Apps@Work Container 1.6.0 for iOS

September 15, 2020

For complete product documentation see:
MobileIron Apps@Work Container Product Documentation Home Page

Contents

Release notes

The following provides information specific to this release:

About the Apps@Work Container

The Apps@Work Container is a custom iOS app designed to display the web-based Apps@Work enterprise app storefront in a full-screen web view or "container." This Apps@Work app displays the same app storefront as the Apps@Work web clip, but it also does the following:

- The Apps@Work app allows MobileIron Core to badge the Apps@Work app home screen icon when app updates are available.
  The badge displays the number of updates and featured apps which have not yet been installed on the device.
- The Apps@Work app allows the app storefront to remain accessible to users in restrictive deployments where MobileIron Core enforces restrictions to disable Safari on devices.

Download, rebrand, and sign the Apps@Work Container app if you want device users to have these features.

NOTE: When using MobileIron Core 9.5.0.0 through the most recently released version as supported by MobileIron, you can provide custom branding assets such as the app name, app color, and banner icon. You provide these in the MobileIron Core Admin Portal in **Apps > Apps@Work Settings**. For details, see "Apps@Work Branding" in the Apps@Work Guide.

New features and enhancements summary

For a summary of features introduced in previous releases, see MobileIron App@Work Container Product Documentation.

This release provides the following new features and enhancements:

- **Pasteboard notifications**: Users will no longer see pasteboard notifications.

Support and compatibility

The information in this section includes the components MobileIron supports with this product.

NOTE: The information provided is current at the time of this release. For MobileIron product versions released after this release, see that product version's release notes for the most current support and compatibility information.

Support policy

MobileIron defines supported and compatible as follows:

TABLE 1. SUPPORTED AND COMPATIBLE DEFINITIONS

Term	Definition
Supported product versions	The functionality of the product and version with currently supported releases was systematically tested as part of the current release and, therefore, will be supported.
Compatible product versions	The functionality of the product and version with currently supported releases has not been systematically tested as part of the current release, and therefore not supported. Based on previous testing (if applicable), the product and version is expected to function with currently supported releases.

MobileIron end of sale and support policy

For the MobileIron End of Sale and Support Policy, see MobileIron End of Sale and Support Policy.

Apps@Work Container supported and compatible table

The following table provides supported and compatible information available at the time of this release. For newer versions of the components, refer to the component release notes for that version.

TABLE 2. SUPPORT AND COMPATIBILITY FOR THE APPS@WORK APP

Product	Supported versions	Compatible versions
MobileIron Core and Connected Cloud	10.7.0.1, 10.8.0.0	10.1.0 - 10.4.0.4 , 10.5.2.1, 10.6.0.2
Mobile@Work for iOS	12.4.0	12.0.0 – 12.3.1
iOS	iOS 11 to iOS 13.7 iOS 14 beta 8	iOS 10
Xcode	11.7	10.2
macOS	10.15.6	10.12.5

Resolved issues

For resolved issues provided in previous releases, see MobileIron Apps@Work Container Product Documentation Home Page.

This release provides the following new resolved issue:

- **IA-322**: Fixed Face ID compatibility issue with Apps@Work Container.

Known issues

For known issues in previous releases, see MobileIron App@Work Container Product Documentation.

This release does not include any new known issues.

Limitations

For limitations in previous releases, see MobileIron App@Work Container Product Documentation.

This release does not include any new known limitations.

Installing Xcode command line tools

Older versions of Xcode do not automatically include Xcode command line tools, which are necessary for the branding/signing script to run. You can install command line tools from Xcode (Xcode >Preferences > Downloads). For more information, see Technical Note TN2339 Building from the Command Line with Xcode FAQ.

Badge update delay

Due to an AppConnect limitation, it takes two AppConnect app check-ins for MobileIron Core to push the APNS badge token to the device. Therefore, badge updates do not work until the second time the Apps@Work app checks-in with Core. If this is an issue for a device user, you can instruct the device user to do the following:

1. Terminate the Apps@Work app.
   Terminating the app is different from closing it. For example, to terminate an app on iOS 9 devices, touch the app card and flick it up and off the screen.
2. Relaunch the Apps@Work app.
3. In Mobile@Work, tap **Settings > Check for Updates**.
4. Tap **Continue**.

NOTE: You can use the Apps@Work app **WITHOUT** doing any of the following:
- enabling AppConnect (**Settings > System Settings > Additional Products > Licensed Products**)
- enabling the AppConnect global policy

Downloading the Apps@Work app

The package is available as a separate file in the following KB: Accessing the Apps@Work Container App and Documentation. You will need to click through a separate license agreement before being able to download the file.

Preparing and distributing Apps@Work Container

The following provides the steps for preparing and distributing Apps@Work Container:

Procedure overview

The following figure outlines the process of preparing and distributing Apps@Work Container. Perform these procedures using the Safari browser on a Mac OS X computer.

FIGURE 1. OVERVIEW OF PREPARING AND DISTRIBUTION APPS@WORK CONTAINER

Creating a distribution certificate

The distribution certificate authenticates that the app comes from a source that is trusted by Apple.

Procedure

1. Log in to Apple's iOS Dev Center.
2. Under Developer Program, select **Certificates, Identifiers, and Profiles**.

3. Click **Certificates**.
4. Under Certificates, select **Production**.
5. Click **+** to add a certificate.
6. In the Production section, select **In House and Ad Hoc**.
7. Click **Continue**.

   NOTE: If you already have a distribution certificate, contact the person who created the certificate to export the certificate along with its private key to a .CER or .P12 file. Save this file to your Downloads folder and proceed to Adding the distribution certificate to your Login Keychain.
8. Complete the process for generating a Certificate Signing Request (CSR).
9. Save the CSR file to your desktop.
10. Click **Browse**.
11. Select the CSR file.
12. Click **Submit**.
    This step sends the CSR to Apple for approval. When the certificate request is approved, a Download button displays in the Actions column. If the Download button does not display, try refreshing the page.
13. Click **Download**.
14. Click **Save File**.
    This step saves the file in your Downloads folder.

Next steps

See Adding the distribution certificate to your Login Keychain.

Adding the distribution certificate to your Login Keychain

The following describes how to add the distribution certificate to your Login Keychain.

Procedure

1. Double-click the certificate in your Downloads folder.
   This step opens Keychain Access and adds the certificate to your Keychain.

   NOTE: No confirmation message is displayed.
2. In Keychain Access, confirm that the distribution certificate is listed in the Login Keychain.
   The certificate should have the following name: “iPhone Distribution: <Company Name>”.
3. If the certificate is not in the Login Keychain, check the other keychains.
4. If the certificate is in another keychain, move it to the Login Keychain.

Creating a unique app ID

The app ID grants your app access to the distribution certificate you added to your Login Keychain. Do the following steps on a Mac OS X computer using Safari

Procedure

1. Log in to Apple's iOS Dev Center.
2. Under Developer Program Resources, select **Certificates, Identifiers & Profiles**.
3. Click **Identifiers**.
4. Click **App IDs**.
5. Click **+** to add an ID.
6. In the Description field, enter a brief description of the app.
   Device users do not see this text.
7. Select **Explicit App ID**.
8. In the Bundle Identifier field, enter a unique identifier for the app bundle.
   We recommend using the following format:
   `com.yourcompany.storefrontcontainer`
9. Under the App Services section, select **Push Notifications** under the app services section (see Creating a push certificate.
10. Click **Continue**.
    The new app ID displays in the App IDs page.
11. Click **Submit**.

Creating a push certificate

Push notification enables badging for the app. Configuring push notifications requires a push certificate, also known as an APNS certificate. Do the following steps on a Mac OS X computer using Safari.

Procedure

1. Log in to Apple's iOS Dev Center.
2. Under Developer Program Resources, select "Certificates, Identifiers & Profiles".
3. Click **Certificates**.
4. Click **Production**.
5. Click **+**.
6. Select **Apple Push Notification Service SSL (Production)**.

7. Click **Continue**.
8. Select the app ID you created for this app.
9. Click **Continue**.
10. Follow the displayed instructions to create a CSR and save it to your desktop.

Next steps

See Adding the push certificate to your Login Keychain.

Adding the push certificate to your Login Keychain

The following describes how to add the push certificate to your Login Keychain.

Procedure

1. Open the push certificate you saved in your Downloads folder.
   The certificate will be labeled “Apple Production IOS Push Services: <bundle ID>”.
   This step adds the certificate to your Login Keychain and opens the Keychain Access app.
2. In the Keychain Access app, click the arrow to the left of the push certificate.
   This step displays the private key for the certificate.
3. Select the certificate and the private key.
4. Select **File** > **Export Items**.
   This step saves the information to a file in p12 format.
5. Enter the password for exporting when prompted.
6. Click **OK**.
7. Enter your Login Keychain password when prompted.
   This is typically your login password.
8. Click **Allow**.
9. In the SSL Certificate Assistant window, click **Done**.

Next steps

See Creating a distribution provisioning profile.

Creating a distribution provisioning profile

The distribution provisioning profile associates the distribution certificate with your app and authorizes devices to use the app.

Procedure

1. Log in to Apple's iOS Dev Center.
2. Under Developer Program Resources, select **Certificates, Identifiers & Profiles**.
3. Click **Provisioning Profiles**.
4. Select **Distribution**.
5. Click **+**.
6. Under Distribution, select **In House**.
7. Click **Continue**.
8. Select **App ID**.
9. Click **Continue**.
10. Select the distribution certificate you created.
11. Click **Continue**.
12. The following prompt is displayed:
    Do you need additional entitlements?
13. Click **Continue**.
14. Enter a name for the profile.
    Enter a name that distinguishes the profile from others, such as "Storefront Container In House Profile."
15. Click **Generate**.
    The file will have a .mobileprovision extension.

Next steps

See Signing and rebranding your custom Apps@Work app.

Signing and rebranding your custom Apps@Work app

The script included in the package you downloaded from the MobileIron support site completes the following tasks:

- rebrands the app (optional)
- signs the app

Rebranding requirements

You can rebrand the following elements in your custom app:

- app title (This is the app name displayed on the home screen; the default is Apps@Work.)
- icons (The default icons are the same as those used for the Apps@Work web clip.)

If you are rebranding the app, prepare the icons as specified by Apple on Icon and Image Sizes.

You will use these icons in Running the script. You will also use the Icon@2x.png icon in Distributing the app when you upload the Apps@Work app to MobileIron Core.

Accessing the default app icon

If you are not re-branding the app icon, then you need to acquire the default app icon included in the Apps@Work Container package.

Procedure

1. In the folder in which you extracted the Apps@Work Container package, open the **Payload** folder.
2. Right-click the **WebContainer** file to display the context menu.
3. Click **Show Package Contents**.
   The default set of app icons are in the displayed folder.
4. Copy (but do not move) **Icon@2x.png** to your Downloads folder.

You will use the default app icon in Distributing the app when you upload the Apps@Work app to MobileIron Core.

Signing requirements

Assemble the following items before running the signing/rebranding script:

- provisioning profile (created in "Creating a distribution provisioning profile" on Creating a distribution provisioning profile)
- name of the distribution certificate (created in "Creating a distribution certificate" on Creating a distribution certificate)
- app version (You can choose your preferred version number for the app. We recommend using the version of the Apps@Work app package itself. However, you must increment this version if you are distributing multiple updates to the Apps@Work app, such as revised app icons.)

Running the script

When you have assembled the required items, complete the following steps to run the signing/rebranding script.

Procedure

1. Make sure you have installed Xcode command line tools.
   See Installing Xcode command line tools.
2. Extract the Apps@Work Container package you downloaded from the MobileIron support site.
3. Place your provisioning profile in the folder containing the extracted contents.

4. If you are rebranding the app icons, create a new directory in the same folder which you extracted the package and place your rebranded icons in this folder.
   Be sure the dimensions and file names of all of your icons exactly match the dimensions and filenames of the icons specified by Apple on Icon and Image Sizes.
5. Open Terminal.
6. In the folder containing the extracted contents, run the script with the following command:
   ./brandAndSign.py --provisioningProfile <path_to_ProvisioningProfile.mobileprovision> --signingIdentity "<distribution certificate common name>" --appTitle "<app title>" --appIconsDirectory <directory> --appBundleVersion <build version> --appDisplayVersion <release version>

   Example

   ./brandAndSign.py --provisioningProfile ./ProvisioningProfile.mobileprovision --signingIdentity "iPhone Distribution: <Company Name>" --appTitle "My App" --appIconsDirectory ./AppIcons/ --appBundleVersion 2.1.3.57 --appDisplayVersion 2.1.3

NOTE: The version options should be used only if you to manage the versions to track things like updated branding for an app that has already been deployed. The appBundleVersion parameter assigns a custom bundle version, which is for internal version tracking. The appDisplayVersion parameter assigns a custom version to the app for display to app users.

Distributing the app

The steps for distributing the signed app resemble those of distributing an in-house app. However, you must also upload the APNS certificate you created in Creating a push certificate. The MobileIron Core process for adding an in-house app presents a field for uploading the APNS certificate when you upload the Apps@Work app.

NOTE: The following procedure uses MobileIron Core 9.4.0.0.

Procedure

1. Make sure you have access to the Icon@2x.png icon.
   - If you are rebranding, this icon is in the icon directory you created when you signed the app.
   - If you are not rebranding, this is the default app icon you acquired as described in Accessing the default app icon.
2. Select **Apps > App Catalog**.
3. Click **Add+**.
4. Select **In-House**.
5. Next to **Upload In-House App**, click **Browse** to navigate to and select the signed Apps@Work app IPA.
6. Click **Next**.
7. Optionally add a description.

8. Click **Next**.
9. In the **Apps@Work Catalog** section, uncheck **Feature this App in the Apps@Work catalog**.
10. In the **Icon and Screenshots** section, click **Replace Icon** to navigate to and select the icon for the app.
11. Click **Next**.
12. In the **Managed App Settings** section, select **Send installation request or send convert unmanaged to managed app request (iOS 9 and later) on device registration or sign-in**.
    This step ensures that newly-registered devices receive a prompt to install this app. This is an important step for the Apps@Work app because the app does not display as a separate entry in Apps@Work.
13. In the **APNS Messaging Configuration** section, next to the **APNS Certificate** field, click **Browse** to navigate to and select the APNS certificate that you created in Creating a push certificate.
14. For **Password**, enter the password for the APNS certificate.
15. Click **Finish**.
    MobileIron Core automatically creates an AppConnect app configuration and AppConnect container policy for the Apps@Work app. Core uses these settings to automatically distribute the required configuration to the Apps@Work app on installed devices without requiring end-user interaction.
16. In **Apps > App Catalog**, select the Apps@Work app.
17. Select **Actions > Apply To Labels**.
18. Select the appropriate labels.
19. Click **Apply**.

After these steps, Apps@Work app will be installed on devices when they register with MobileIron Core.

Distributing the app to registered users

To distribute the app to devices that are *already* registered, use one of the following procedures:

- Send a message from the Core Admin Portal
- Ask device users to check for updates

Send a message from the Core Admin Portal

From the Core Admin Portal send a message to device users.

Procedure

1. In the Core Admin Portal, go to **Apps > App Catalog**.
2. Select the Apps@Work app.

3. Select **Actions > Send Message**.
4. In the **Send App Installation Request dialog**, click **Apply**.

Ask device users to check for updates

Ask devices to do the following to check for updates.

Procedure

1. Go to **Settings > Check for Updates.**
2. Tap **Continue**.