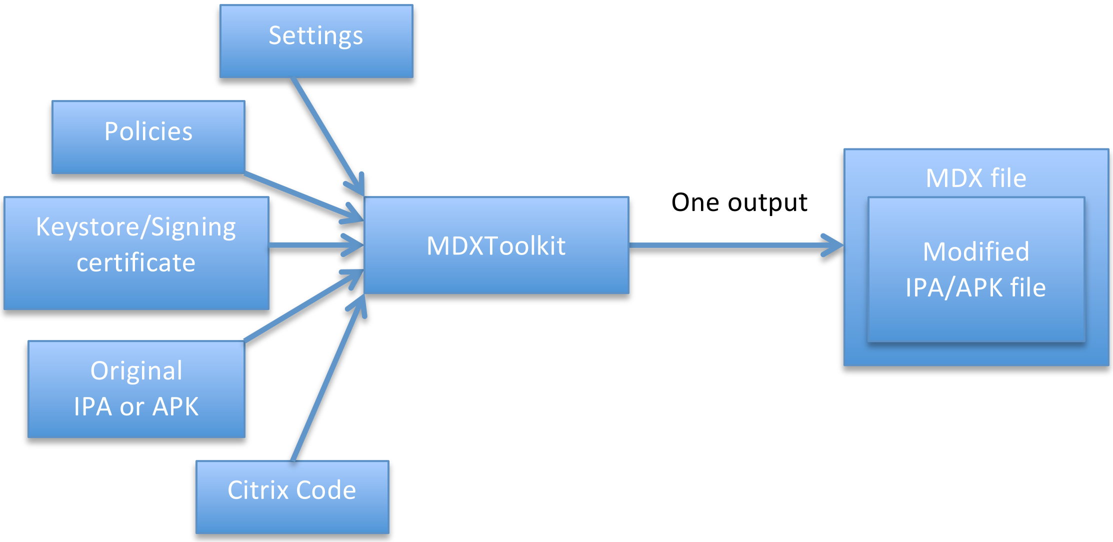
- **Enterprise administrators** wrap custom (in-house) applications and some third-party applications. The administrators upload the wrapped applications on their XenMobile deployment.
- **Independent Software Vendors (ISVs)** wrap applications that they develop and upload the wrapped applications to an app store. By wrapping the applications before release, the ISV simplifies the enterprise administrator's job. As an additional benefit, the ISV can list their wrapped applications in the Citrix Worx App Gallery. This can be used in conjunction with the Citrix Worx SDK. See the Worx SDK section for more.

There are only minor differences between these two use cases, but the remaining documentation will discuss each case separately when necessary.

Enterprise Application Wrapping

For enterprise application wrapping, the administrator begins with an iOS application (.ipa) or an Android application (.apk). Third-party applications should be acquired directly from the application vendor. In particular, iOS applications downloaded from the Apple store are encrypted and cannot be wrapped.

When the MDX Toolkit wraps the application, it uses various inputs as shown in the diagram below. These inputs are merged, resulting in the creation of an .mdx file. The .mdx file contains the modified application along with various settings and policies. The administrator can then upload the .mdx file to XenMobile for deployment to users.



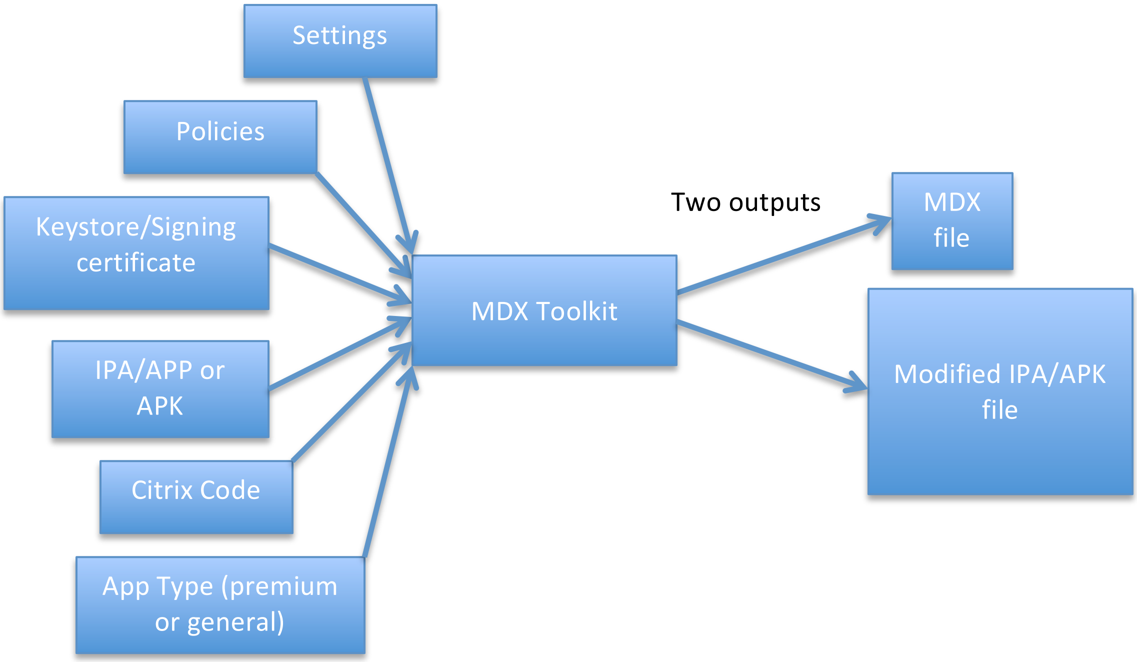
ISV Application Wrapping

Often in conjunction with the Citrix Ready program, an ISV will use the MDX Toolkit to wrap an application before releasing it to customers. Instead of creating a .mdx file with an embedded application file, these two files are output separately. The ISV can deliver the application file through normal app stores, by hosting it themselves, or by distributing it to their enterprise customers. The ISV can then deliver the .mdx file directly to their customer or through the Citrix Worx App Gallery.

If the application is hosted on a normal app store, users who don't have Citrix XenMobile will still be able to download and run the application normally in an unmanaged mode. If an unmanaged user later installs the XenMobile Worx Home application and also attaches to an enterprise that deploys the ISV application, then the ISV application may become managed. There are two wrapping modes that control this behavior:

- In the *general* wrapping mode, all Citrix policies will be ignored for the unmanaged user. If Worx Home is installed and opened, the user will be asked if the application should be managed by the company.
- In the *premium* wrapping mode, some Citrix policies will be enforced even in the unmanaged case. If the user later becomes managed by an enterprise, then the application will automatically become managed.

The illustration below shows the inputs and outputs of ISV application wrapping.



There are a few key differences between ISV application wrapping and enterprise application wrapping:

- The ISV must specify a wrapping mode of general or premium.
- As mentioned earlier, with ISV wrapping the application is not embedded in the .MDX file.
- iOS applications may be supplied in either .ipa or .app format.
- After uploading their wrapped application to an app store, the ISV can use the MDX Toolkit to embed the app store URL into the .mdx file.

More info about ISV application wrapping and verification can be found here.

Worx SDK

The Citrix Worx App SDK exposes a set of APIs to developers to use within their applications. The SDK offers a full range of capabilities using the Citrix MDX app container technology (Worx technology). These APIs give the app developer knowledge of what is going on within their Worx-enabled app and allow them to add custom functionality – such as security and policy enforcement, and so on.

For iOS apps, developers can embed the Worx App SDK Framework into their app with a single line of code.

For Android, developers simply need to import/copy the libraries into their app to use them.

More detailed information can be found in the Worx SDK section later in this document.

The MDX Toolkit

The MDX Toolkit contains three tools for wrapping mobile applications as well as an SDK for use by developers. It also contains support libraries for these three tools. These tools are:

- A Mac OS GUI tool that can wrap both iOS and Android applications
- A Mac OS command-line tool that wraps iOS applications
- A Java command-line tool that wraps Android applications

The installer for these tools must run on Mac OS. The minimum OS version supported by the Toolkit is Mac OS X 10.9. Customers who only need to wrap Android apps may do so on a Windows computer using the Java command line tool. See the FAQ for more details on this scenario.

Before running the toolkit, several prerequisite items must be installed first. The prerequisites needed for wrapping are different for iOS applications and for Android applications. If you only need to wrap applications for one platform, you only need to satisfy the prerequisites for that platform.

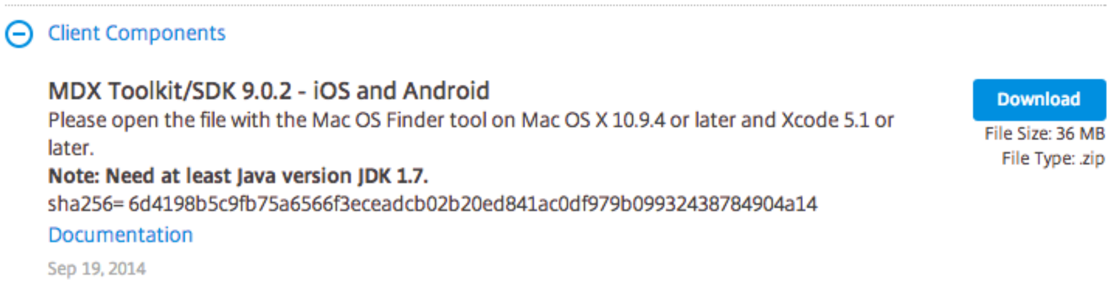
The following sections discuss the MDX Toolkit download, the MDX Toolkit installation, prerequisites for Android, and prerequisites for iOS.

Downloading the Toolkit

Start by downloading the MDX Toolkit version which matches your XenMobile deployment from the Citrix website.

At the time of update, the latest version is XenMobile 9.0.2. This build can be downloaded from the following location:

https://www.citrix.com/downloads/xenmobile/product-software/xenmobile-90-enterprise-edition.html



Client Components

MDX Toolkit/SDK 9.0.2 - iOS and Android

Please open the file with the Mac OS Finder tool on Mac OS X 10.9.4 or later and Xcode 5.1 or later.

Note: Need at least Java version JDK 1.7.

sha256= 6d4198b5c9fb75a6566f3eceadcb02b20ed841ac0df979b09932438784904a14

Documentation

Sep 19, 2014

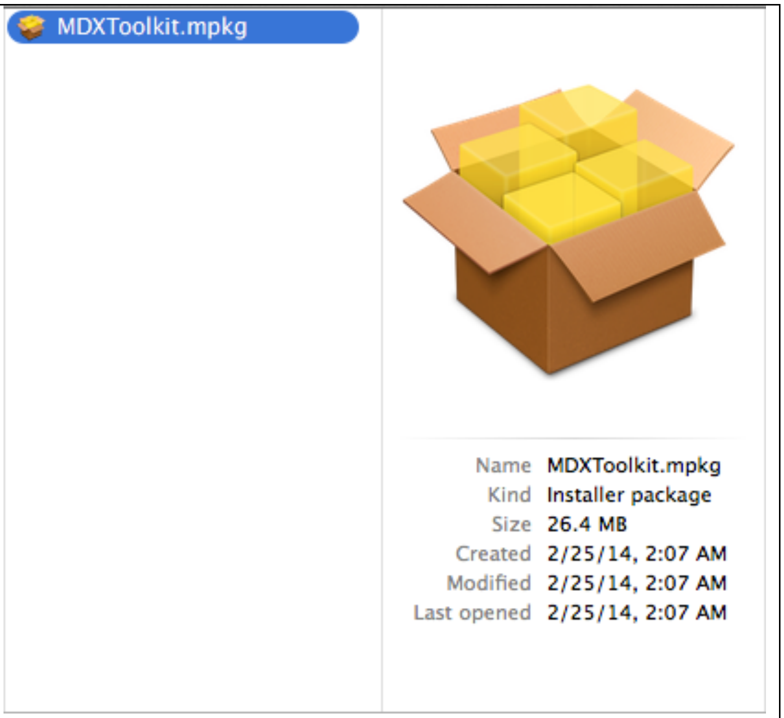
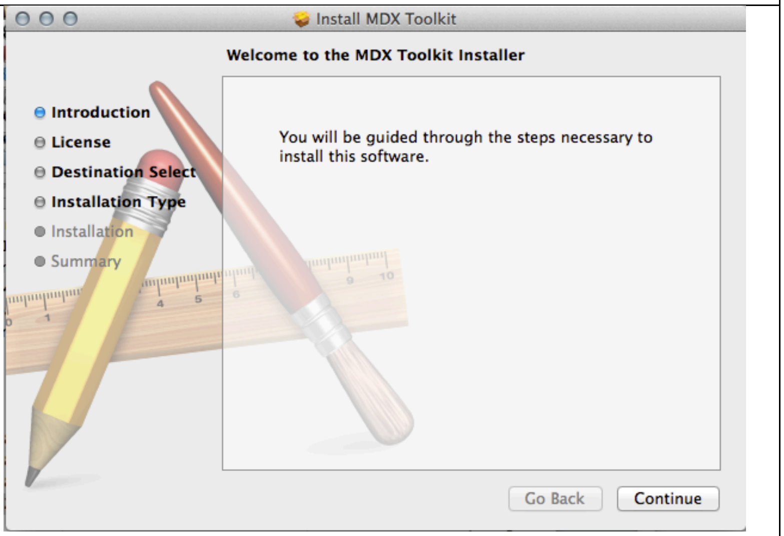
File Size: 36 MB

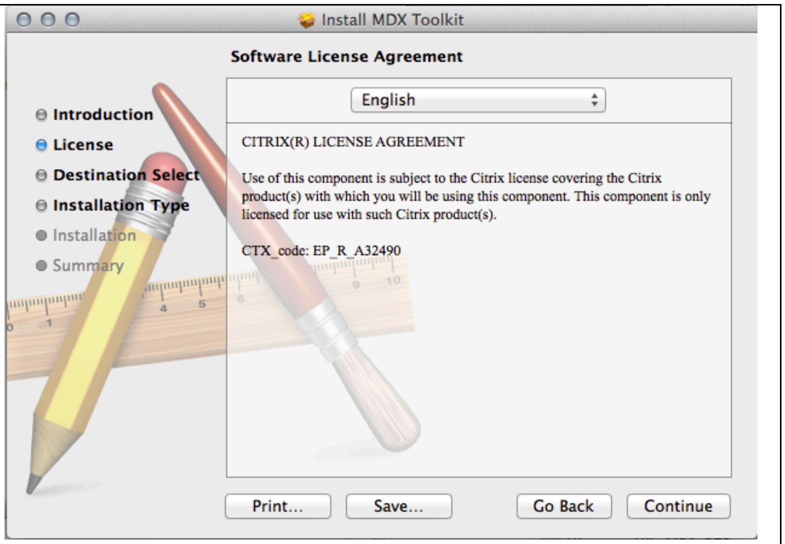
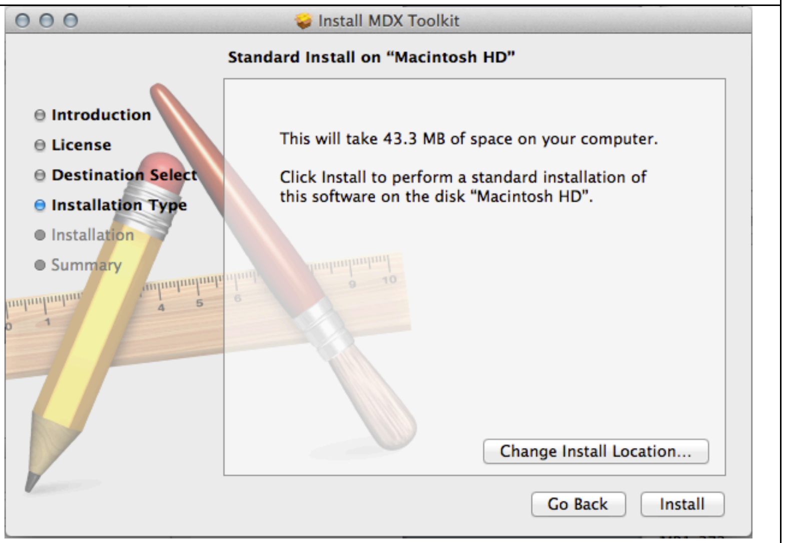
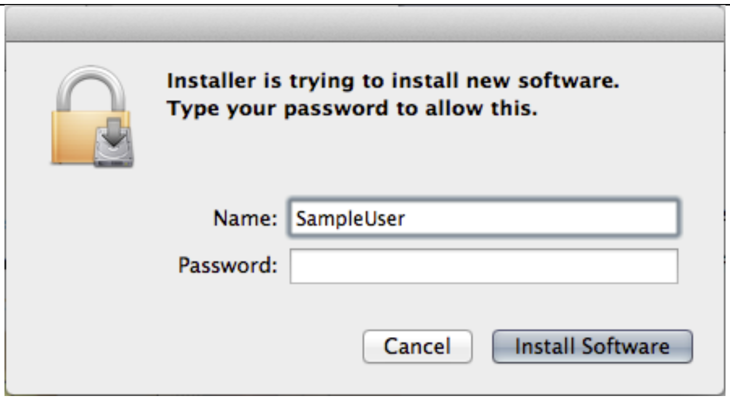
File Type: .zip

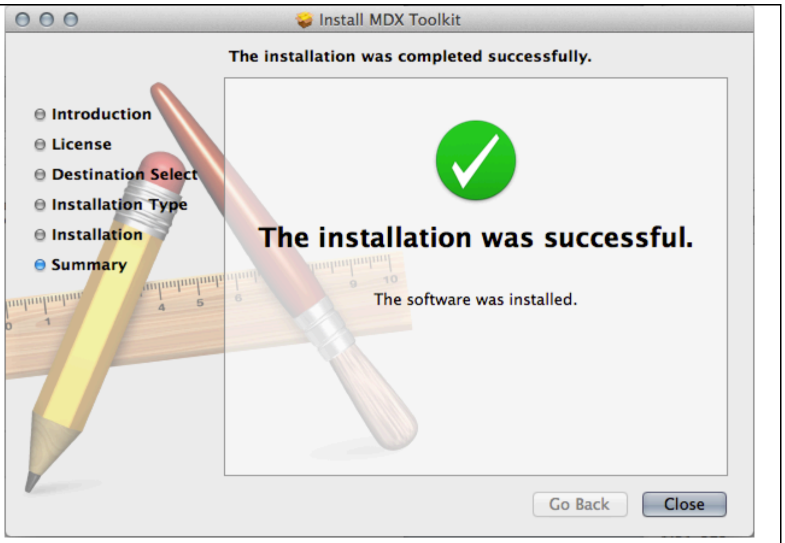
After the package has been downloaded, follow the steps in the next section of this document to install the MDX Toolkit.

Installing the MDX Toolkit

After downloading the MDX Toolkit, follow the instructions below to install the package.

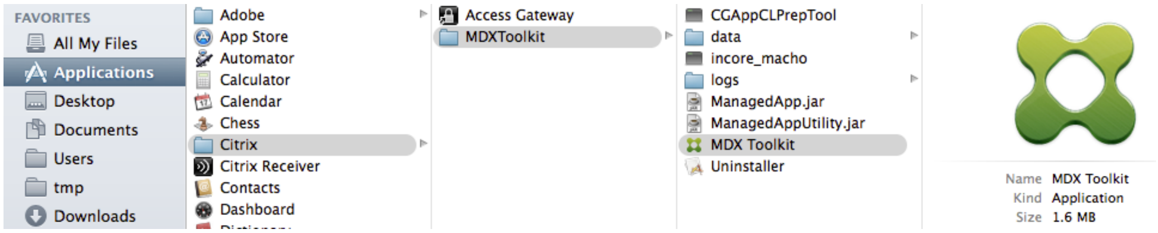
1.	Double click the **MDXToolkit.mpkg** that was downloaded.	 MDXToolkit.mpkg Name MDXToolkit.mpkg Kind Installer package Size 26.4 MB Created 2/25/14, 2:07 AM Modified 2/25/14, 2:07 AM Last opened 2/25/14, 2:07 AM
2.	The Mac installer will begin. Click **Continue.**	 Install MDX Toolkit Welcome to the MDX Toolkit Installer You will be guided through the steps necessary to install this software. Introduction License Destination Select Installation Type Installation Summary Go Back Continue

3.	Read the Software License Agreement and then click **Continue**.	 Install MDX Toolkit Introduction License Destination Select Installation Type Installation Summary English Print… Save… Go Back Continue Software License Agreement CITRIX(R) LICENSE AGREEMENT Use of this component is subject to the Citrix license covering the Citrix product(s) with which you will be using this component. This component is only licensed for use with such Citrix product(s). CTX_code: EP_R_A32490
4.	Select the desired install location and then click **Install.**	 Install MDX Toolkit Introduction License Destination Select Installation Type Installation Summary Change Install Location… Go Back Install Standard Install on "Macintosh HD" This will take 43.3 MB of space on your computer. Click Install to perform a standard installation of this software on the disk "Macintosh HD".
5.	Enter administrator password then click **Install Software**.	 Installer is trying to install new software. Type your password to allow this. Name: SampleUser Password:

6.	If the toolkit has been successfully installed, the final screen will give a success message. Click **Close** to continue.	 The installation was completed successfully. The installation was successful. The software was installed. Install MDX Toolkit Introduction License Destination Select Installation Type Installation Summary Go Back Close

The toolkit will be installed in the following location:

/Applications/Citrix/MDXToolkit/MDX Toolkit.app



Name MDX Toolkit
Kind Application
Size 1.6 MB

Prerequisites for Application Wrapping

To wrap applications after installing the MDX Toolkit, some third-party utilities are required. The Android and iOS requirements are explained in detail in the respective sections below.

Android Prerequisites

Here is a table of the Android wrapping prerequisites:

Prerequisite	Version (if applicable)
Java JDK	1.7
Android SDK	19.0.0 or later
Valid Keystore	User generated

Each prerequisite is explained in greater detail below.

- **Java JDK 1.7** – The MDX Toolkit requires the use of the Java JDK 1.7. This version needs to be installed on your computer and set as the default. Java JDK version 1.8 may be used but is not officially supported for XenMobile.

  To verify the Java version you are currently running, please run the following command from your terminal:

  ```
  java -version
  ```

  A sample output of the command is below. The version number is highlighted. **Note**: version numbers may be different depending on environment.

  ```
  java version "1.7.0_55"
  Java(TM) SE Runtime Environment (build 1.7.0_55-b15)
  Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed
  mode)
  ```

  To verify that you have the full Java JDK installed and not just the Java Runtime Environment, run the following command:

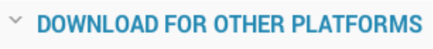
  ```
  keytool
  ```

  If this command is not found, the full JDK is not installed or is not present on your PATH environment variable. You may put the PATH of the JDK into your android_settings.txt file located in the same directory as your MDXToolkit (default is /Application/Citrix/MDXToolkit/android_settings.txt).

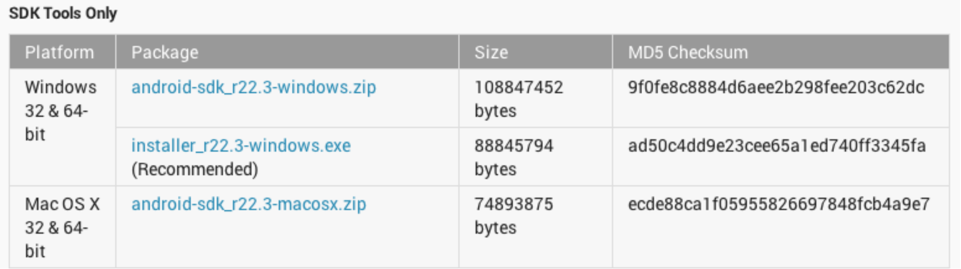
If your computer doesn't already have a suitable JDK installed, download it here and follow the installation instructions here.

- **Android SDK** – To wrap Android applications, you need to download the Android SDK from the Google website here. Then, you need to run the SDK Manager (part of the download) to choose the specific SDK version by following the instructions below.

  Scroll to the bottom of the web page and click on the drop-down arrow next to **DOWNLOAD FOR OTHER PLATFORMS**



Select the appropriate platform (Mac or Windows) from the **SDK Tools Only** section and download the compressed file.

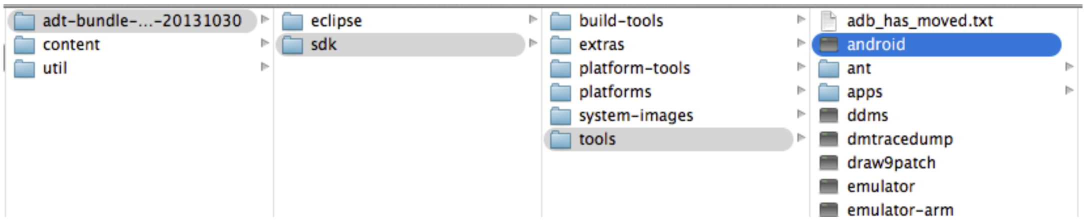


SDK Tools Only

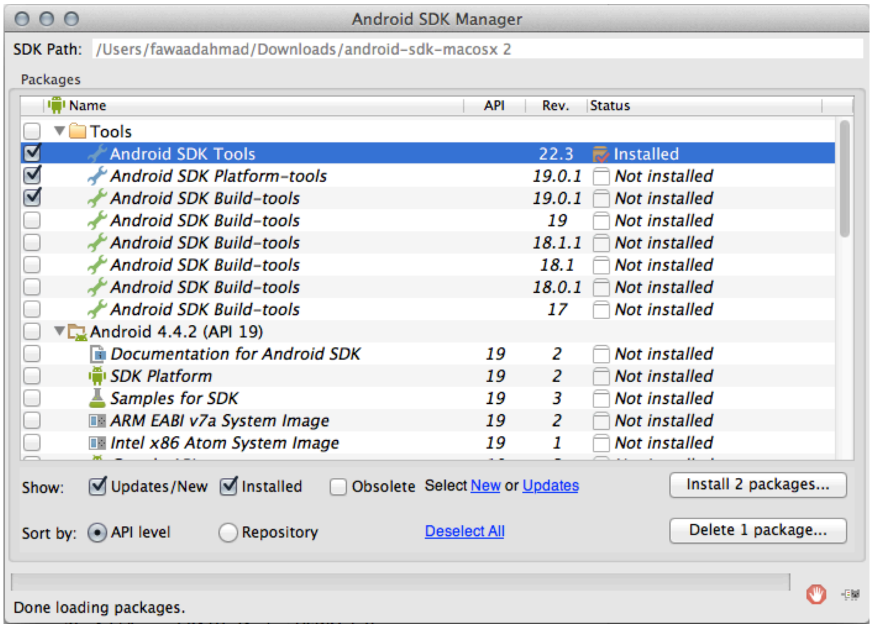
Platform	Package	Size	MD5 Checksum
Windows 32 & 64-bit	android-sdk_r22.3-windows.zip	108847452 bytes	9f0fe8c8884d6aee2b298fee203c62dc
	installer_r22.3-windows.exe (Recommended)	88845794 bytes	ad50c4dd9e23cee65a1ed740ff3345fa
Mac OS X 32 & 64-bit	android-sdk_r22.3-macosx.zip	74893875 bytes	ecde88ca1f05955826697848fcb4a9e7

Once downloaded, unzip the folder to a location of your choice (for example, ~/Desktop/AndroidSDK/). This should be a location you can easily access.

Next, navigate to the folder where you installed the Android SDK. Navigate deeper into the sdk/tools folder and open the Android ("android") application.



Install the latest version of Android SDK Tools, Android SDK Platform-tools, and Android SDK Build-tools



SDK Path: /Users/fawaadahmad/Downloads/android-sdk-macosx 2

Packages

	Name	API	Rev.	Status
□	Tools			
☑	Android SDK Tools		22.3	Installed
☑	Android SDK Platform-tools		19.0.1	□ Not installed
☑	Android SDK Build-tools		19.0.1	□ Not installed
□	Android SDK Build-tools		19	□ Not installed
□	Android SDK Build-tools		18.1.1	□ Not installed
□	Android SDK Build-tools		18.1	□ Not installed
□	Android SDK Build-tools		18.0.1	□ Not installed
□	Android SDK Build-tools		17	□ Not installed
□	Android 4.4.2 (API 19)			
□	Documentation for Android SDK	19	2	□ Not installed
□	SDK Platform	19	2	□ Not installed
□	Samples for SDK	19	3	□ Not installed
□	ARM EABI v7a System Image	19	2	□ Not installed
□	Intel x86 Atom System Image	19	1	□ Not installed

Show: ☑ Updates/New ☑ Installed □ Obsolete Select New or Updates

Sort by: ● API level ○ Repository

Done loading packages.

Add the location of the newly installed folders to the PATH variable in your environment. If this does not work, you can also specify it in the android_settings.txt file. A sample android_settings.txt file is given under the /Applications/Citrix/MDXToolkit/ directory.

1. Find the location of your downloaded SDK (such as /Android/SDK/).
2. Add the following to the end of the "PATH =" line in your settings.txt file (separated by ":" on Mac/Unix, and ";" on Windows):

```
":Android/SDK/platform-tools:/Android/SDK/build-
tools/20.0.0:/Android/SDK/tools"
```

3. You can edit your android_settings.txt file using the command below. You will be prompted to enter your User password and the file will open in your terminal window.

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

- **Valid Keystore** – The keystore is a file that contains certificates used to sign your Android application. You will create a keystore one time and retain this file for current and future wrapping usage. If you do not use the same keystore when rewrapping a previously deployed application, upgrades of that application will not work. Instead, your users need to manually remove the older version before installing the newer version.

  A sample command that creates a keystore is shown below (trailing backslash denotes line continuation).

  ```
  keytool -genkey \
  -dname "cn=Android, o=Android, c=US" \
  -keystore my.keystore \
  -storepass android \
  -alias wrapkey \
  -keypass android \
  -keysize 1024 -sigalg  SHA1withRSA -keyalg RSA
  ```

  The –*keystore* parameter indicates the filepath of the keystore. You can use any filename and path.

  The –*alias* parameter is the name of the certificate you are creating.

  The –*storepass* and –*keypass* parameters specify the keystore and certificate passwords. It is important to remember these passwords because they aren't recoverable. Also make sure to use your own values and not the values shown in the example.

  All signing certificates allowed by Google are supported. Note that the SHA-256 algorithm is only supported for devices running Android 4.3 or later.

  Since the certificate is used to validate the authenticity of the application, you should safeguard the keystore. More information about signing Android applications can be found here.

iOS Prerequisites

Here is a table of the pre-iOS 8 wrapping prerequisites:

Prerequisite	Version (if applicable)
Apple Account	Enterprise
Xcode	5.0+
Xcode Command line tools	October 2013
Valid Certificate and Provisioning Profile	User Generated

iOS 8 Wrapping Prerequisites:

Prerequisite	Version (if applicable)
OSX	Mavericks - 10.9 or newer
Xcode	5.1+
Xcode Command line tools	April 2013
MDXToolkit	9.0.2 or later
Valid Certificate and Provisioning Profile	User Generated

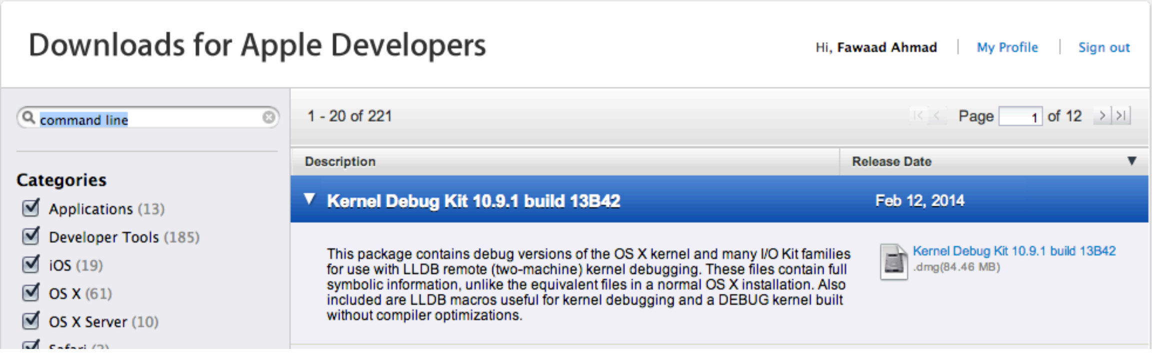
To obtain access to the prerequisites, you must register for an Apple distribution account. There are two types: Enterprise and Developer. We **strongly** recommend Enterprise accounts.

Below each prerequisite is explained in greater detail.

- **Apple Account** –
  - Enterprise
    - Deploy and test applications on an unlimited number of devices
    - Deploy without an App ID
    - Distribute Developer Certificate throughout company so developers can sign apps as well
  - Developer
    - Limited to 100 devices (testing, NOT deployment)
    - Each App ID must be specified when deploying

- **Xcode** –Install Xcode from the Mac App Store, which can be found here.

- **Xcode command-line tools** – The installation of the command-line tools for the Mavericks OS is automatic. For pre-Mavericks installation, follow the instructions below.
    - Navigate to the Downloads page on your Apple developer account located here.
    - Type and enter "command line" in the search bar on the left side of the page.



Downloads for Apple Developers

Hi, **Fawaad Ahmad** | My Profile | Sign out

command line

1 - 20 of 221   Page 1 of 12

Categories

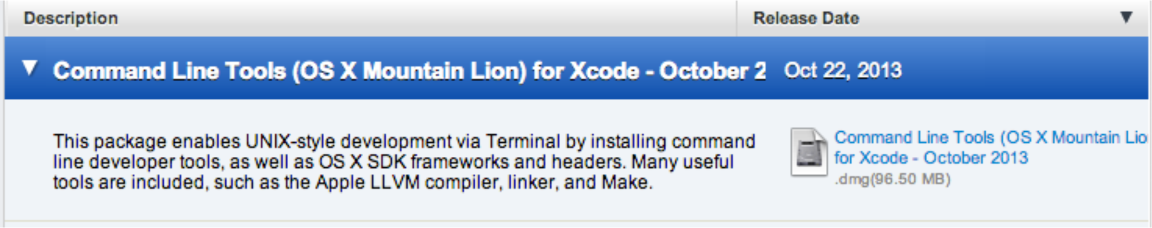
- ☑ Applications (13)
- ☑ Developer Tools (185)
- ☑ iOS (19)
- ☑ OS X (61)
- ☑ OS X Server (10)

Description	Release Date ▼
▼ **Kernel Debug Kit 10.9.1 build 13B42**	**Feb 12, 2014**

This package contains debug versions of the OS X kernel and many I/O Kit families for use with LLDB remote (two-machine) kernel debugging. These files contain full symbolic information, unlike the equivalent files in a normal OS X installation. Also included are LLDB macros useful for kernel debugging and a DEBUG kernel built without compiler optimizations.

Kernel Debug Kit 10.9.1 build 13B42 .dmg(84.46 MB)

  - Download and install the latest Command Line Tools for Xcode for your specific OS (for example, Mountain Lion).



Description	Release Date ▼
▼ **Command Line Tools (OS X Mountain Lion) for Xcode - October 2**	**Oct 22, 2013**

This package enables UNIX-style development via Terminal by installing command line developer tools, as well as OS X SDK frameworks and headers. Many useful tools are included, such as the Apple LLVM compiler, linker, and Make.

Command Line Tools (OS X Mountain Lio for Xcode - October 2013 .dmg(96.50 MB)

  - After the command line tools are installed, make sure you launch Xcode once so it can automatically trigger a check for the prerequisites.

- **iOS Certificate and Provisioning Profile**
    - If you do not have these, please follow the steps on creating a new pair located in the Appendix. Make sure both the private key and the certificate are installed on your Mac Keychain Access before wrapping. If you are transferring the certificate and profile pair to a new computer, follow the steps in for importing existing certificate in the Appendix.
    - **iOS8 requirements –**
        - Valid certificate that contains a valid OU field in the subject
        - Before generating a new certificate, check if the existing one has OU, in which case it can be used. Certificate must be installed in the Keychain of the wrapping workstation
        - Provisioning profile that references above certificate
        - Remove old certificate and provisioning profile from keychain and Mac, respectively

Wrapping Enterprise Applications

The most common wrapping scenario is wrapping enterprise applications. The following sections explain the ways to wrap the applications. Make sure the MDX Toolkit is installed and the prerequisites described earlier have been met.

NOTE: The paths referenced below are based on standard installation locations. Please adjust these for your system or environment as necessary.

Command Line for Android

The most comprehensive way to wrap an Android application is by using the command line. A major benefit of this method is that you can write scripts to automate the wrapping process. The command-line tool provides numerous customization options.

The command below shows the most basic wrapping of an application. Default settings are used and the app is signed with the provided keystore. Modify the bold information for your specific system. The trailing backslash signifies the command continues to the next line. Please remove these symbols before running the command.

Because the /Applications/ directory is restricted, you may need to run the following command in super user mode. To do this, you simply need to 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
```

The following are examples of additional options you may add to the command above:

```
-appName "Wrapped Sample app"

-appDesc "This is my newly wrapped Android application."
```

For inline documentation, use the –*help* option. Additionally, the table on the next page shows the available options.

```
java -jar /Applications/Citrix/MDXToolkit/ManagedAppUtility.jar –help
```

Usage and complete set of options:

```
java –jar ManagedAppUtility.jar [options] [command] [command options]
```

	Keyword	Description
Options		
	-help	Display command format
Commands		
	wrap	Wrap the input file
	sign	Sign the input file
	setinfo	Modifies settings in MDX file
	getInfo	Retrieve settings from MDX file
	version	Displays wrapper version

Each command is further described below with its relevant command options. Please note the Optional and Required tags and descriptions in the right-hand column.

```
java –jar ManagedAppUtility.jar wrap [command options]
```

Wrap command options:

-Help	Displays command specific help.	When used, this parameter overrides all others.
Basic Information		
-In	Path of APP to be wrapped.	Required. App must exist and must have correct extension.
-Out	Path for resulting MDX/APP file.	Optional. If omitted, use same dir/filename except with .mdx extension.
Application Type		
-AppType	Values are MDXOnly, General, and Premium.	Optional. If omitted, defaults to **MDXOnly**.
Signing Parameters		
-KeyStore	Path to the keystore file.	Optional. These must all be specified in which case the APK is signed. Alternately they must all be omitted, in which case the APK isn't signed.
-StorePass	Password for the keystore.	
-KeyAlias	Name of the specific key in the keystore.	
-KeyPass	Password for the specific key in the keystore.	
-SigAlg	Algorithm to use when signing.	Optional.
-DebugCert	Create and use a temporary keystore.	Optional. If included, the four parameters above must also be included.
Policy and Settings		
-AppName	Application name.	Optional. Defaults to value read from APP if possible.
-AppDesc	Application description.	Optional. Defaults to value read from APP if possible.

`-MinPlatform`	Minimum allowed SDK level.	Optional. Defaults to blank.
`-MaxPlatform`	Maximum allowed SDK level.	Optional. Defaults to blank.
`-ExcludedDevices`	List of disallowed hardware platforms.	Optional. Defaults to blank.
`-StoreURL`	URL of the APP on an app store.	Optional. Defaults to blank.
`-PolicyXML`	A path that contains a replacement XML policy definition file.	Optional. Defaults to usage of the internal policy definitions. A template is given in the MDXToolkit directory under the /data/MDXSDK_Android/ folder

```
java -jar ManagedAppUtility.jar sign [command options]
```

Sign command options:

`-Help`	Displays command specific help.	When used, this parameter overrides all others.
Basic Information		
`-In`	Path of APP or MDX containing an APP.	Required. File must exist and must have the expected extension.
`-Out`	Path to output APP or MDX.	Optional. If omitted, signing will be done in place.
Signing Parameters		
`-KeyStore`	Path to the keystore file.	Required.
`-StorePass`	Password for the keystore.	
`-KeyAlias`	Name of the specific key in the keystore.	
`-KeyPass`	Password for the specific key in the keystore.	
`-SigAlg`	Algorithm to use when signing.	Optional.
`-DebugCert`	Create and use a temporary keystore.	Optional.

```
java -jar ManagedAppUtility.jar setinfo [command options]
```

SetInfo command options:

`-Help`	Displays command specific help.	When used, this parameter overrides all others.
Basic Information		
`-In`	Path of MDX to be modified. This file will be modified in place.	Required. File must exist and must have correct extension.
`-Out`	Path of resulting MDX.	Optional. If omitted, update will be done in place.
Policy and Settings		
`-AppName`	Application name.	Optional. If not supplied, it remains unchanged.
`-AppDesc`	Application description.	Optional. If not supplied, it

		remains unchanged.
`-MinPlatform`	Minimum allowed SDK level.	Optional. If not supplied, it remains unchanged.
`-MaxPlatform`	Maximum allowed SDK level.	Optional. If not supplied, it remains unchanged.
`-ExcludedDevices`	List of disallowed hardware platforms.	Optional. If not supplied, it remains unchanged.
`-StoreURL`	URL of the APP on an app store.	Optional. If not supplied, it remains unchanged.

```
java –jar ManagedAppUtility.jar getinfo [command options]
```

GetInfo command options:

`-Help`	Displays command specific help.	When used, this parameter overrides all others.
Basic Information		
`-In`	Path of MDX used to retrieve settings.	Required.
`-Out`	Path of settings file to be written.	Optional. If present, the settings will be written to this file.

Command Line for iOS

Like the Android command line, this command line also allows you to wrap applications. The trailing backslash signifies that the example continues to the next line. Please remove these symbols before running the command.

To perform these commands, you must navigate to the `/Applications/Citrix/MDXToolkit/` directory on your command line.

The basic iOS wrapping command line looks as follows:

```
./CGAppCLPrepTool \
Wrap \
-Cert CERTIFICATE \
-Profile PROFILE \
-in INPUT_FILE \
-out OUTPUT_FILE
```

The following is an example of this option:

```
./CGAppCLPrepTool \
Wrap \
-Cert “iPhone Developer: Joe Admin (12MMA4ASQB)” \
-Profile “team_profile.mobileprovision” \
-in “~/Desktop/SampleApps/Sample.ipa” \
-out “~/Desktop/SampleApps/Sample.mdx”
```

Here are examples of additional options you may add to the command above:

```
-appName “Wrapped Sample app”

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

For a full list of the options, type this into your terminal. A table is also given for reference on the next page.

```
./CGAppCLPrepTool
```

Usage and complete set of options:

```
./CGAppCLPrepTool [options] [commands] [command options]
```

	Keyword	**Description**
Options		
	`-help`	Display command format
Commands		
	`wrap`	Wrap the input file
	`sign`	Sign the input file
	`setinfo`	Modifies settings in MDX file
	`version`	Displays wrapper version

The iOS commands are described in detail below.

```
./CGAppCLPrepTool  wrap [command options]
```

Wrap command options:

Help		
`-Help`	Displays command specific help.	When used, this parameter overrides all others.
Basic Information		
`-In`	Path of APP to be wrapped.	Required. App must exist and must have correct extension.
`-Out`	Path for resulting MDX/APP file.	Optional. If omitted, use same dir/filename except with .mdx extension.
Application Type		
`-IsSdk`	Values are **true** or **false**. If no value specified, **true** is assumed.	Optional. If omitted, defaults to **false**.
`-AppMode`	Values are 1 or 2.	Optional. This parameter is only valid when the IsSDK parameter is true. A value of 1 means Premium and a value of 2 means General.
Signing Parameters		
`-Cert`	Certificate.	Required.
`-Profile`	Profile.	
Policy and Settings		
`-AppName`	Application name.	Optional. Defaults to value read from APP if possible.
`-AppDesc`	Application description.	Optional. Defaults to value read from APP if possible.
`-MinPlatform`	Minimum allowed SDK level.	Optional. Defaults to blank.
`-MaxPlatform`	Maximum allowed SDK level.	Optional. Defaults to blank.
`-ExcludedDevices`	List of disallowed hardware platforms.	Optional. Defaults to blank.

-StoreURL	URL of the APP on an app store.	Optional. Defaults to blank.
-PolicyXML	A path that contains a replacement XML policy definition file.	Optional. Defaults to usage of the internal policy definitions. A template is located in the MDXToolkit directory - /data/policy_metadata.xml

```
./CGAppCLPrepTool  sign [command options]
```

Sign command options:

-Help	Displays command specific help.	When used, this parameter overrides all others.
Basic Information		
-In	Path of APP or MDX containing an APP.	Required. File must exist and must have the expected extension.
-Out	Path to output APP or MDX.	Optional. If omitted, signing will be done in place.
Signing Parameters		
-Cert	Certificate.	Required.
-Profile	Profile.	

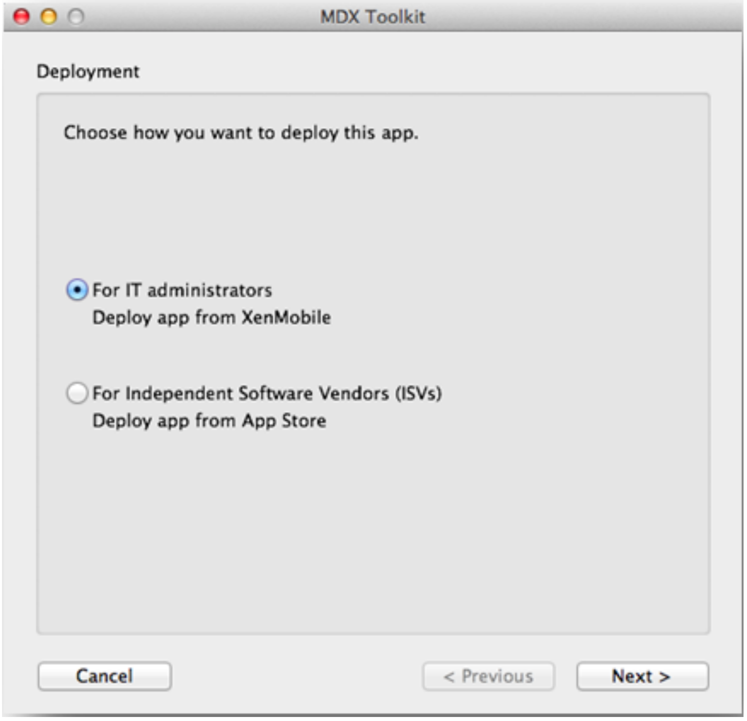
```
./CGAppCLPrepTool  setinfo [command options]
```

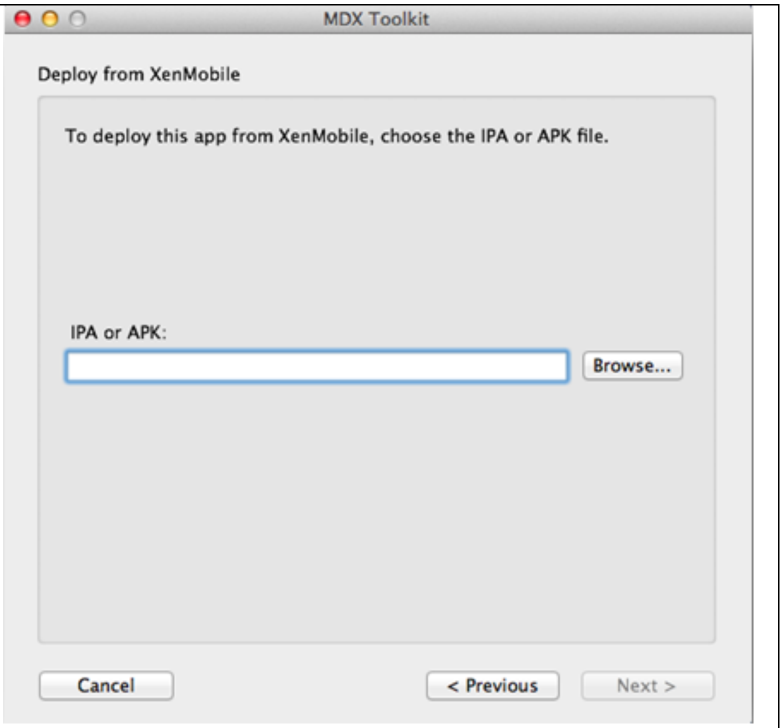
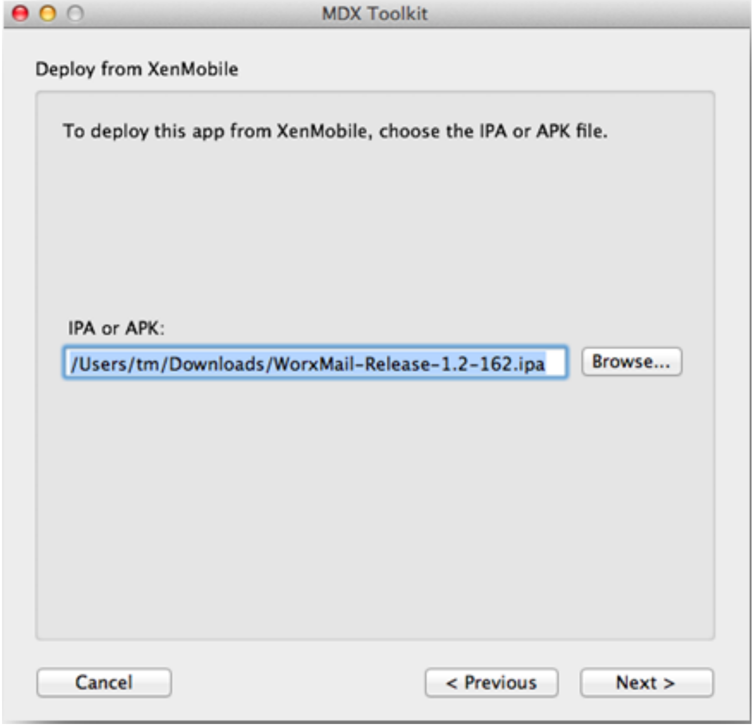
SetInfo command options:

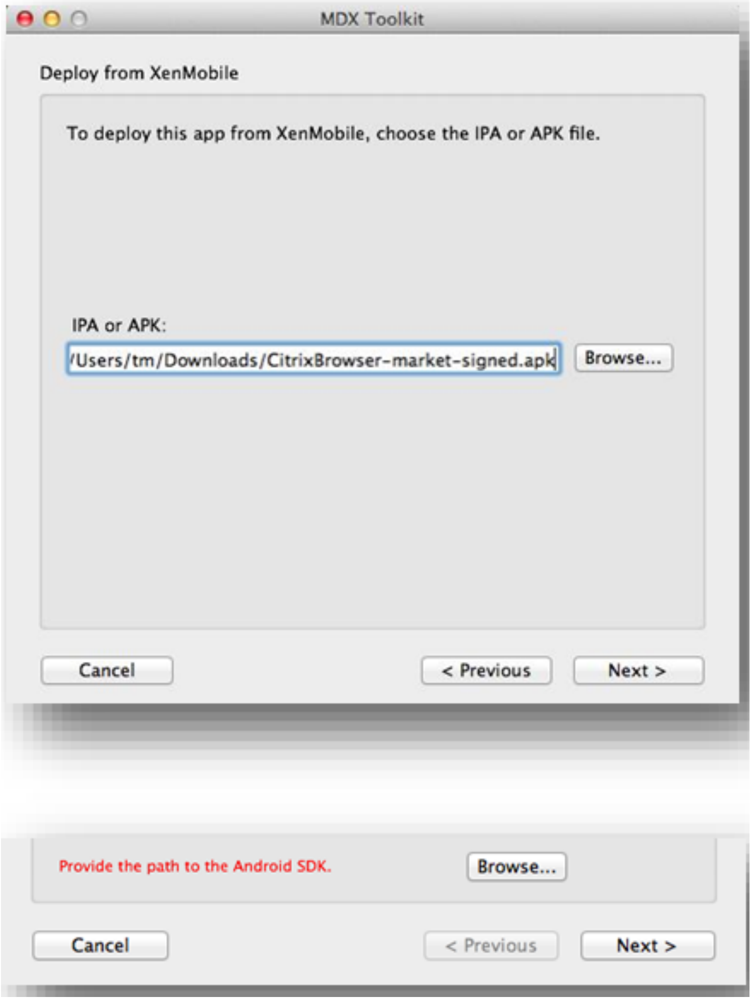
-Help	Displays command specific help.	When used, this parameter overrides all others.
Basic Information		
-In	Path of MDX to be modified. This file will be modified in place.	Required. File must exist and must have correct extension.
-Out	Path of resulting MDX.	Optional. If omitted, update will be done in place.
Policy and Settings		
-AppDesc	Application description.	Optional. If not supplied, it remains unchanged.
-MinPlatform	Minimum allowed SDK level.	Optional. If not supplied, it remains unchanged.
-MaxPlatform	Maximum allowed SDK level.	Optional. If not supplied, it remains unchanged.
-ExcludedDevices	List of disallowed hardware platforms.	Optional. If not supplied, it remains unchanged.
-StoreURL	URL of the APP on an app store.	Optional. If not supplied, it remains unchanged.

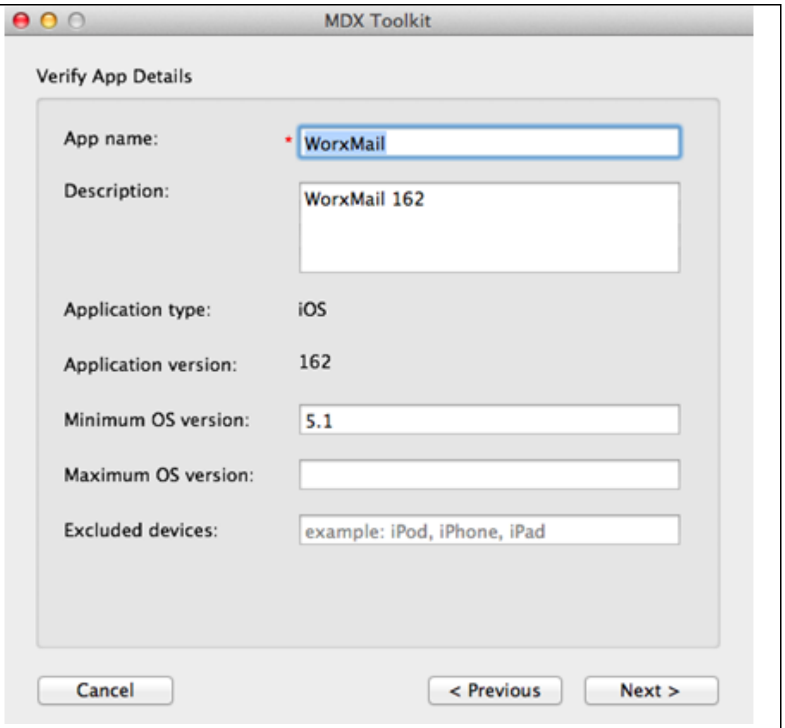
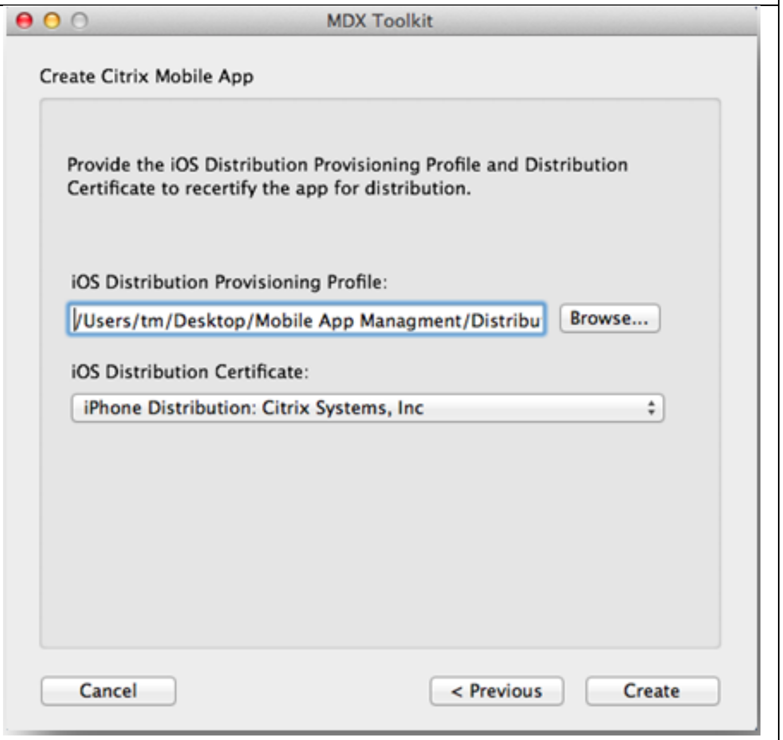
Wrapping with the GUI Tool

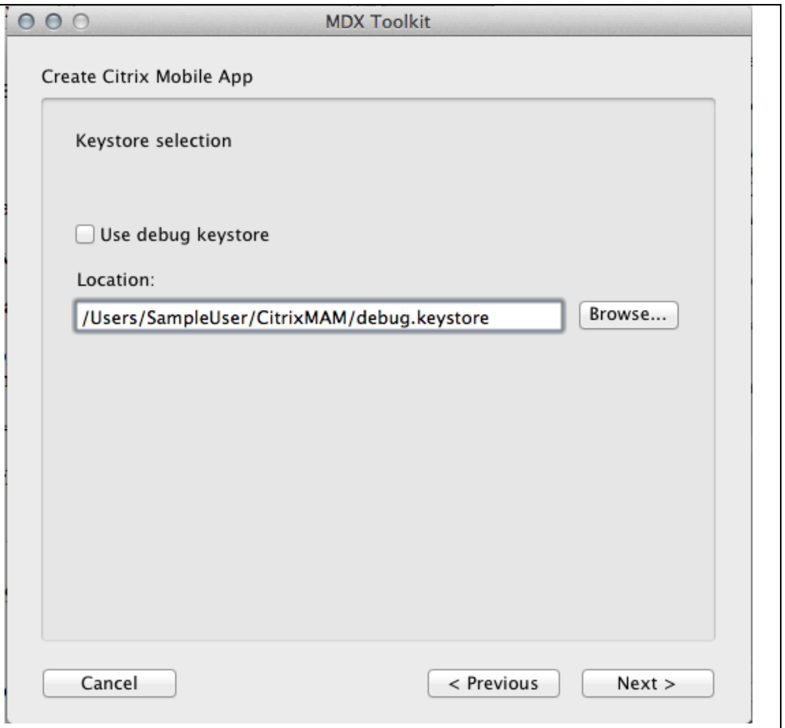
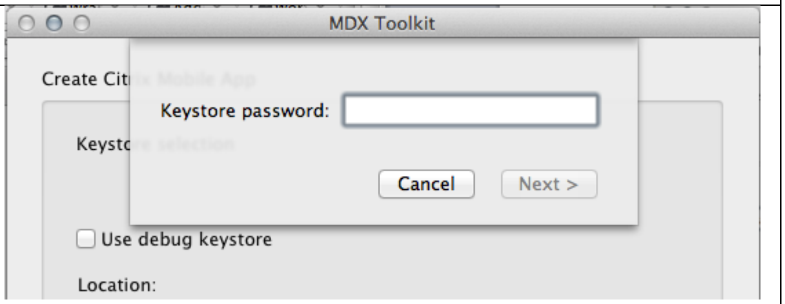
In the workflow below, common steps use a white background in the first column. Steps specific to iOS use a gray background and steps specific to Android use a green background.

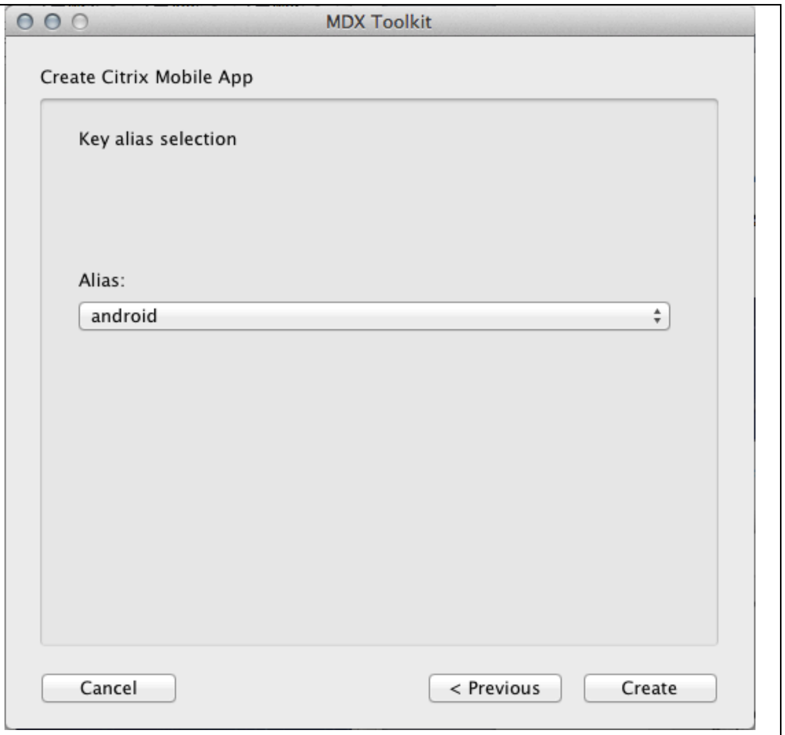
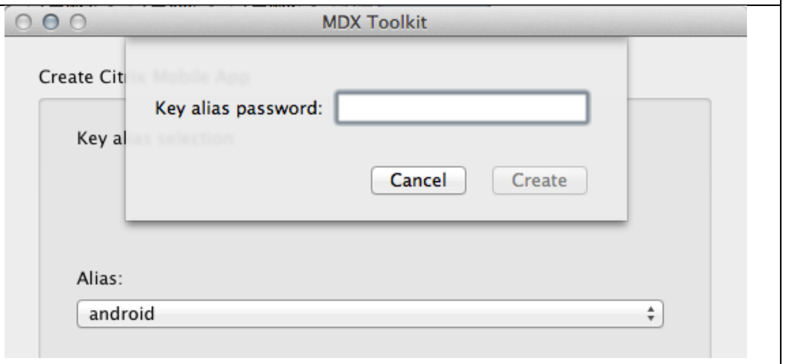
1.	Open the Citrix MDX Toolkit from your Applications folder.	
2.	Select **For IT administrators** and click **Next.**	 Choose how you want to deploy this app. ● For IT administrators Deploy app from XenMobile ○ For Independent Software Vendors (ISVs) Deploy app from App Store

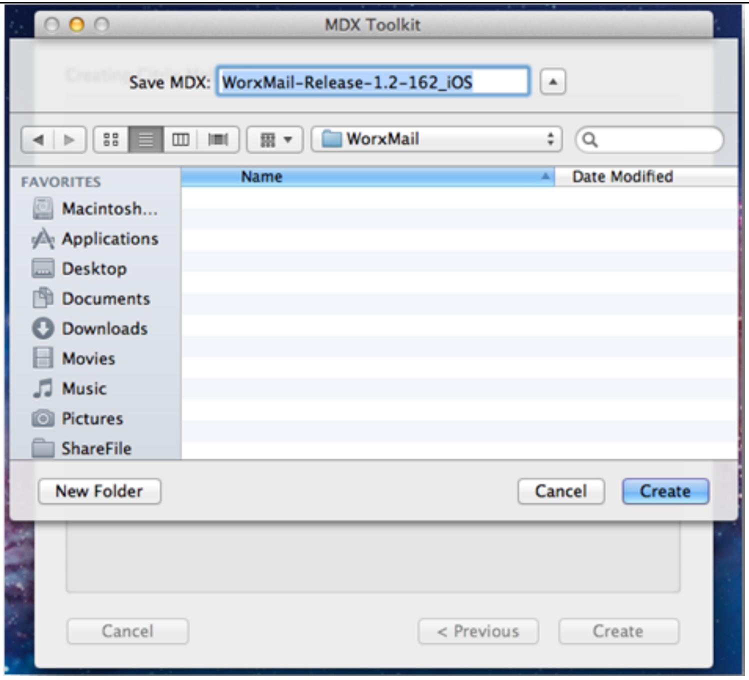
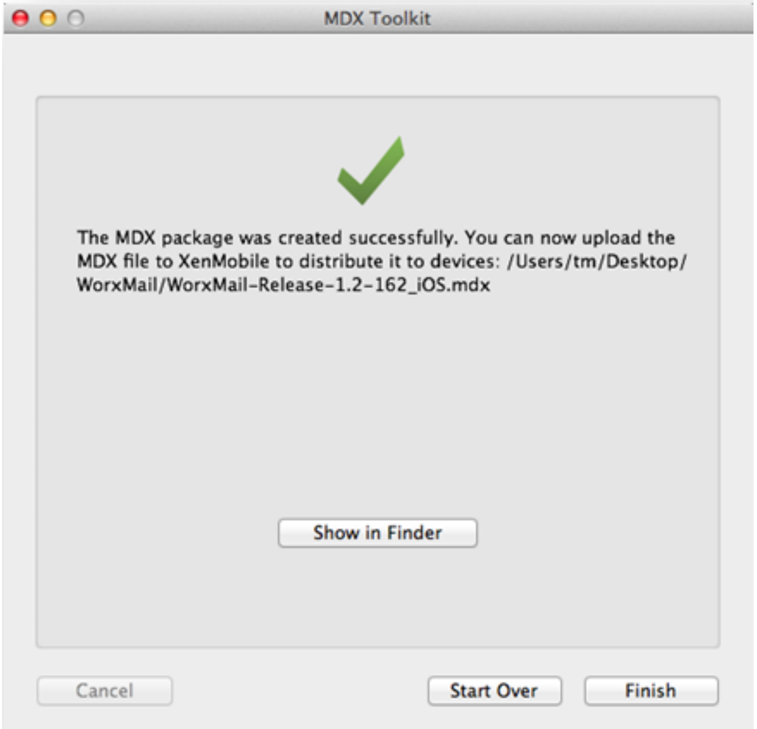
3.	Click **Browse** and then select either an .ipa file or an .apk file.	 MDX Toolkit Deploy from XenMobile To deploy this app from XenMobile, choose the IPA or APK file. IPA or APK:
4a.	Click **Next** after selecting an appropriate file (An iOS example is shown in the figure).	 MDX Toolkit Deploy from XenMobile To deploy this app from XenMobile, choose the IPA or APK file. IPA or APK: /Users/tm/Downloads/WorxMail-Release-1.2-162.ipa

4b.	The first time you use the MDX Toolkit to prepare an .apk file, you may be prompted to provide the path to the Android SDK. Click **Browse** and provide the path to the high level Android SDK folder. This folder should contain the platform-tools, tools, and build-tools folders that you downloaded in the prerequisites section. Click **Next** after selecting the proper SDK. If you are prompted to select the SDK again, please make sure you have pointed to the correct SDK location.	 MDX Toolkit Deploy from XenMobile To deploy this app from XenMobile, choose the IPA or APK file. IPA or APK: /Users/tm/Downloads/CitrixBrowser-market-signed.apk Provide the path to the Android SDK.

5.	The app name and description are automatically initialized from the application. However, you can change them as desired. Optionally, provide a minimum and maximum OS version for which the app was designed and a comma-separated list of devices you would like to exclude. All of this information can also be modified when the app is published to the App Controller. Once finished, click **Next.**	 Verify App Details App name: * WorxMail Description: WorxMail 162 Application type: iOS Application version: 162 Minimum OS version: 5.1 Maximum OS version: Excluded devices: example: iPod, iPhone, iPad
6a.	For iOS, click **Browse** and select the iOS Distribution Provisioning Profile. The associated Distribution Certificate will auto-populate in the iOS Distribution Certificate drop-down list. **Note**: If the associated distribution certificate does not have the private key installed into Keychain Access, the iOS Distribution Certificate will not auto-populate. It is mandatory to have both key and certificate installed onto your Mac Keychain Access before getting to this screen. Refer back to the prerequisites section of this document.	 Create Citrix Mobile App Provide the iOS Distribution Provisioning Profile and Distribution Certificate to recertify the app for distribution. iOS Distribution Provisioning Profile: /Users/tm/Desktop/Mobile App Managment/Distribu iOS Distribution Certificate: iPhone Distribution: Citrix Systems, Inc

6b.	For Android, click **Browse** and then select the keystore containing the certificate you would like to use. For testing purposes, you may simply use the debug keystore and continue to step 7 below. The debug keystore is automatically created. Click **Next.**	 MDX Toolkit Create Citrix Mobile App Keystore selection ☐ Use debug keystore Location: /Users/SampleUser/CitrixMAM/debug.keystore
	When prompted, enter the keystore password and then click **Next.**	 MDX Toolkit Create Cit Keysto Keystore password: ☐ Use debug keystore Location:

	Select the Alias (certificate) from the drop-down list and then click **Create.**	 MDX Toolkit Create Citrix Mobile App Key alias selection Alias: android
	Enter the certificate password and then click **Create.**	 MDX Toolkit Create Cit Key alias password: Key al Alias: android

7.	A dialog box appears prompting you to provide a name for the resultant MDX file and a location to which you would like to save the file. Click **Create.**	 Save MDX: WorxMail-Release-1.2-162_iOS MDX Toolkit WorxMail FAVORITES Macintosh... Applications Desktop Documents Downloads Movies Music Pictures ShareFile Name Date Modified New Folder Cancel Create Cancel < Previous Create
8.	You will receive a confirmation screen stating that the file has been created successfully. Click **Finish** or **Start Over** if you would like to wrap another app.	 The MDX package was created successfully. You can now upload the MDX file to XenMobile to distribute it to devices: /Users/tm/Desktop/WorxMail/WorxMail-Release-1.2-162_iOS.mdx MDX Toolkit Show in Finder Cancel Start Over Finish

Wrapping ISV Applications

This section will highlight the key additions needed for wrapping ISV applications. Refer to the Wrapping Enterprise Applications section for the common format.

Command Line for Android

To wrap as an ISV application, the –*apptype* parameter must be set using one of the first two examples below.

To wrap an app as a Premium app, which means with some policy enforcement, add the following option:

```
-apptype Premium
```

To wrap as a General app, which means no policy enforcement, add the following instead:

```
-apptype General
```

If you need to upload the wrapped APK file to an App Store or webserver and the URL is known when wrapping, add the option shown below. Make sure the *apptype* parameter is set.

```
-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/Sample.mdx \
-storeURL \
“https://play.google.com/store/apps/details?id=com.zenprise”
```

Command Line for iOS

To generate wrapped ISV applications for iOS, start with the basic wrapping command from the Wrapping Enterprise Applications section above. Then add the options shown below for a Premium app or a General app.

Premium:

```
      -issdk “yes” –appMode “1”
```

General:

```
      -issdk “yes” –appMode “2”
```

If you wish to upload the wrapped IPA file to the Apple App Store or a webserver and the URL is known at the time of wrapping, also add the following option:

```
      -storeURL “https://itunes.apple.com/us/app/worx-home-by-
      citrix/id434682528?mt=8”
```

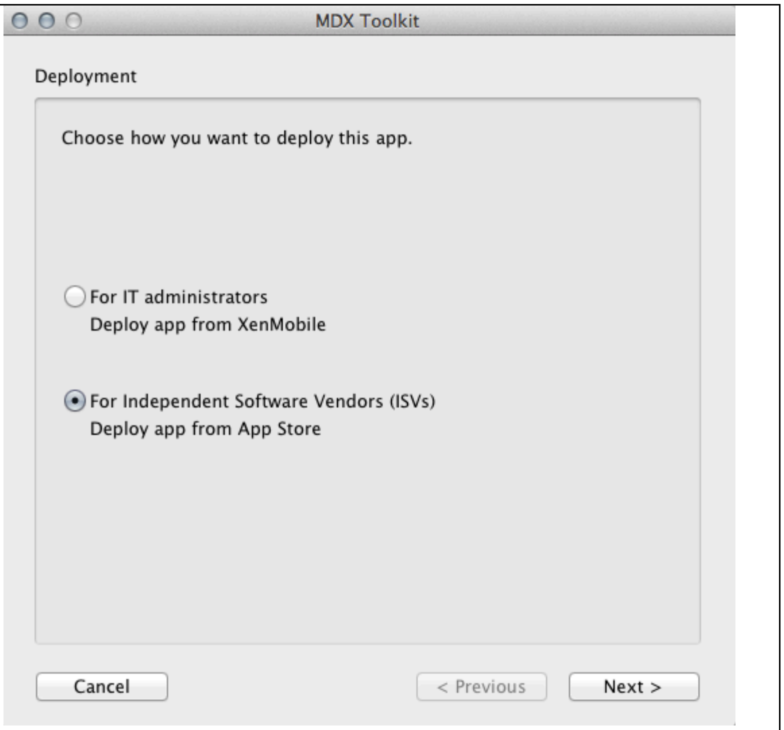
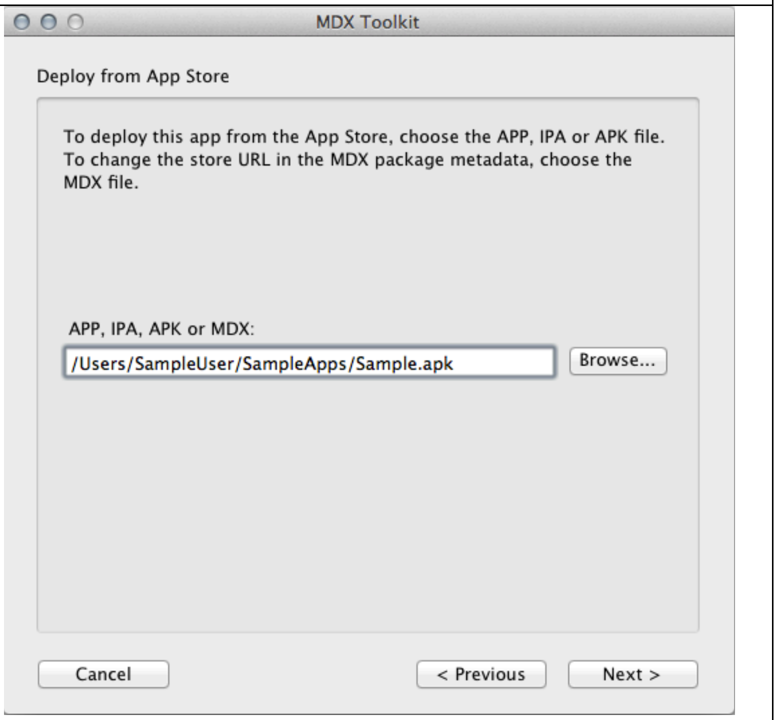
If you do not know the URL at the time of wrapping, you can modify the .MDX file later with the following command:

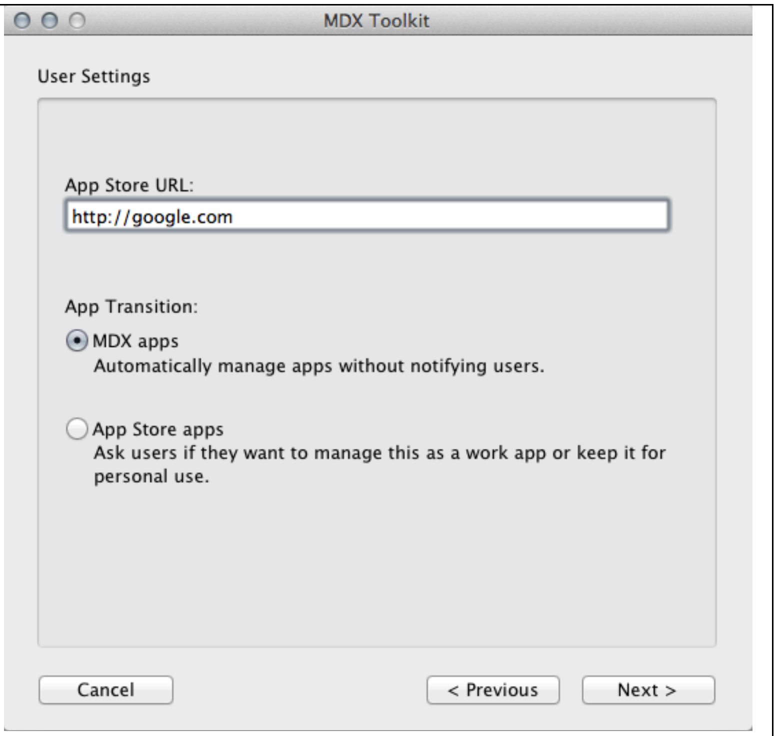
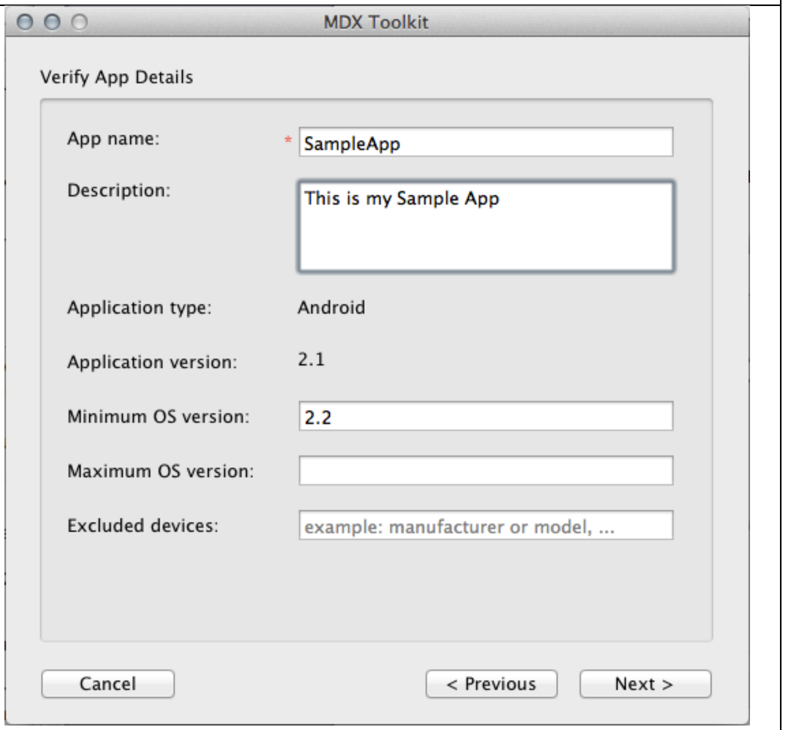
```
      ./CGAppCLPrepTool \
      setinfo \
      -in "~/Desktop/SampleApps/Sample.mdx" \
      -out “~/Desktop/SampleApps/Sample.mdx "  \
      -storeURL
      "https://itunes.apple.com/us/app/w1browser/id579414750?ls=1&mt=8"
```

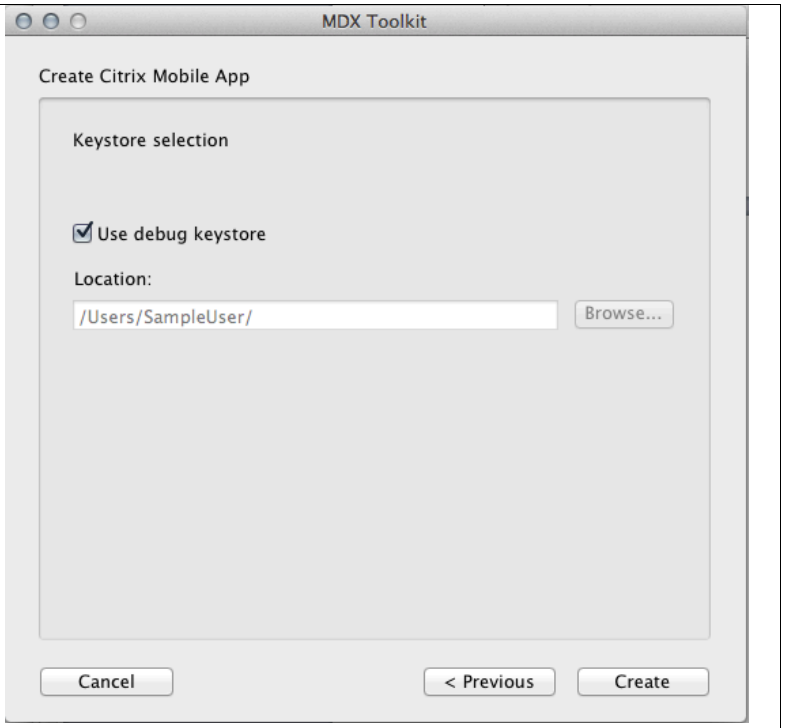
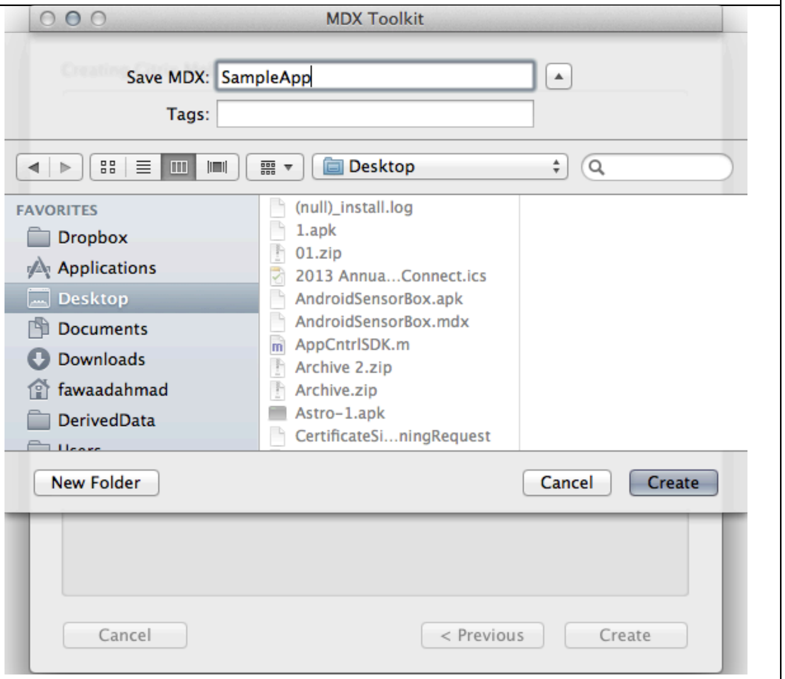
If you wish to resign an IPA file, you can do so with the following command:

```
      ./CGAppCLPrepTool \
      sign \
      –Cert “iPhone Developer: Joe Admin (12MMA4ASQB)” \
      –Profile “team_profile.mobileprovision” \
      –in “~/Desktop/SampleApps/Sample.ipa” \
      –out “~/Desktop/SampleApps/Sample.ipa”
```

Wrapping with the GUI Tool

1.	This is very similar to GUI wrapping of enterprise applications. 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
2.	Select **Browse**. Find the APP, IPA, APK, or MDX file to wrap and then click **Next**.	 Deploy from App Store To deploy this app from the App Store, choose the APP, IPA or APK file. To change the store URL in the MDX package metadata, choose the MDX file. APP, IPA, APK or MDX: /Users/SampleUser/SampleApps/Sample.apk

3.	Enter the App Store URL, if available. Choose **MDX apps** for Premium or **App Store apps** for General apps. Click **Next**. If the URL is not known, you may either leave the field blank or enter a placeholder such as “https://play.google.com/store/apps/details?id=com.zenprise”	 User Settings 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.
4.	Modify any desired application information. Click **Next**.	 Verify App Details App name: * SampleApp Description: This is my Sample App Application type: Android Application version: 2.1 Minimum OS version: 2.2 Maximum OS version: Excluded devices: example: manufacturer or model, ...

5.	Choose keystore/provisioning profile. Details can be found in the Wrapping Enterprise Applications section. Click **Create**.	 MDX Toolkit Create Citrix Mobile App Keystore selection ☑ Use debug keystore Location: /Users/SampleUser/
6.	Save application in desired location. Click **Create**.	 MDX Toolkit Save MDX: SampleApp Tags: Desktop FAVORITES Dropbox Applications Desktop Documents Downloads fawaadahmad DerivedData (null)_install.log 1.apk 01.zip 2013 Annua...Connect.ics AndroidSensorBox.apk AndroidSensorBox.mdx AppCntrlSDK.m Archive 2.zip Archive.zip Astro-1.apk CertificateSi...ningRequest New Folder Cancel Create Cancel < Previous Create
7.	A success message appears. If you see an error, please see Troubleshooting section.	

MDX Policies

When wrapping applications, users have the ability to preset the policies that will be enforced within their app. To have the application use these specific policies, you will have to wrap the app by using the command line tool. For information on how to create custom policies, see Creating Custom Policies in the Worx SDK section of this document.

Android Policies

A sample policy XML file is located at /Applications/Citrix/MDXToolkit/data/MDXSDK_Android/default_sdk_policies.xml. Modify the values here to the desired values then proceed to wrap the app as described in the preceding wrapping sections with the added *–policyxml* parameter as shown here.

```
      -policyxml /Applications/Citrix/MDXToolkit/data/MDXSDK_Android/
default_sdk_policies.xml
```

Default Policies

When wrapping as a Premium app, the policies below may be enforced on the unmanaged app. General apps do not have any policy enforcement, so any changes will take effect once the app becomes managed.

Each policy is listed below its respective category. The default values are set to FALSE/OFF, but they may be changed. All policies missing from this list require the use of Worx Home and a full XenMobile environment – they will be ignored when running in an unmanaged mode (ISV app).

a. **Device Security:**
   - Block jailbroken or rooted
   - Require device encryption
   - Require device PIN or password
   - Require device pattern screen lock

b. **Network Requirements:**
   - Require Wifi

c. **App Restrictions:**
   - Block camera
   - Block mic record
   - Block location services
   - Block SMS compose
   - Block screen capture
   - Block device sensor
   - Block NFC
   - Block application logs

d. **Network Access:**
   - Network Access

iOS Policies

A sample policy XML file is located at /Applications/Citrix/MDXToolkit/data/ policy_metadata.xml. Modify the values here to the desired values then proceed to wrap the iOS app as described in the preceding wrapping sections with the added *–policyxml* parameter as shown here.

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

Default Policies

When wrapping as a Premium app, the following policies may be enforced on the unmanaged app. General apps do not have any policy enforcement, so any changes will take effect once the app becomes managed.

Each policy is listed below its respective category. The default values are set to FALSE/OFF, but they may be changed. All policies missing from this list require the use of Worx Home and a full XenMobile environment – they will be ignored when running in an unmanaged mode (ISV app).

a. BlockLogs
b. CutAndCopy
c. DefaultLoggerLevel
d. DefaultLoggerOutput
e. DisableBackup
f. DisableCamera
g. DisableEmail
h. DisableLocation
i. DisableMicrophone
j. DisableDictation
k. DisablePrinting
l. DisableSms
m. DisableiCloud
n. DisableAirDrop
o. DisableSocialMedia
p. DocumentExchange
q. MaxLogFiles
r. MaxLogFileSize
s. InternalWifiNetworks
t. InboundURLFilter
u. NetworkAccess
v. ObscureScreen
w. OutboundURLFilter
x. Paste
y. RequireInternalNetwork
z. WifiOnly

Worx SDK

This section will highlight the uses of the Worx App SDK, how developers can consume it in their respective applications, and deploy the Worx-enabled app to the app store. Application developers can leverage the APIs to take advantage of Citrix wrapping technology to check for and enforce additional policies, check whether an application is wrapped, or if their application is managed. Using the steps in the preceding Wrapping section, the admin can wrap a newly Worx-enabled app as an ISV app making sure to specify whether they want a General (App Store) or a Premium (MDX) app.

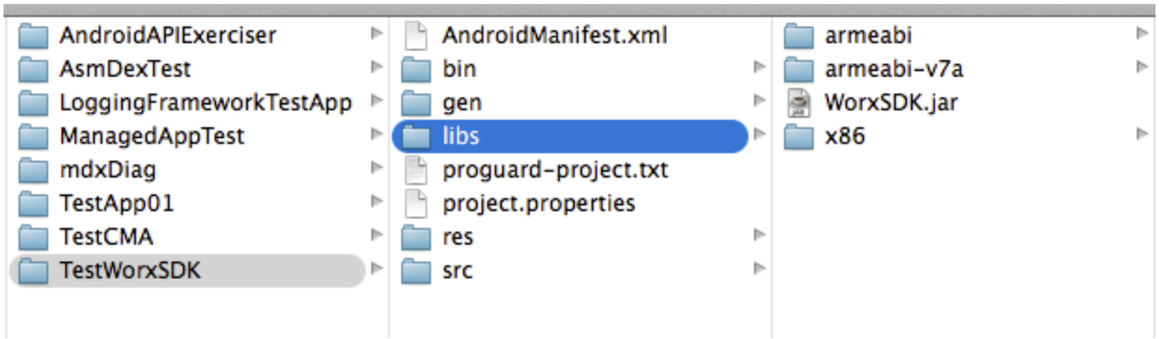
When using the Worx Framework, the admin may also add custom policies to the application. These policies are attached to the app when it is wrapped and enforced once it becomes managed.

Worx SDK for Android

This section lists instructions for adding the Worx SDK to your Android application. The steps are based on the Eclipse IDE. Android Studio and other IDEs may have different steps. See the Libraries section to determine potential conflicts.

Adding the SDK to your Application

- Add the WorxSDK.jar file and the accompanying libraries to your application's libs folder. This example uses the sample app TestWorxSDK. If the libs folder is not present, you will need to create it. You will only need to copy the libs for the platforms the app will be running on. If unsure, copy them all.

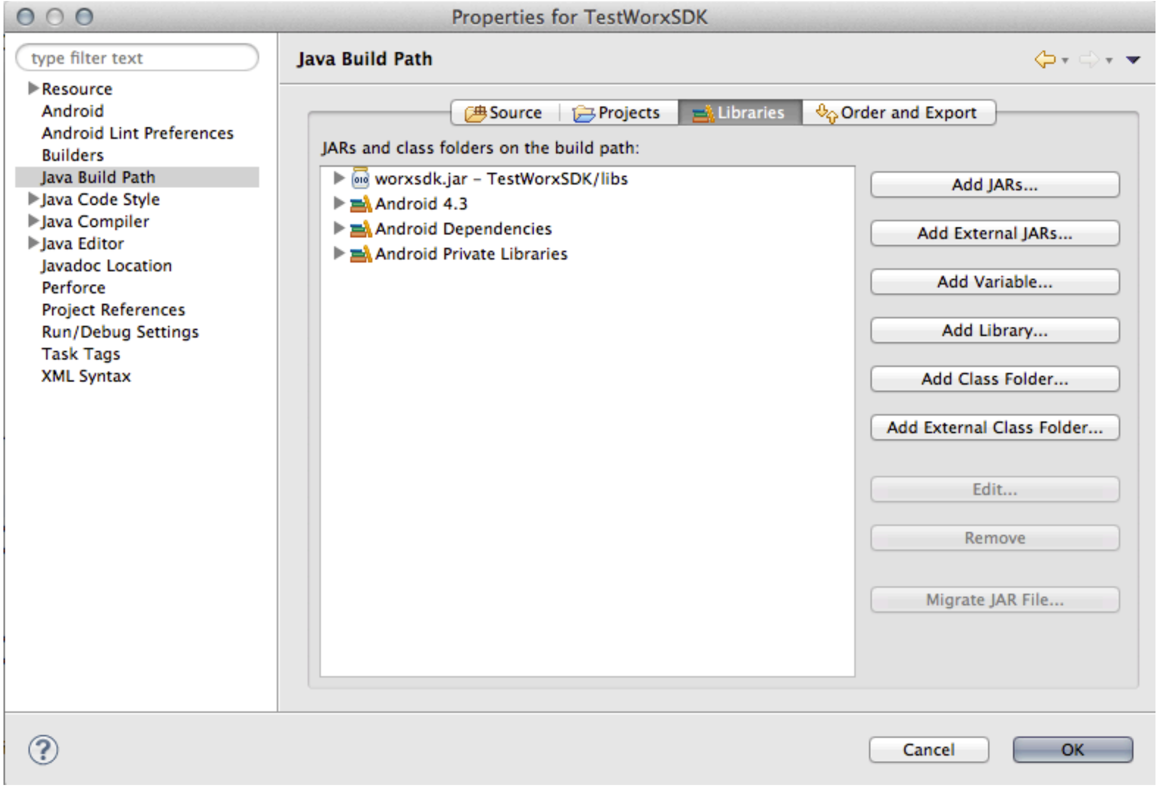


AndroidAPIExerciser
AsmDexTest
LoggingFrameworkTestApp
ManagedAppTest
mdxDiag
TestApp01
TestCMA
TestWorxSDK

AndroidManifest.xml
bin
gen
libs
proguard-project.txt
project.properties
res
src

armeabi
armeabi-v7a
WorxSDK.jar
x86

- Click the build properties for the application and add the WorxSDK as a library on your Java build path. You should then be able to use the APIs in the section below.



Properties for TestWorxSDK

Java Build Path

JARs and class folders on the build path:

worxsdk.jar – TestWorxSDK/libs
Android 4.3
Android Dependencies
Android Private Libraries

APIs definitions

There are several APIs that are exposed for use by the developer.

Class name - com.citrix.worx.sdk.MDXApplication

isManaged

public static boolean isManaged (Context context)

Returns **true** if the application is managed. This means the Citrix Worx Home app is installed on the device and it is providing policies to your application.

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 application has been wrapped with the MDXToolkit.

Parameters:

context – The Android context that is making this call

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

Class name - com.citrix.worx.sdk.MDXPolicies

getPoliciesXML

public static String getPoliciesXML (Context context)

Returns a `String` that contains the policy XML blob currently applied to the application by WorxHome. Will return 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. Will return `null` if no value found.

Parameters:

context – The Android context that is making this call

policyName – The name of the policy to search for

Example: `String value = MDXPolicies.getPolicyValue(context, policyName);`

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. Returns a null `String`.

Parameters:

context – The Android context that is making this call
policyName – The policy name that will be monitored.
Messenger – The messenger that will receive messages when the policy value changes.

Example: `MDXPolicies.setPolicyChangeMessenger(context, policyName, messenger);`

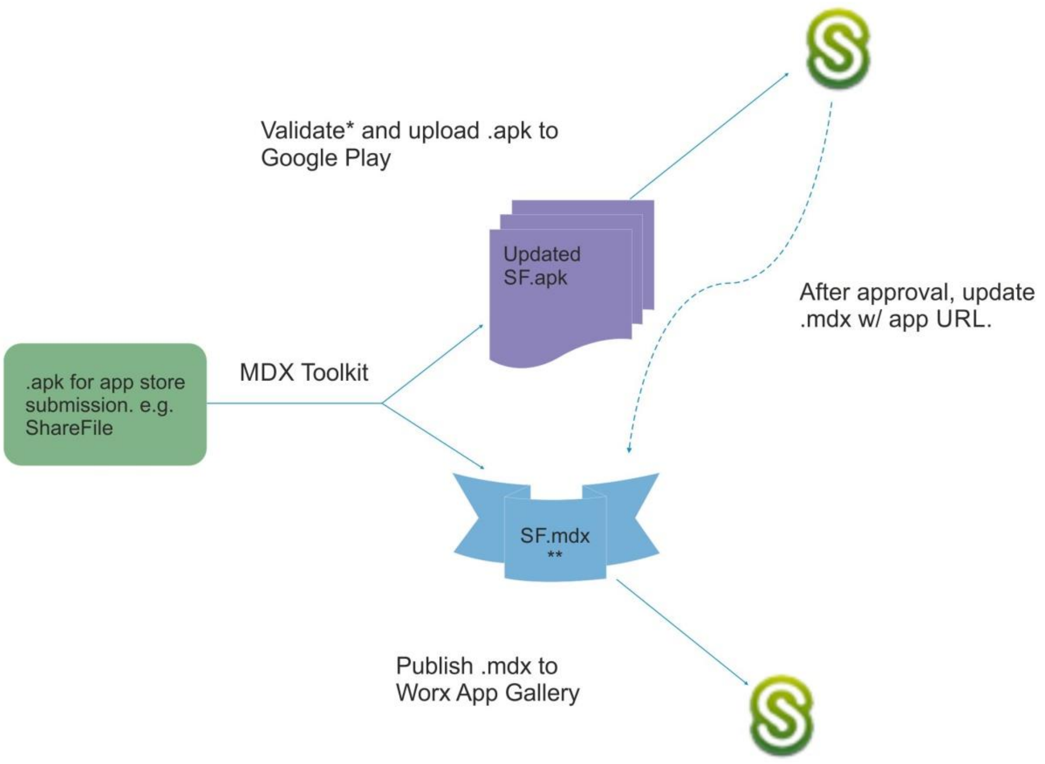
Libraries

The following table lists the libraries that may cause conflicts with similar libraries in the app. It is highly recommended to use the Citrix versions of the libraries to avoid conflicts.

Operating System	Libraries	Conflict status
Android	OpenSSL	Will have conflicts

Publishing Steps

The figure shows a high-level architectural view of the publishing process using ShareFile as an example:


Validate* and upload .apk to Google Play
Updated SF.apk
After approval, update .mdx w/ app URL.
.apk for app store submission. e.g. ShareFile
MDX Toolkit
SF.mdx **
Publish .mdx to Worx App Gallery

* Validation of .apk/.mdx is part of "Citrix Ready" program.

** The finalized .mdx should be submitted to Citrix App Gallery.

Once the WorxSDK has been added to your Android application, follow the steps below.

1. With the MDX Toolkit used to generate updated .apk and .mdx packages for Android, use the .apk (to be submitted to app store) as the input.

2. Install the updated .apk on device to verify all app functionalities are working correctly.

3. Test the MDX functionality with XenMobile setup.

4. Submit the original .apk to Citrix for validation and certification.

5. After validation, submit the generated .apk to the Google Play Store for approval.

6. After approval of the app, run the MDX Toolkit to update the app download URL in the .mdx file.

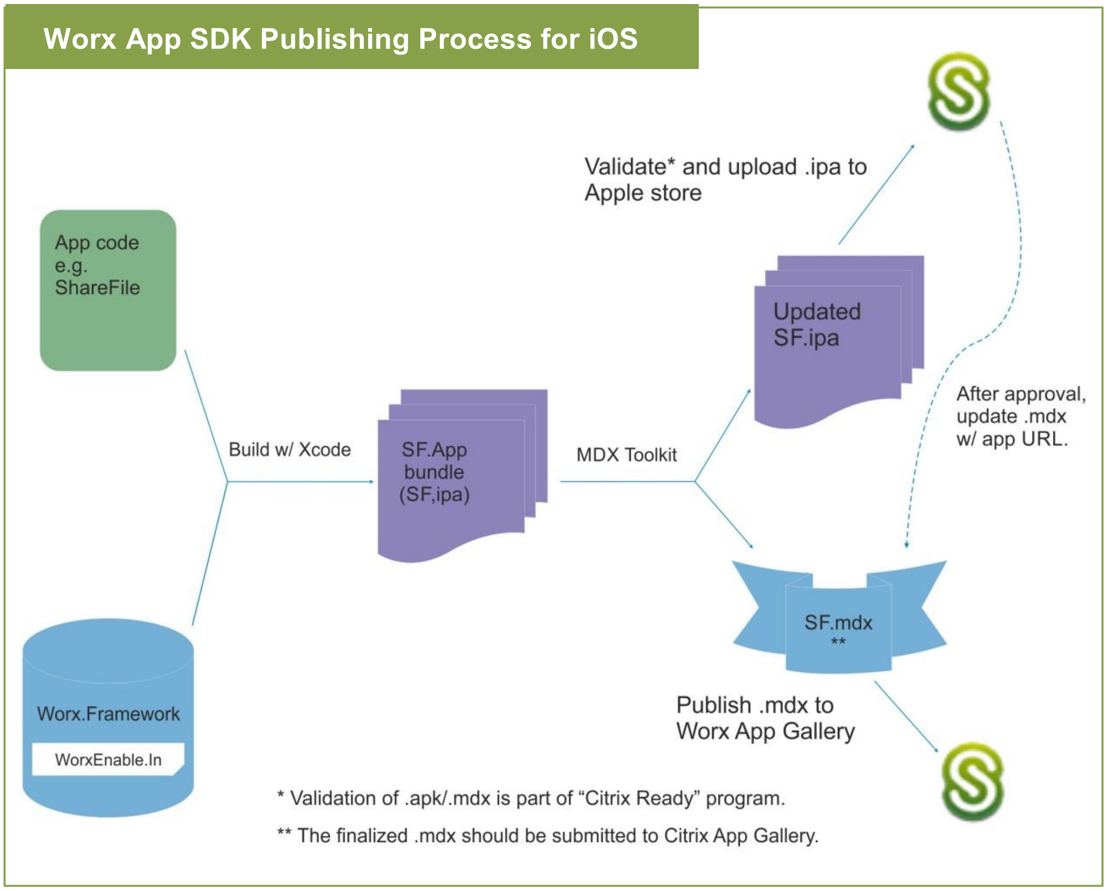
7. Submit the final .mdx to the Citrix Ready app catalog for wider distribution.

Worx SDK for iOS

Below are the instructions for adding the Worx SDK to your iOS application and publishing it to the Worx Gallery.

Publishing Steps

Below is a high-level architectural depiction of the publishing process using ShareFile as an example. The detailed steps can be found below the image.



Worx App SDK Publishing Process for iOS
App code e.g. ShareFile
Worx.Framework
WorxEnable.In
Build w/ Xcode
SF.App bundle (SF,ipa)
MDX Toolkit
Validate* and upload .ipa to Apple store
Updated SF.ipa
After approval, update .mdx w/ app URL.
SF.mdx **
Publish .mdx to Worx App Gallery
* Validation of .apk/.mdx is part of "Citrix Ready" program.
** The finalized .mdx should be submitted to Citrix App Gallery.

1. Build and test your app as is customary.

2. Install the Citrix MDX Toolkit.

The Worx App SDK files are added in the data/MDXSDK folder in the application folder – Worx.Framework, CitrixLogger.framework, default_polices.xml (default policy template file), and 3rd Party Libraries.

3. Integrate MDX into your app project by following these steps:

   a. Add the data/MDXSDK folder to the Apple Xcode project. To do so, you can drag the data/MDXSDK folder to the Xcode project.

b. Revise a line of code in the pre-compiled header file in the app project to import WorxEnable.h from Worx.framework as shown in the following example.

```
#ifdef     OBJC__
_
//import MDX extensions
#import <AVFoundation/AVFoundation.h>
#import <SystemConfiguration/SCNetworkReachability.h>
#import <Worx/WorxEnable.h>
#endif
```

c. Add the following to “other linker flags” if they do not appear:

- `–lsqlite3`
- `–ObjC`
- `–lxml2`
- `-u _FIPS_text_start`
- `-u _FIPS_text_end`
- `-u _FIPS_rodata_start`
- `-u _FIPS_rodata_end`

d. Add the following framework and libraries:

- AVFoundation.framework
- CFNetwork.framework
- CoreLocation.framework
- Libresolv.dylib
- Libz.dylib
- MessageUI.framework
- MobileCoreServices.framework
- QuickLook.framework
- Security.framework
- SystemConfiguration.framework
- CoreTelephony.framework
- Social.framework

e. Add “Run Script Build Phase” for the project with the following script:

```
/Applications/Citrix/MDXToolkit/incore_macho --debug -exe
"$CONFIGURATION_BUILD_DIR/$EXECUTABLE_PATH"
```

Replace the path of the MDXToolkit with the proper path as needed.

f. For XCODE 5 and above change the default flag from Yes to No for Strip Linked Product. This setting can be found in build setting of the project.

g. Compile the project and generate the app binaries.

4. Compile and archive the project to generate the app bundle that contains the embedded Worx framework, such as the .ipa package.

5. Wrap the application as an ISV app as shown in a previous section.

6. Install and run the generated .ipa app bundle to test its functionality.

7. Test the functionality of the app using a XenMobile environment.

8. Submit the following files to Citrix for verification:

- Original .ipa
- .ipa file with the embedded Worx framework
- Modified .ipa and generated .mdx file created after wrapping

9. After validation, submit the generated .ipa to the Apple App Store with the Application Loader tool for approval.

10. After Apple approves the app, run the .mdx file through the MDX Toolkit to update the App Store URL.

11. Submit the App Store URL of the final .mdx file to the Citrix Ready Worx App Gallery for distribution.

Public APIs

```
typedef enum {
 MDX_APPMODE_MDXSpecific, //MDX Premium App Mode
 MDX_APPMODE_GeneralAppStore //General App Store App Mode
} MDX_APPMODE;

@interface MDXSDK : NSObject

///// Set App Mode from one of the modes mentioned in the MDX_APPMODE
enum
+(BOOL) CTXMDX_SetAppMode:(MDX_APPMODE)appMode;

///// Get policies values set by admin (managed mode) or default policy
template
///// (unmanaged mode). ISV can use this API to pre-configure
application settings
/////
+(NSString*) CTXMDX_GetValueOfPolicy:(NSString*)policyName
DefaultValue:(NSString*)defaultValue;

////// Check if MDX-Manager(WorxHome) is installed
+(BOOL) CTXMDX_IsMDXAccessManagerInstalled;

@end
```

Libraries

The following table lists the libraries that might cause conflicts with similar libraries in the app. It is highly recommended to use the Citrix versions of the libraries to avoid conflicts.

Operating System	Libraries	Conflict status
iOS	OpenSSL 1.0.1H	No major modification as is with FIPS support. Will have conflicts.
iOS	Thrift	No major changes but some improvements done internally. Will have conflicts.

Creating Custom Policies

ISVs that want to create their own customized set of policies may do so by adding new policies to the default_sdk_policies.xml or policy_metadata.xml file in the MDX Toolkit using the formats below. The bolded text can be modified for a specific purpose.

String

```
    <Policy>
        <PolicyName>Sample_Policy</PolicyName>
        <PolicyType>string</PolicyType>
        <PolicyCategory>Required_Category</PolicyCategory>
        <PolicyDefault>My Sample String</PolicyDefault>
        <PolicyStrings>
             <Title res_id="Policy Title”>Sample String Policy</Title>
            <Description res_id="SAMPLE_POLICY_DESC">
                    Please enter the sample policy value
             </Description>
        </PolicyStrings>
    </Policy>
```

Boolean

```
    <Policy>
        <PolicyName>Sample_Policy</PolicyName>
        <PolicyType>string</PolicyType>
        <PolicyCategory>Required_Category</PolicyCategory>
        <PolicyDefault>false</PolicyDefault>
        <PolicyStrings>
            <Title res_id="Policy Title”>Sample Boolean Policy</Title>
            <BooleanTrueLabel res_id="POLICY_ON">On</BooleanTrueLabel>
            <BooleanFalseLabel res_id="POLICY_OFF">Off</BooleanFalseLabel>
            <Description res_id="SAMPLE_POLICY_DESC">
                    If On, the app will do something If Off, the app not allowed to do
                    something.

                    Default value is Off.
             </Description>
        </PolicyStrings>
    </Policy>
```

Enum

```
    <Policy>
        <PolicyName>Sample_Policy</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>Required_Category</PolicyCategory>
    <PolicyDefault>Value1</PolicyDefault>
    <PolicyStrings>
        <Title res_id="Sample Policy Title”>Sample Enum Policy</Title>
        <Description res_id="SAMPLE_POLICY_DESC">
                Sample policy description.

                Default value is Yes.
         </Description>
    </PolicyStrings>
</Policy>
```

Once you have added all your desired policies, you may then wrap the app while specifying the location of the modified policy XML file. More information on this process is located in the MDX Policies section.

FAQ

Q: Do I need to install Java 7 to run the tool?
A: Yes, XenMobile 9.0 requires the use of Java JDK 1.7. Java JDK 1.8 is not officially supported.

Q: Do I need to rewrap my apps even though I've done this before?
A: Yes, the XenMobile software changes significantly between releases. You must rewrap the original application each time a new toolkit comes out.

Q: Can I wrap a previously wrapped APK or IPA file?
A: No, Citrix do not support rewrapping SDK applications with modified APK or IPA files. You will need to wrap the original application file.

Q: Can I use the wrapping toolkit on Windows?
A: Citrix does not currently support the toolkit on Windows. However, Android applications can be wrapped on Windows using the Android command-line tool. You must first copy both ManagedApp.jar and ManagedAppUtility.jar to a directory on your Windows computer. Also make sure the Android wrapping prerequisites have been met. You may then wrap using the command line options explained earlier in this document.

Q: Why did the wrapping process fail?
A: The log file will contain the information and progress from wrapping. Check these logs for error messages and warnings. Check to see if the error is listed in the Known Issues/Limitations section below. If not, please list the steps you took to encounter the issue, collect the log from the failed wrap attempt, as well as any additional information you have and send that to the sales or escalation engineer who has been working with you. If that person is not available, send the information to Support@citrix.com.

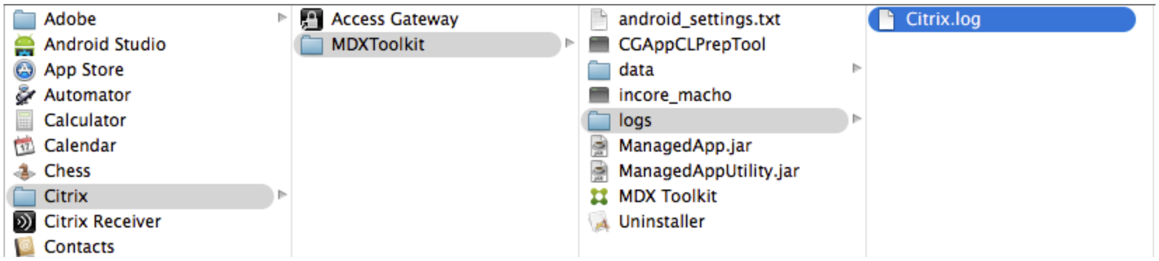
Q: Why won't my wrapped application run?
A: First check if the unwrapped application runs. If it doesn't, the application itself is the root problem. If the unwrapped application does run, please collect logs from Worx Home and send that to the sales or escalation engineer who has been working with you. If that person is not available, send the information to Support@citrix.com.

Q: Why isn't the Worx SDK working with my app?
A: First check that the SDK being used is from the same toolkit as the wrapper being used 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.jar is added to the project dependencies. If you experience additional problems, please reach out to Citrix Ready or Citrix Support.

Troubleshooting

Collecting wrapping logs

Navigate to the logs folder shown below (at location /Applications/Citrix/MDXToolkit/logs/). Logs can be viewed using the default Console application on Mac or any standard text editor.



Common issues and steps to take if they occur

General:

- If you are having problems running the app when it is wrapped.
    - Check to see if the unwrapped app has the same problem. If it does, this is an app specific issue and out of our control. If it does not have the same problem, collect the logs from Worx Home (document here) and email them to Support@citrix.com.

Android:

- If you can't install the wrapped application onto your device.
    - Ensure that you are using a valid keystore or provisioning profile/certificate pair.
- Collect command-line wrapping logs using the steps in the section above and send them to Support@citrix.com

iOS

- If there is an issue with the Key/Certificate
    - Request to reissue the certificate from the Mac's Keychain Access. This will generate a new private key. Then download the certificate and provisioning profile from the Apple developer website.
- Xcode command line tools missing.
    - For Mac OS X 10.9+ (Mavericks) - run the MDX Toolkit. This will kick off an automatic install of the command-line tools.
    - For Mac OSX 10.8 (Mountain Lion) - Install Xcode from the Apple App Store. Download the Command line tools from the Apple Developer page. After Command line tools are installed, run Xcode at least once to install any prerequisites. See iOS Prerequisites section for more details.

Known Issues/Limitations

Android:

- Icons are required in the application for the app to be wrapped.

iOS:

- Citrix detects and warns if XAMARIN developed apps are wrapped. These apps are currently not supported.
- Apps developed with cross-compilers are not supported.

Appendix

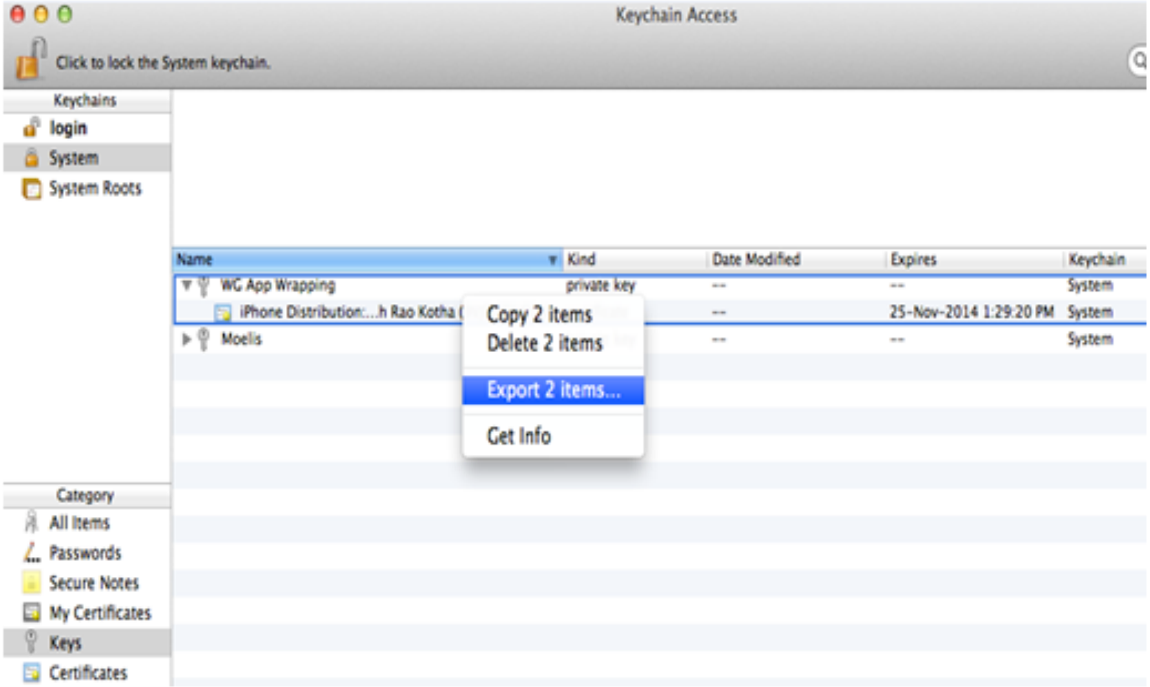
If you already have created a certificate and profile, you may transfer them from one Mac to another using the steps in the next section.

Importing a Certificate and Profile onto a New Computer

Export Existing Certificate

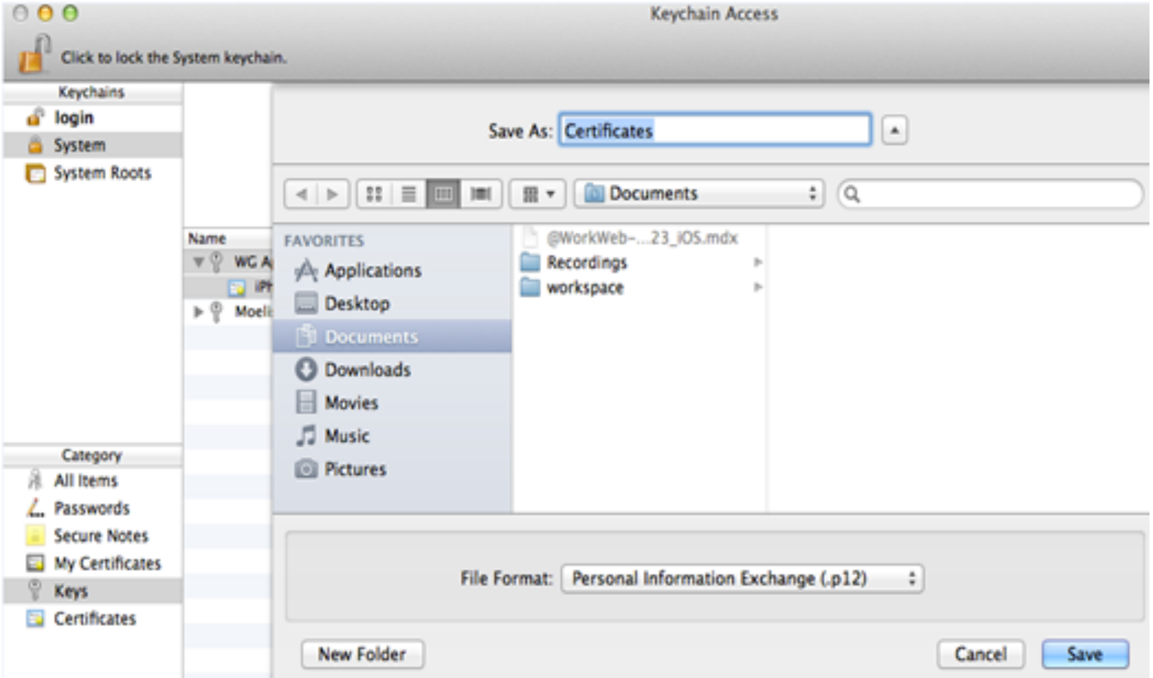
Open Key chain Access on the Mac where you have originally generated the certificate and login.

Select **Keys** from the left pane and expand the certificate that you want to export. After selecting the certificate and its key, right-click and then click **Export 2 Items**



Name	Kind	Date Modified	Expires	Keychain
WG App Wrapping	private key	--	--	System
iPhone Distribution:...h Rao Kotha (		--	25-Nov-2014 1:29:20 PM	System
Moelis		--	--	System

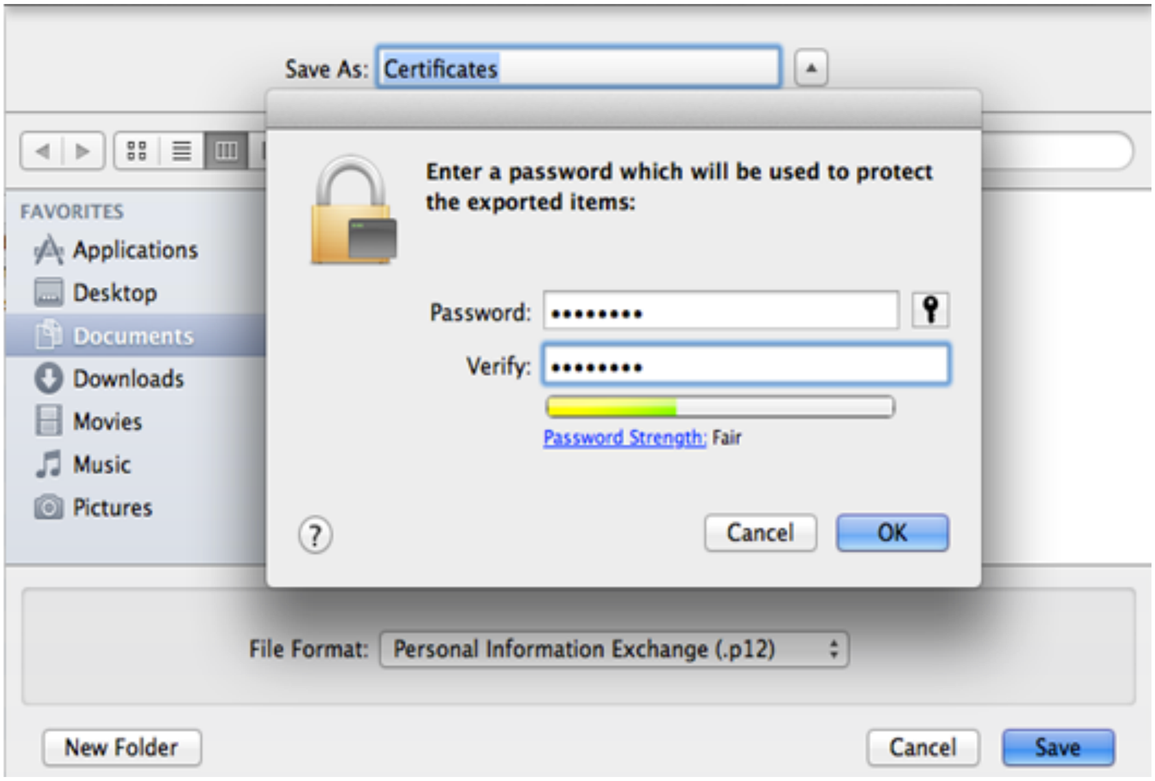
Select the location where you want to save the certificate and then select the file format as .p12



Save As: Certificates

File Format: Personal Information Exchange (.p12)

Enter a password that will be used to protect the exported certificate.



Save As: Certificates

Enter a password which will be used to protect the exported items:

Password: ••••••••

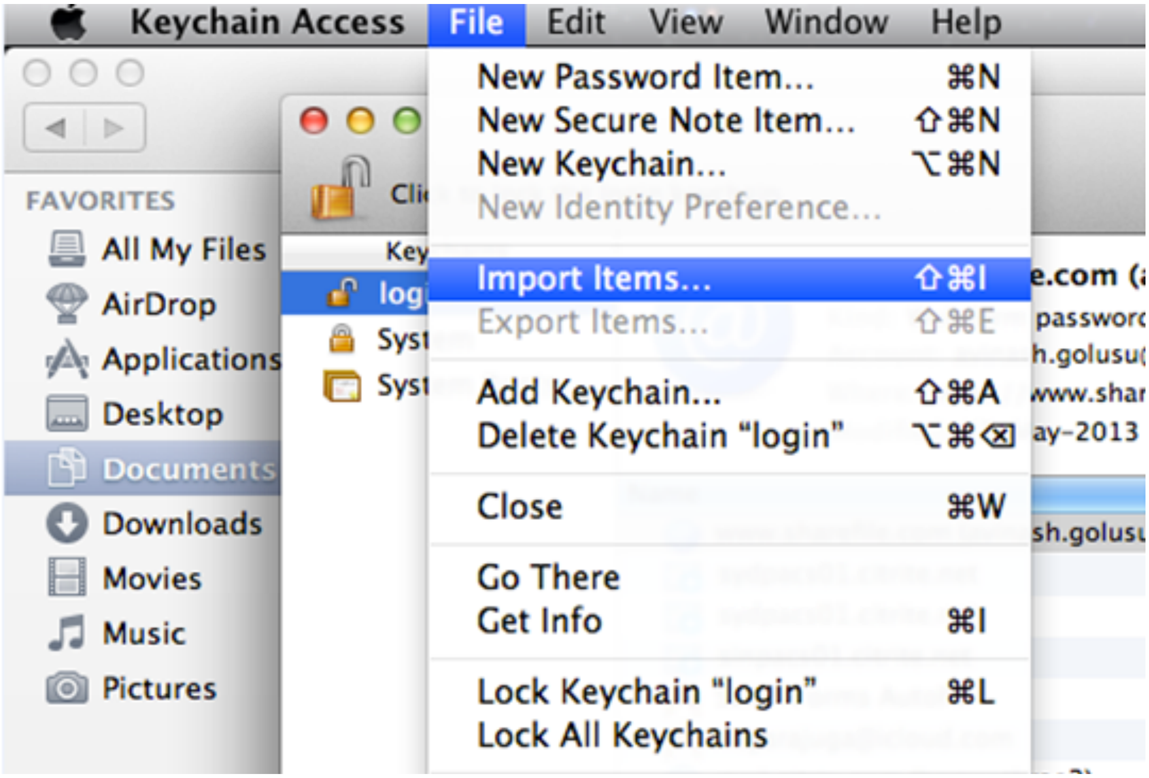
Verify: ••••••••

Password Strength: Fair

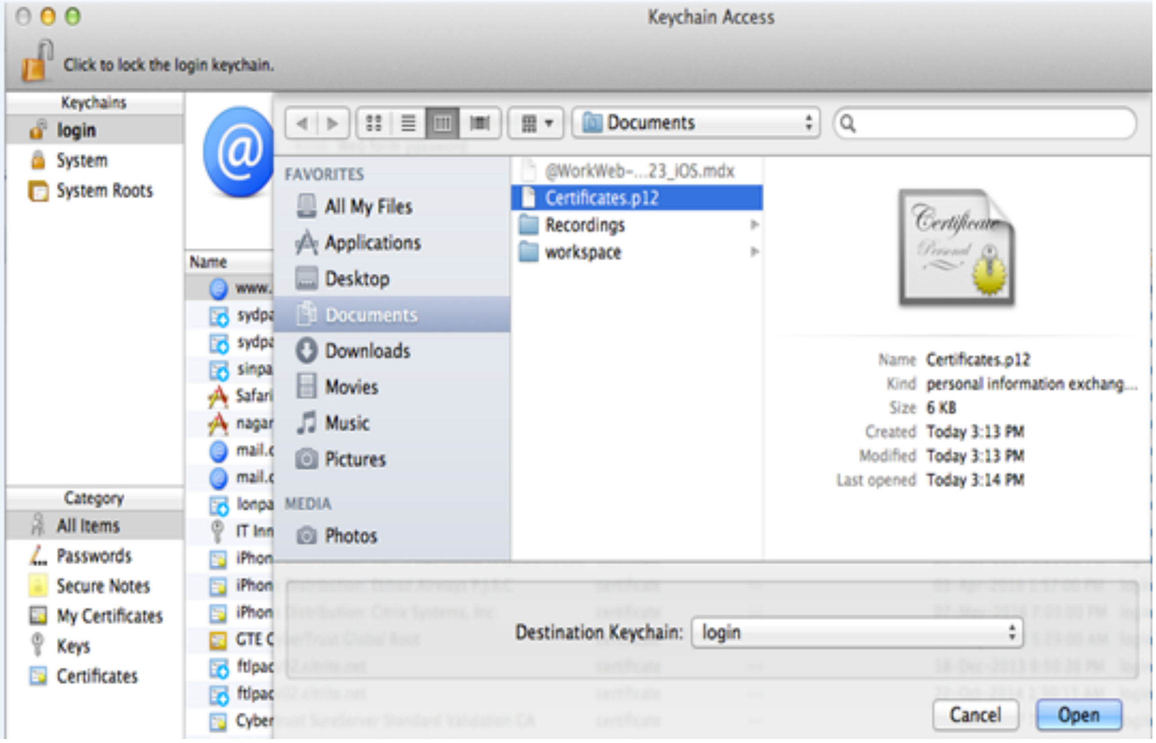
File Format: Personal Information Exchange (.p12)

Import Existing Certificate

Connect to the computer you want to import the certificate to. Login to Keychain access and click on **Import Items.**



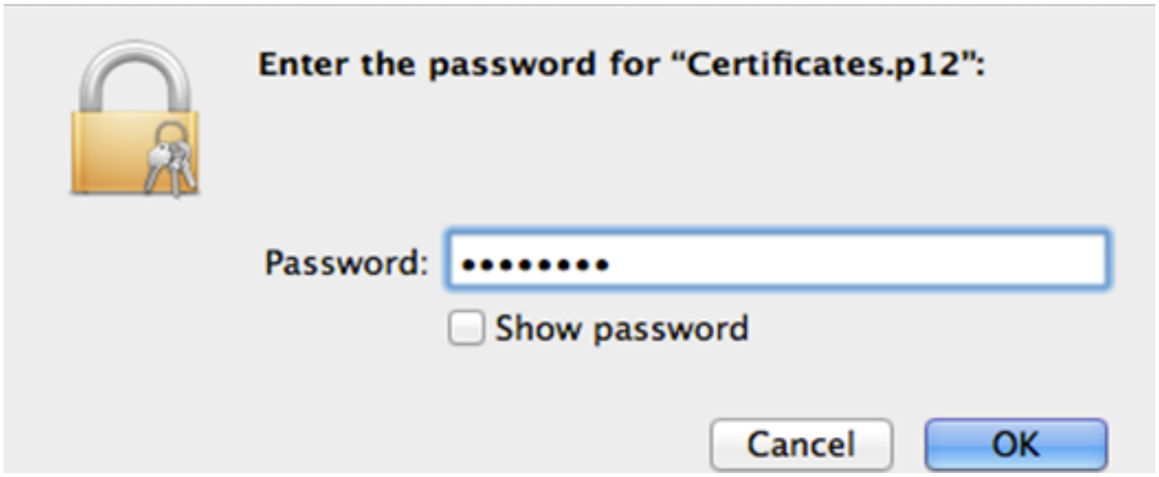
Navigate to the location where you saved the .p12 format certificate and then click **Open**.



Name Certificates.p12
Kind personal information exchang...
Size 6 KB
Created Today 3:13 PM
Modified Today 3:13 PM
Last opened Today 3:14 PM

Destination Keychain: login

Provide the password that you entered when exporting the certificate.

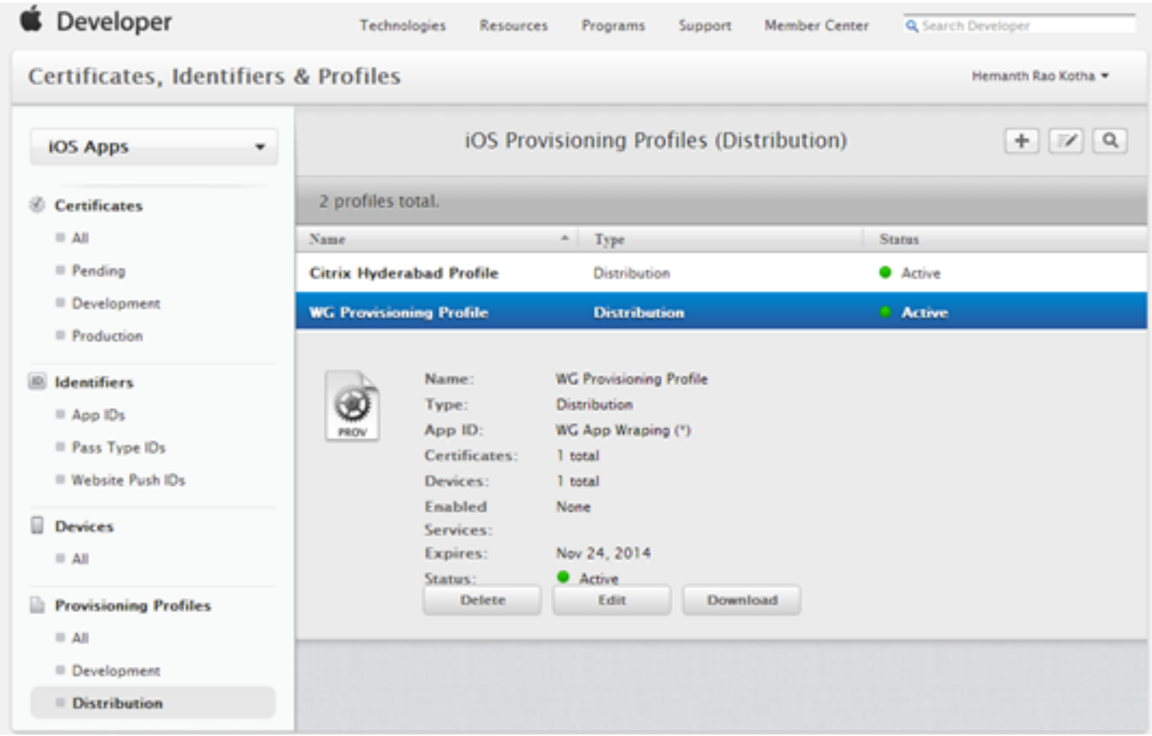


Enter the password for "Certificates.p12":

Password: ••••••••

□ Show password

Log on to the corresponding Apple developer account and download the provisioning profiles associated with the certificate.



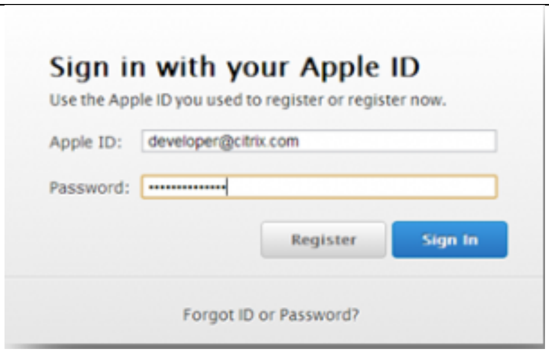
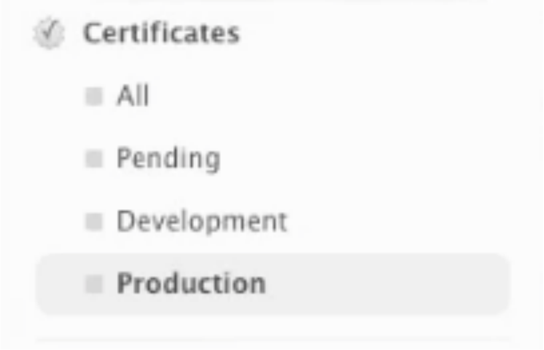
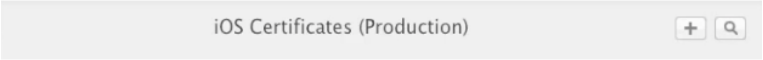
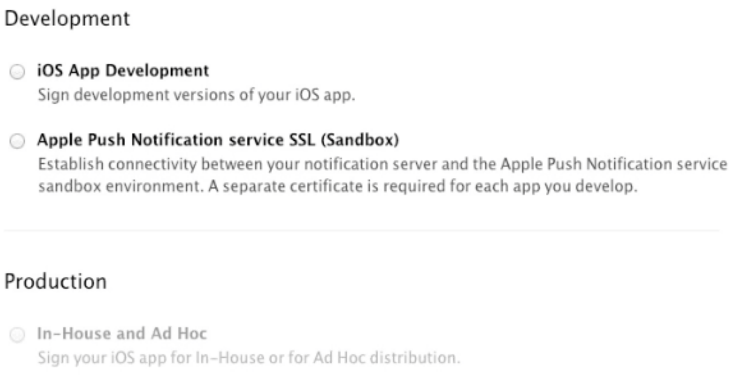
iOS Provisioning Profiles (Distribution)

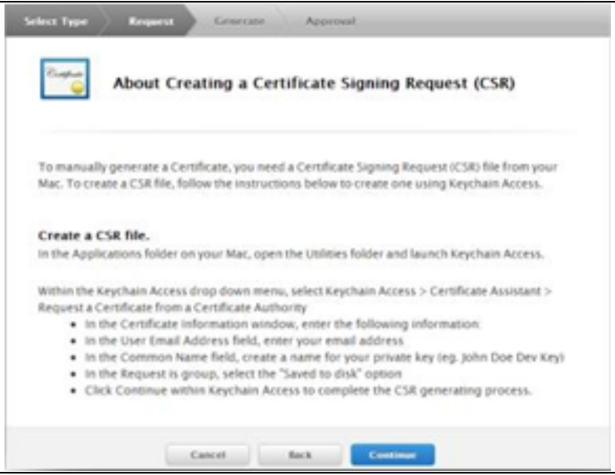
2 profiles total.

Name	Type	Status
Citrix Hyderabad Profile	Distribution	● Active
WG Provisioning Profile	Distribution	● Active

Name:	WG Provisioning Profile
Type:	Distribution
App ID:	WG App Wraping (*)
Certificates:	1 total
Devices:	1 total
Enabled Services:	None
Expires:	Nov 24, 2014
Status:	● Active

Creating an Apple Enterprise Provisioning Profile

1.	Log on to your account at the following link: https://developer.apple.com/account	 **Sign in with your Apple ID** Use the Apple ID you used to register or register now. Apple ID: developer@citrix.com Password: ••••••••••••••
2.	If this is the first time you are logging on to the portal, you will have to request a certificate. Click the **Production** button under the **Certificates** tab in the left pane.	
3.	Click the Plus Sign **(+)** in the upper-right corner of the page.	
4.	Select the **In-House and Ad Hoc** button under **Production** and then click **Continue** at the bottom of the page.	 Development ○ **iOS App Development** Sign development versions of your iOS app. ○ **Apple Push Notification service SSL (Sandbox)** Establish connectivity between your notification server and the Apple Push Notification service sandbox environment. A separate certificate is required for each app you develop. Production ○ In-House and Ad Hoc Sign your iOS app for In-House or for Ad Hoc distribution.

<table>
<tr><td>5.</td><td>Click <b>Continue</b>.</td><td>

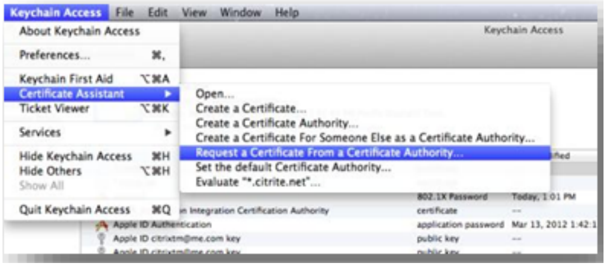
About Creating a Certificate Signing Request (CSR)

To manually generate a Certificate, you need a Certificate Signing Request (CSR) file from your Mac. To create a CSR file, follow the instructions below to create one using Keychain Access.

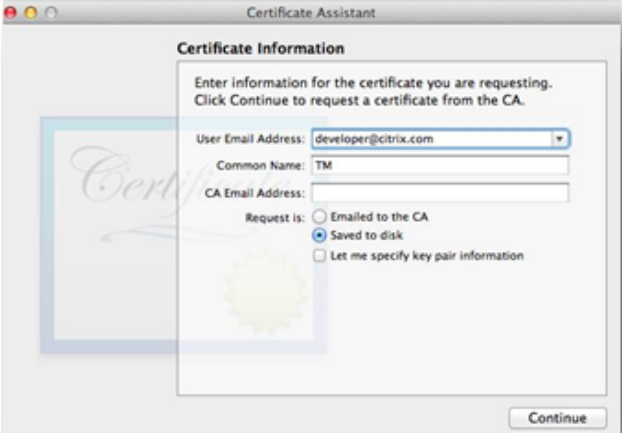
Create a CSR file.
In the Applications folder on your Mac, open the Utilities folder and launch Keychain Access.

Within the Keychain Access drop down menu, select Keychain Access > Certificate Assistant > Request a Certificate from a Certificate Authority
- In the Certificate Information window, enter the following information:
- In the User Email Address field, enter your email address
- In the Common Name field, create a name for your private key (eg. John Doe Dev Key)
- In the Request is group, select the "Saved to disk" option
- Click Continue within Keychain Access to complete the CSR generating process.

</td></tr>
<tr><td>6.</td><td>Next, open <b>Keychain Access</b> on your Mac computer.

Choose Keychain Access > Certificate Assistant > Request a Certificate from a Certificate Authority.</td><td>
</td></tr>
<tr><td>7.</td><td>Enter your email address in the <b>User Email Address</b> field.

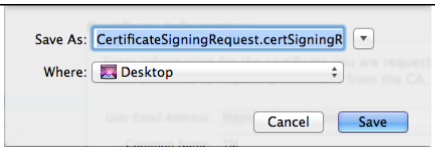
Enter your name in the <b>Common Name</b> field.

Select the <b>Saved to disk</b> option and then click <b>Continue.</b></td><td>

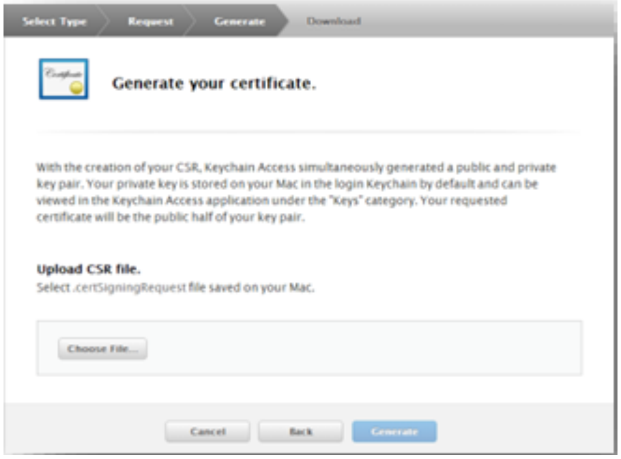
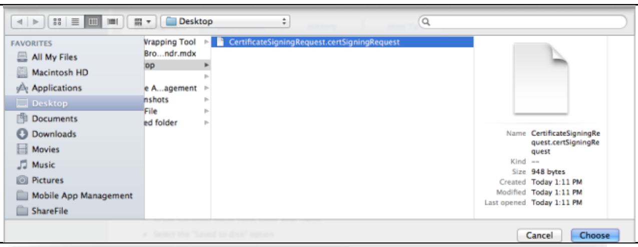
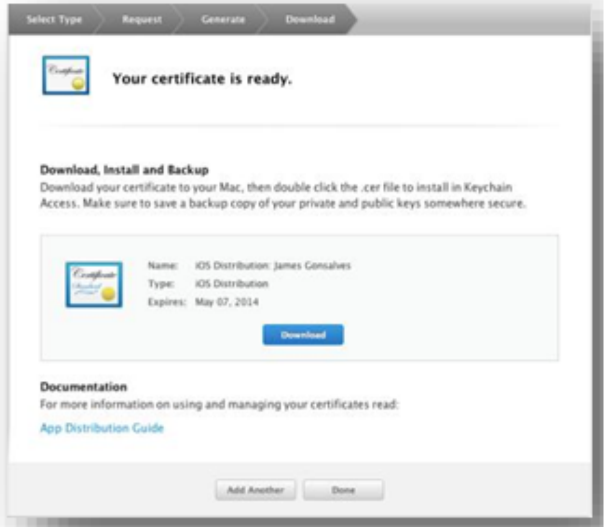
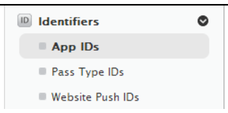
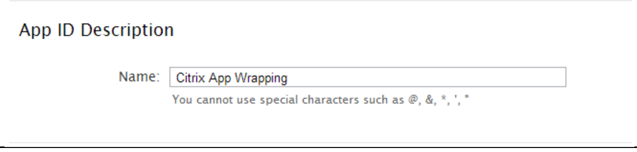
Certificate Information

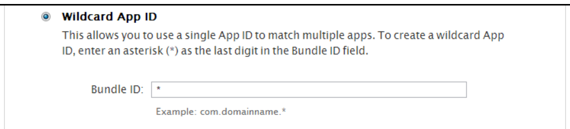
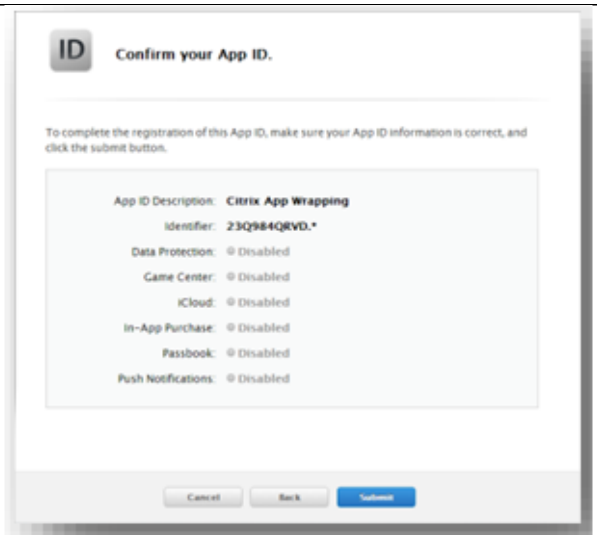
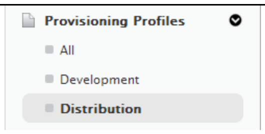
Enter information for the certificate you are requesting.
Click Continue to request a certificate from the CA.

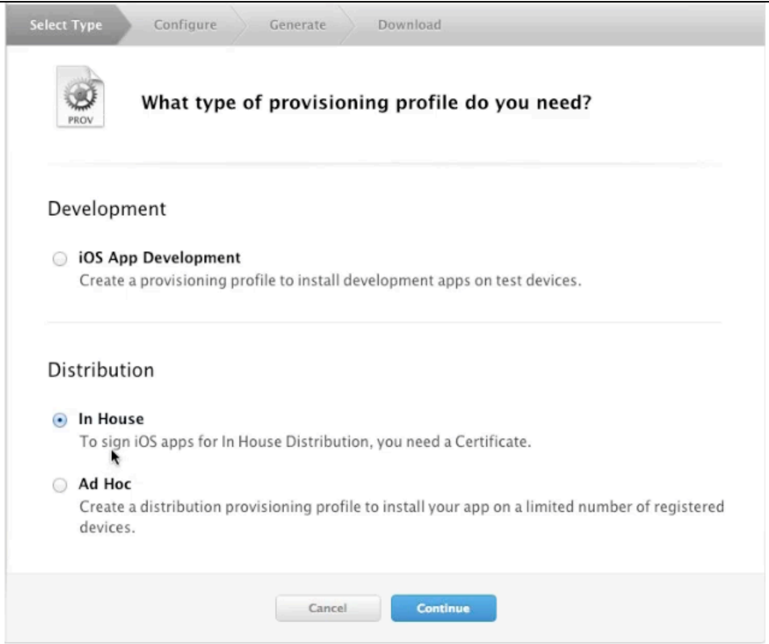
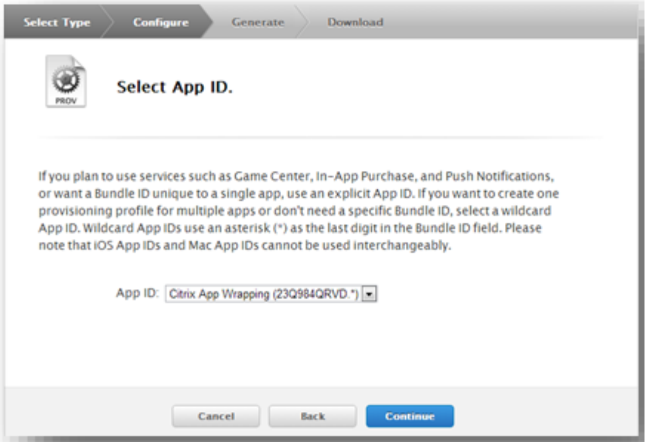
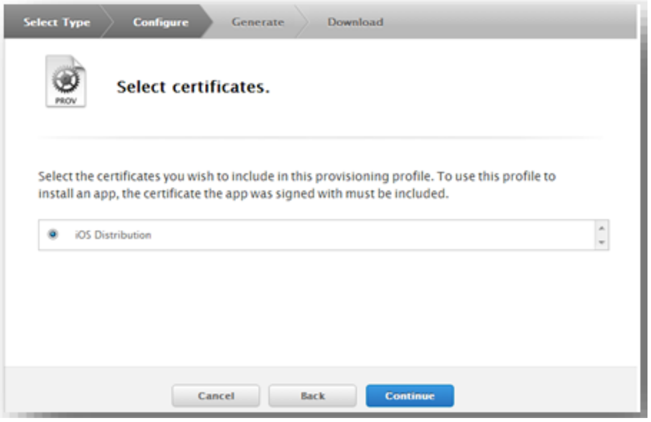
User Email Address: developer@citrix.com
Common Name: TM
CA Email Address:
Request is: ○ Emailed to the CA
● Saved to disk
□ Let me specify key pair information

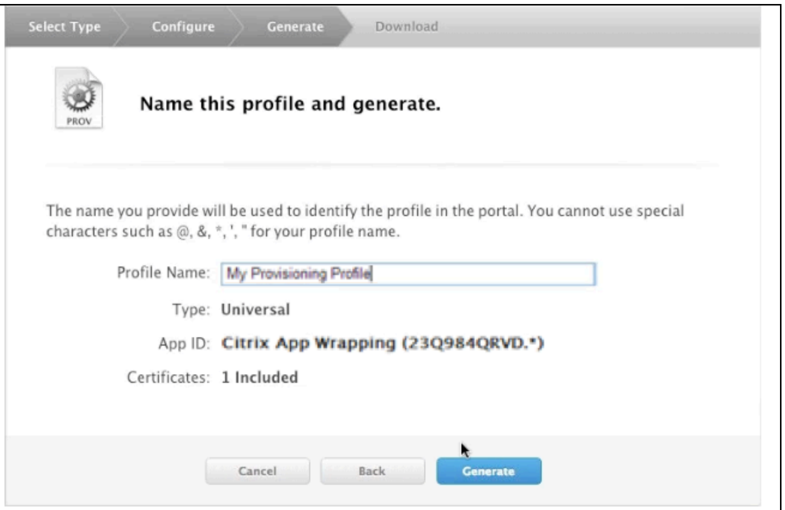
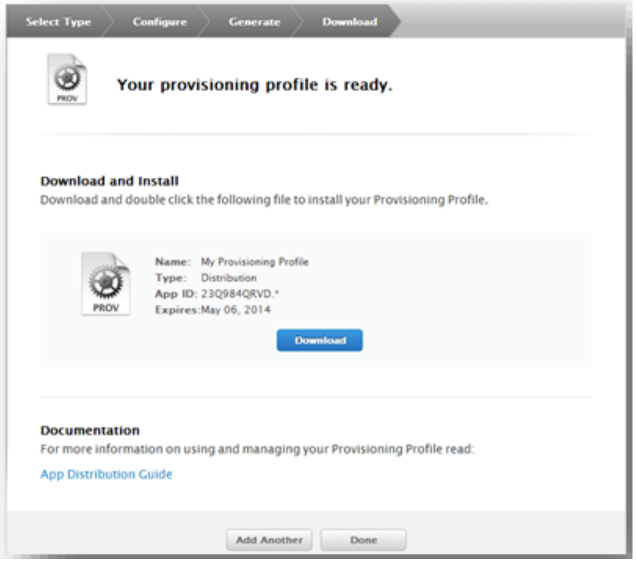
</td></tr>
<tr><td>8.</td><td>Save the Certificate Signing Request to a convenient folder on your computer.</td><td>
Save As: CertificateSigningRequest.certSigningR
Where: Desktop

</td></tr>
</table>

9.	Navigate back to the developer portal and then click **Choose File.**	 Generate your certificate. With the creation of your CSR, Keychain Access simultaneously generated a public and private key pair. Your private key is stored on your Mac in the login Keychain by default and can be viewed in the Keychain Access application under the "Keys" category. Your requested certificate will be the public half of your key pair. Upload CSR file. Select .certSigningRequest file saved on your Mac.
10.	Select the Certificate Signing Request file and click **Choose.** Click **Generate**.	 Desktop FAVORITES All My Files Macintosh HD Applications Desktop Documents Downloads Movies Music Pictures Mobile App Management ShareFile Vrapping Tool Bro...ndr.mdx op e A...agement nshots File ed folder CertificateSigningRequest.certSigningRequest Name CertificateSigningRequest.certSigningRequest Kind -- Size 948 bytes Created Today 1:11 PM Modified Today 1:11 PM Last opened Today 1:11 PM
11.	The certificate will appear in the portal. Download the certificate and install it on your Mac.	 Your certificate is ready. Download, Install and Backup Download your certificate to your Mac, then double click the .cer file to install in Keychain Access. Make sure to save a backup copy of your private and public keys somewhere secure. Name: iOS Distribution: James Gonsalves Type: iOS Distribution Expires: May 07, 2014 Documentation For more information on using and managing your certificates read: App Distribution Guide
12.	Click **Identifiers** and then **App IDs** in the left pane. Click the Plus Sign **(+)** in the upper-right corner of the page.	
13.	Provide an **App ID Description**.	 App ID Description Name: Citrix App Wrapping You cannot use special characters such as @, &, *, ', "

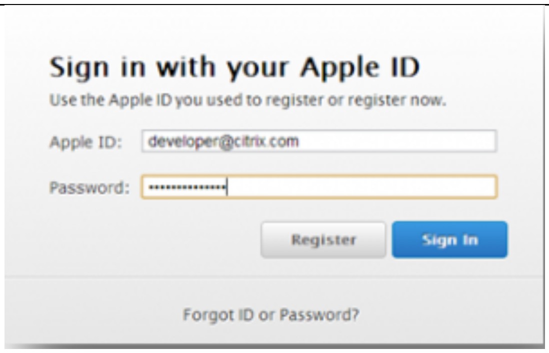
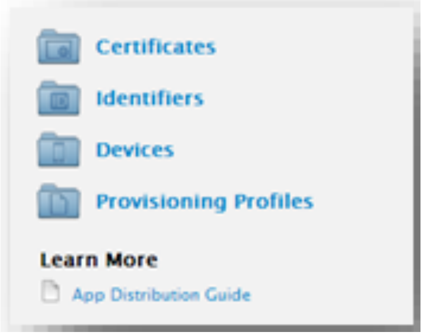
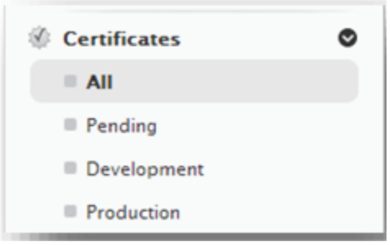
14.	If you're going to prep apps that you have not developed such as the Citrix @Work apps, select the **Wildcard App ID** radio button and enter an asterisk (*****) in the **Bundle Identifier** field. If you're going to prep apps that you own, select the **Explicit App ID** button and enter a unique App ID for each of your apps in the following format com.domainname.appname **Note**: Make sure that the **Data Protection** setting is **disabled**. You will not be able to use background sync for emails with this enabled. Click **Continue**.	 ● **Wildcard App ID** This allows you to use a single App ID to match multiple apps. To create a wildcard App ID, enter an asterisk (*) as the last digit in the Bundle ID field. Bundle ID: * Example: com.domainname.*
15.	Verify the information you entered and then click **Submit.** Click **Done** on the next page.	 Confirm your App ID. To complete the registration of this App ID, make sure your App ID information is correct, and click the submit button. App ID Description: Citrix App Wrapping Identifier: 23Q984QRVD.* Data Protection: Disabled Game Center: Disabled iCloud: Disabled In-App Purchase: Disabled Passbook: Disabled Push Notifications: Disabled
16.	Click **Provisioning** in the left pane, click **Distribution**, and then click the Plus Sign (**+)** in the upper-right corner of the page.	

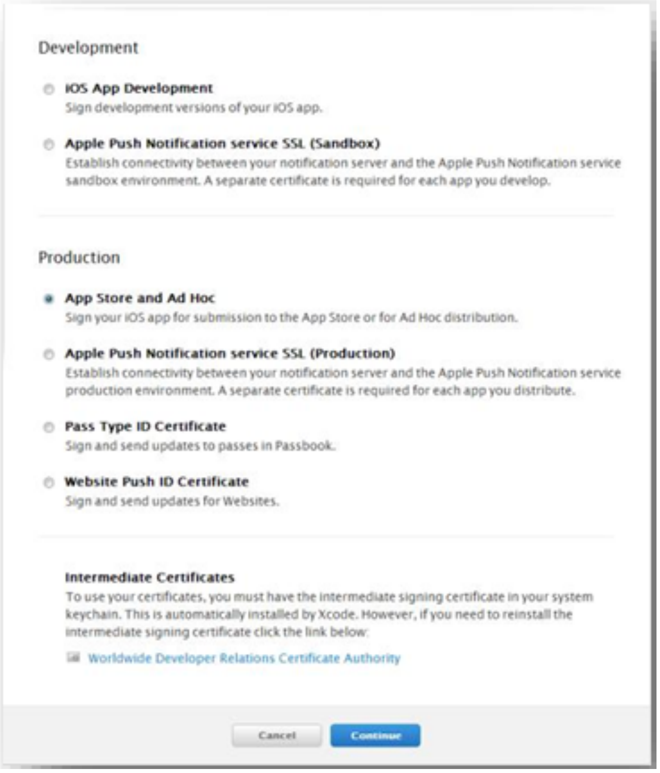
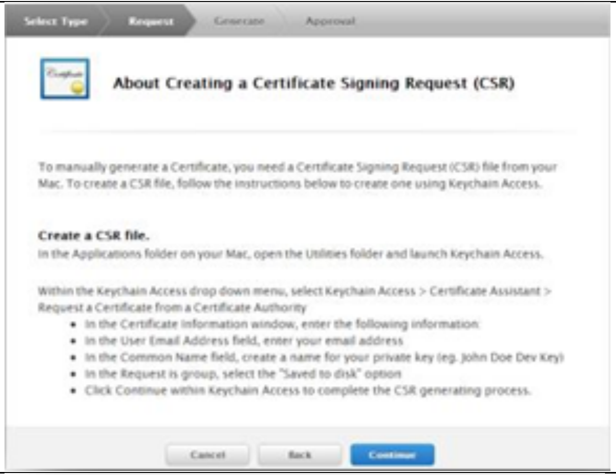
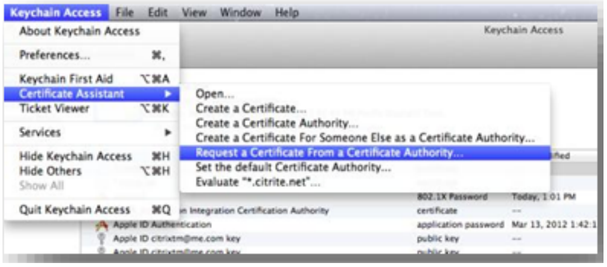
17.	Click the **In-House** button and then click **Continue.**	 What type of provisioning profile do you need? Development ○ iOS App Development Create a provisioning profile to install development apps on test devices. Distribution ● In House To sign iOS apps for In House Distribution, you need a Certificate. ○ Ad Hoc Create a distribution provisioning profile to install your app on a limited number of registered devices.
18.	Select the **App ID** you created.	 Select App ID. If you plan to use services such as Game Center, In-App Purchase, and Push Notifications, or want a Bundle ID unique to a single app, use an explicit App ID. If you want to create one provisioning profile for multiple apps or don't need a specific Bundle ID, select a wildcard App ID. Wildcard App IDs use an asterisk (*) as the last digit in the Bundle ID field. Please note that iOS App IDs and Mac App IDs cannot be used interchangeably. App ID: Citrix App Wrapping (23Q984QRVD.*)
19.	Select the certificate that you created.	 Select Type Configure Generate Download PROV Select certificates. Select the certificates you wish to include in this provisioning profile. To use this profile to install an app, the certificate the app was signed with must be included. ● iOS Distribution Cancel Back Continue

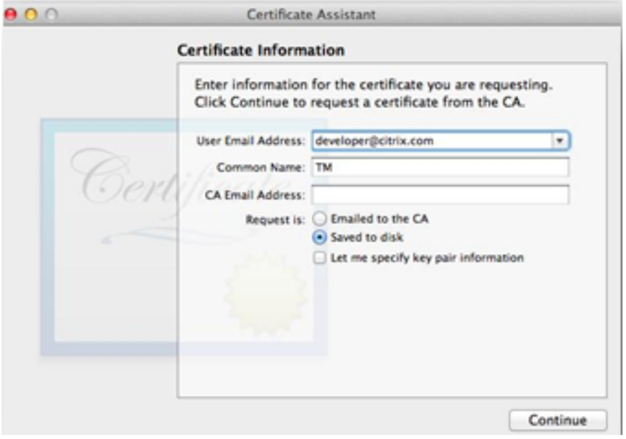
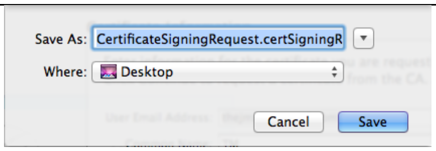
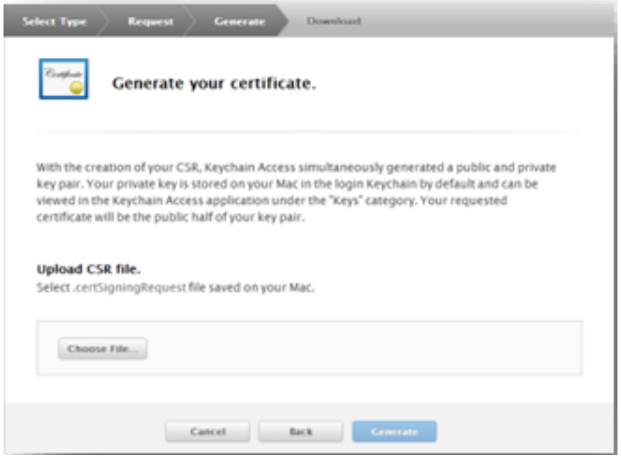
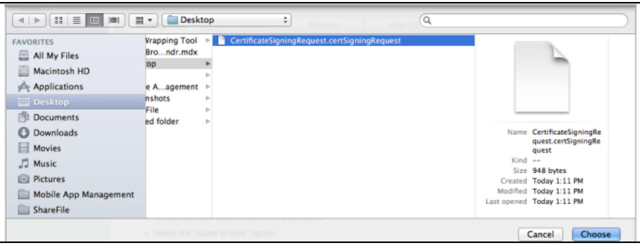
20.	Name the provisioning profile and then click **Generate.**	 Select Type Configure Generate Download PROV Cancel Back Generate **Name this profile and generate.** The name you provide will be used to identify the profile in the portal. You cannot use special characters such as @, &, *, ', " for your profile name. Profile Name: My Provisioning Profile Type: Universal App ID: Citrix App Wrapping (23Q984QRVD.*) Certificates: 1 Included
21.	Click **Download** and save the provisioning profile on your Mac.	 Select Type Configure Generate Download PROV Download Add Another Done **Your provisioning profile is ready.** **Download and Install** Download and double click the following file to install your Provisioning Profile. Name: My Provisioning Profile Type: Distribution App ID: 23Q984QRVD.* Expires:May 06, 2014 **Documentation** For more information on using and managing your Provisioning Profile read: App Distribution Guide

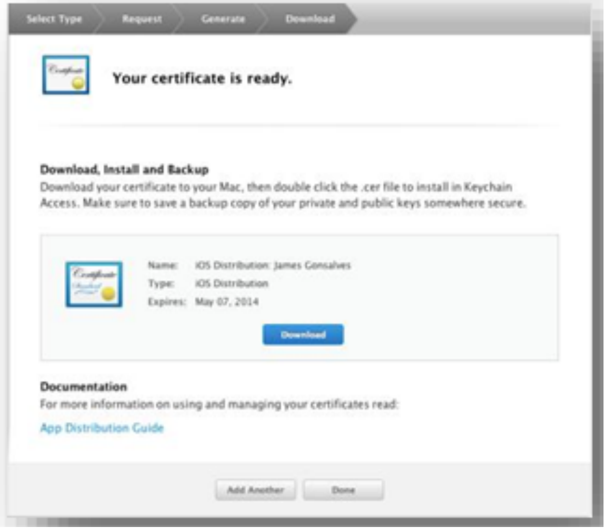
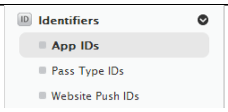
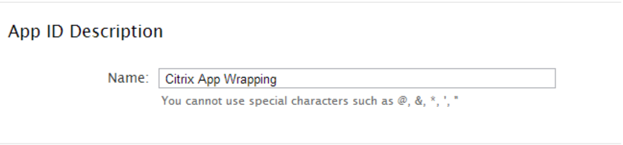
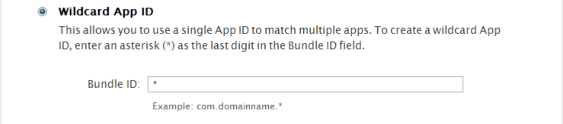
Make sure both the private key and the certificate are installed on your Mac Keychain Access.

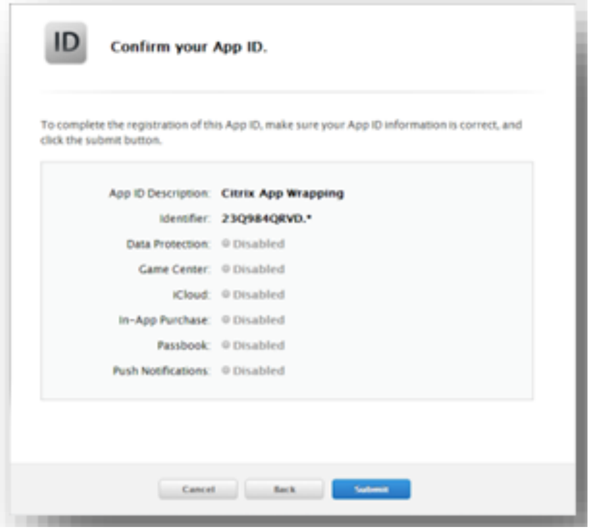
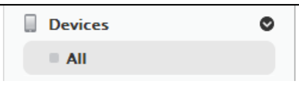
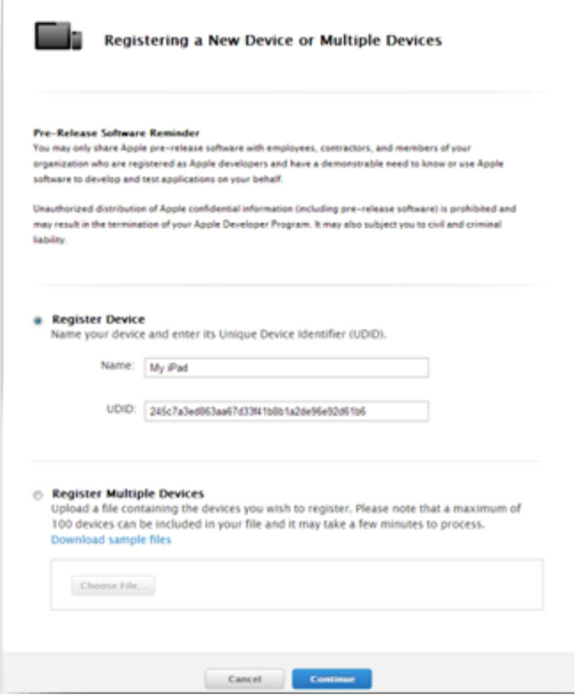
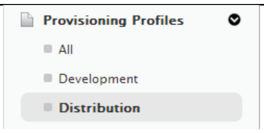
Creating an Apple Developer Provisioning Profile

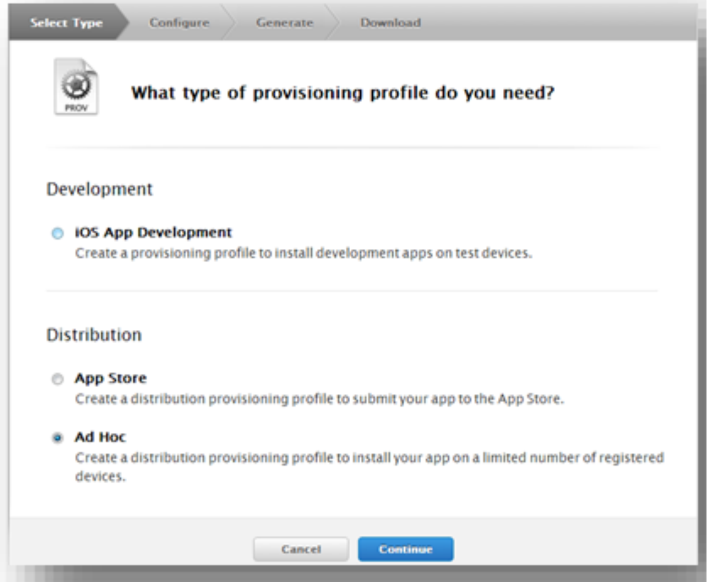
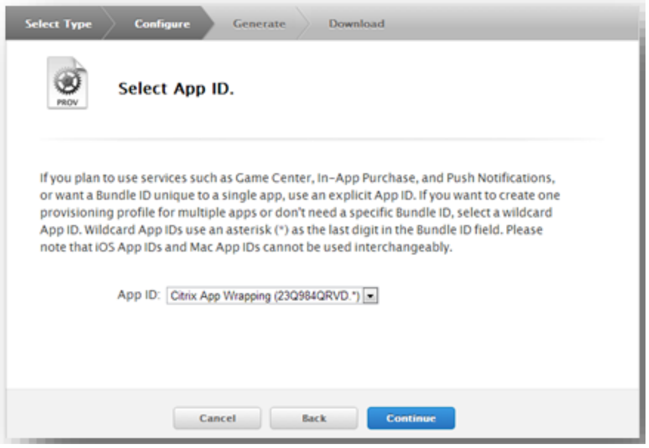
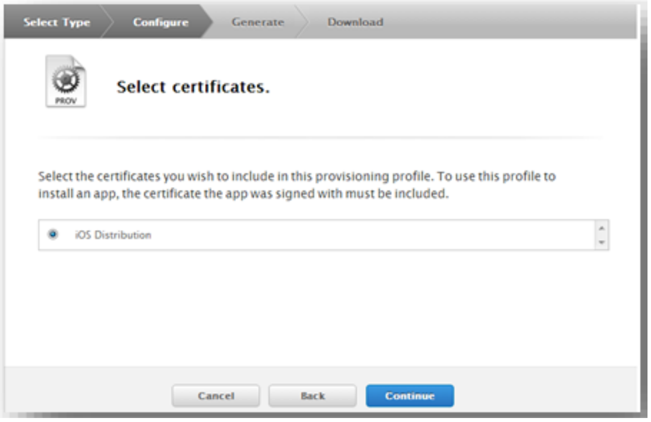
1.	Log on to your account at the following link: https://developer.apple.com/account	 Sign in with your Apple ID Use the Apple ID you used to register or register now. Apple ID: developer@citrix.com Password: ••••••••••••••
2.	If this is the first time you are logging on to the portal, you will have to request a certificate. Click **Certificates** in the left pane.	 Certificates Identifiers Devices Provisioning Profiles Learn More App Distribution Guide
3.	Click **Production** in the left pane and then click the Plus Sign **(+)** in the upper-right corner of the page.	

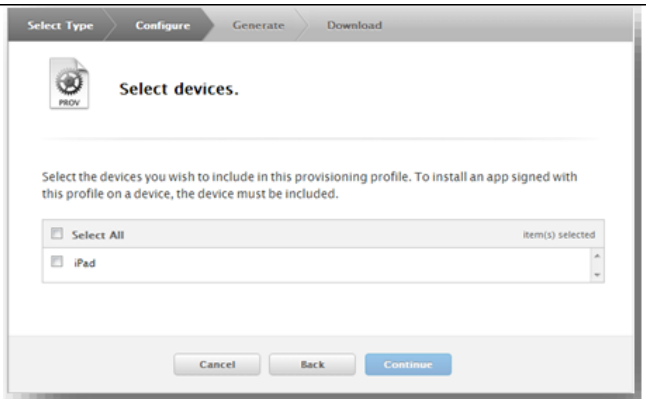
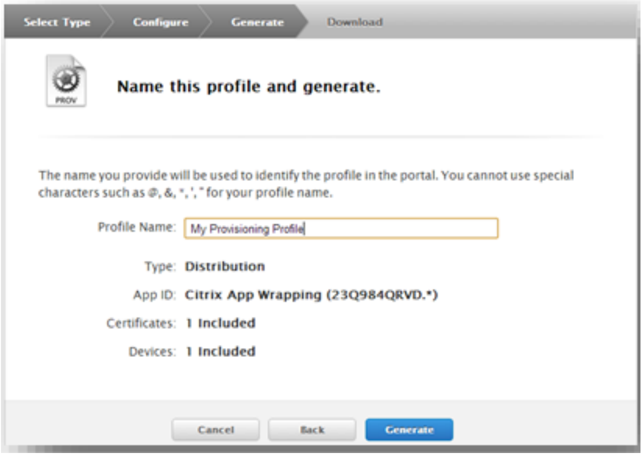
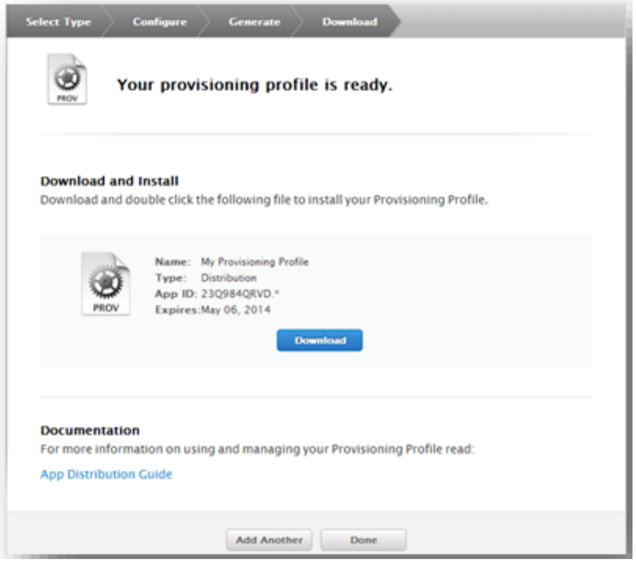
4.	Select the **App Store and Ad Hoc** button and then click **Continue** at the bottom of the page. **Note**: Download and install the WWDR intermediate certificate on the Mac where you will use the MDX Toolkit.	 Development ○ **iOS App Development** Sign development versions of your iOS app. ○ **Apple Push Notification service SSL (Sandbox)** Establish connectivity between your notification server and the Apple Push Notification service sandbox environment. A separate certificate is required for each app you develop. Production ● **App Store and Ad Hoc** Sign your iOS app for submission to the App Store or for Ad Hoc distribution. ○ **Apple Push Notification service SSL (Production)** Establish connectivity between your notification server and the Apple Push Notification service production environment. A separate certificate is required for each app you distribute. ○ **Pass Type ID Certificate** Sign and send updates to passes in Passbook. ○ **Website Push ID Certificate** Sign and send updates for Websites. **Intermediate Certificates** To use your certificates, you must have the intermediate signing certificate in your system keychain. This is automatically installed by Xcode. However, if you need to reinstall the intermediate signing certificate click the link below: Worldwide Developer Relations Certificate Authority
5.	Click **Continue**.	 **About Creating a Certificate Signing Request (CSR)** To manually generate a Certificate, you need a Certificate Signing Request (CSR) file from your Mac. To create a CSR file, follow the instructions below to create one using Keychain Access. **Create a CSR file.** In the Applications folder on your Mac, open the Utilities folder and launch Keychain Access. Within the Keychain Access drop down menu, select Keychain Access > Certificate Assistant > Request a Certificate from a Certificate Authority • In the Certificate Information window, enter the following information: • In the User Email Address field, enter your email address • In the Common Name field, create a name for your private key (eg. John Doe Dev Key) • In the Request is group, select the "Saved to disk" option • Click Continue within Keychain Access to complete the CSR generating process.
6.	Open **Keychain Access** on your Mac computer. Choose **Keychain Access > Certificate Assistant > Request a Certificate from a Certificate Authority**.	 Keychain Access File Edit View Window Help About Keychain Access Preferences... ⌘, Keychain First Aid ⌥⌘A Certificate Assistant ▸ Ticket Viewer ⌥⌘K Services ▸ Hide Keychain Access ⌘H Hide Others ⌥⌘H Show All Quit Keychain Access ⌘Q Open... Create a Certificate... Create a Certificate Authority... Create a Certificate For Someone Else as a Certificate Authority... Request a Certificate From a Certificate Authority... Set the default Certificate Authority... Evaluate "*.citrite.net"... Keychain Access 802.1X Password Today, 1:01 PM n Integration Certification Authority certificate -- Apple ID Authentication application password Mar 13, 2012 1:42:1 Apple ID citrixtm@me.com key public key --

7.	Enter your email address in the **User Email Address** field. Enter your name in the **Common Name** field. Select the **Saved to disk** option and then click **Continue.**	 Certificate Assistant Certificate Information Enter information for the certificate you are requesting. Click Continue to request a certificate from the CA. User Email Address: developer@citrix.com Common Name: TM CA Email Address: Request is: ○ Emailed to the CA ● Saved to disk □ Let me specify key pair information
8.	Save the Certificate Signing Request to a convenient folder on your computer.	 Save As: CertificateSigningRequest.certSigningR Where: Desktop
9.	Navigate back to the developer portal and then click **Choose File.**	 Generate your certificate. With the creation of your CSR, Keychain Access simultaneously generated a public and private key pair. Your private key is stored on your Mac in the login Keychain by default and can be viewed in the Keychain Access application under the "Keys" category. Your requested certificate will be the public half of your key pair. Upload CSR file. Select .certSigningRequest file saved on your Mac.
10.	Select the Certificate Signing Request file and click **Choose.** Click **Generate**.	 Desktop FAVORITES All My Files Macintosh HD Applications Desktop Documents Downloads Movies Music Pictures Mobile App Management ShareFile Vrapping Tool Bro...ndr.mdx op e A...agement nshots File ed folder CertificateSigningRequest.certSigningRequest Name CertificateSigningRequest.certSigningRequest Kind -- Size 948 bytes Created Today 1:11 PM Modified Today 1:11 PM Last opened Today 1:11 PM

11.	The certificate will appear in the portal. Download the certificate and install it on your Mac.	 Select Type · Request · Generate · Download Your certificate is ready. **Download, Install and Backup** Download your certificate to your Mac, then double click the .cer file to install in Keychain Access. Make sure to save a backup copy of your private and public keys somewhere secure. Name: iOS Distribution: James Consalves Type: iOS Distribution Expires: May 07, 2014 Download **Documentation** For more information on using and managing your certificates read: App Distribution Guide Add Another · Done
12.	Click **Identifiers** and then **App IDs** in the left pane. Click the Plus Sign **(+)** in the upper-right corner of the page.	 Identifiers App IDs Pass Type IDs Website Push IDs
13.	Provide an **App ID Description**.	 App ID Description Name: Citrix App Wrapping You cannot use special characters such as @, &, *, ', "
14.	If you're going to prep apps that you have not developed such as the Citrix @Work apps, select the **Wildcard App ID** radio button and then enter an asterisk (*****) in the **Bundle Identifier** field. If you're going to prep apps that you own, select the **Explicit App ID** button and then enter a unique App ID for each of your apps in the following format com.domainname.appname Click **Continue**.	 ● **Wildcard App ID** This allows you to use a single App ID to match multiple apps. To create a wildcard App ID, enter an asterisk (*) as the last digit in the Bundle ID field. Bundle ID: * Example: com.domainname.*

15.	Verify the information you entered and then click **Submit.** Click **Done** on the next page.	 **Confirm your App ID.** To complete the registration of this App ID, make sure your App ID information is correct, and click the submit button. App ID Description: **Citrix App Wrapping** Identifier: **23Q984QRVD.*** Data Protection: ○ Disabled Game Center: ○ Disabled iCloud: ○ Disabled In-App Purchase: ○ Disabled Passbook: ○ Disabled Push Notifications: ○ Disabled
16.	Click **Devices** in the left pane and then click the Plus Sign (**+)** in the upper-right corner of the page. **Note**: If you are using an iOS Developer Enterprise Program, you will not have register devices with the provisioning portal.	
17.	Enter a device **Name**, the associated **UUID** of the device, and then click **Continue.**	 **Registering a New Device or Multiple Devices** **Pre-Release Software Reminder** You may only share Apple pre-release software with employees, contractors, and members of your organization who are registered as Apple developers and have a demonstrable need to know or use Apple software to develop and test applications on your behalf. Unauthorized distribution of Apple confidential information (including pre-release software) is prohibited and may result in the termination of your Apple Developer Program. It may also subject you to civil and criminal liability. ● **Register Device** Name your device and enter its Unique Device Identifier (UDID). Name: My iPad UDID: 245c7a3ed863aa67d33f41b8b1a2de96e92d61b6 ○ **Register Multiple Devices** Upload a file containing the devices you wish to register. Please note that a maximum of 100 devices can be included in your file and it may take a few minutes to process. Download sample files
18.	Click **Provisioning** in the left pane, click **Distribution**, and then click the Plus Sign (**+)** in the upper-right corner of the page.	

19.	Click the **Ad Hoc** button and then click **Continue.** **Note**: If you are using an iOS Developer Enterprise Program, you will have to create an **In House** distribution profile instead of the **Ad Hoc.**	 What type of provisioning profile do you need? Development ○ iOS App Development Create a provisioning profile to install development apps on test devices. Distribution ○ App Store Create a distribution provisioning profile to submit your app to the App Store. ● Ad Hoc Create a distribution provisioning profile to install your app on a limited number of registered devices.
20.	Select the **App ID** you created.	 Select App ID. If you plan to use services such as Game Center, In-App Purchase, and Push Notifications, or want a Bundle ID unique to a single app, use an explicit App ID. If you want to create one provisioning profile for multiple apps or don't need a specific Bundle ID, select a wildcard App ID. Wildcard App IDs use an asterisk (*) as the last digit in the Bundle ID field. Please note that iOS App IDs and Mac App IDs cannot be used interchangeably. App ID: Citrix App Wrapping (23Q984QRVD.*)
21.	Select the certificate that you created.	 Select Type Configure Generate Download PROV Select certificates. Select the certificates you wish to include in this provisioning profile. To use this profile to install an app, the certificate the app was signed with must be included. ● iOS Distribution Cancel Back Continue

22.	Select the devices you want to associate with the provisioning profile and then click **Continue.**	 Select Type Configure Generate Download PROV **Select devices.** Select the devices you wish to include in this provisioning profile. To install an app signed with this profile on a device, the device must be included. □ Select All item(s) selected □ iPad Cancel Back Continue
23.	Name the provisioning profile and then click **Generate.**	 Select Type Configure Generate Download PROV **Name this profile and generate.** The name you provide will be used to identify the profile in the portal. You cannot use special characters such as @, &, *, ', " for your profile name. Profile Name: My Provisioning Profile Type: **Distribution** App ID: **Citrix App Wrapping (23Q984QRVD.*)** Certificates: **1 Included** Devices: **1 Included** Cancel Back Generate
24.	Click **Download** and save the provisioning profile on your Mac.	 **Your provisioning profile is ready.** **Download and Install** Download and double click the following file to install your Provisioning Profile. Name: My Provisioning Profile Type: Distribution App ID: 23Q984QRVD.* Expires: May 06, 2014 **Documentation** For more information on using and managing your Provisioning Profile read: App Distribution Guide

You should now have a valid provisioning profile. Please make sure both the private key and the certificate are installed on your Mac Keychain Access.

Resources

- http://blogs.citrix.com/2013/12/10/how-to-prep-mobile-applications/
- http://www.citrix.com/content/dam/citrix/en_us/documents/go/worx-app-sdk-overview-for-isvs.pdf