MOBILEIRON CORE V10.X MDM SUPPLEMENTAL PROCEDURES

Version 1, Release 1

15 February 2019

Developed by MobileIron and DISA for the DoD

Trademark Information

Names, products, and services referenced within this document may be the trade names, trademarks, or service marks of their respective owners. References to commercial vendors and their products or services are provided strictly as a convenience to our users, and do not constitute or imply endorsement by DISA of any non-Federal entity, event, product, service, or enterprise.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

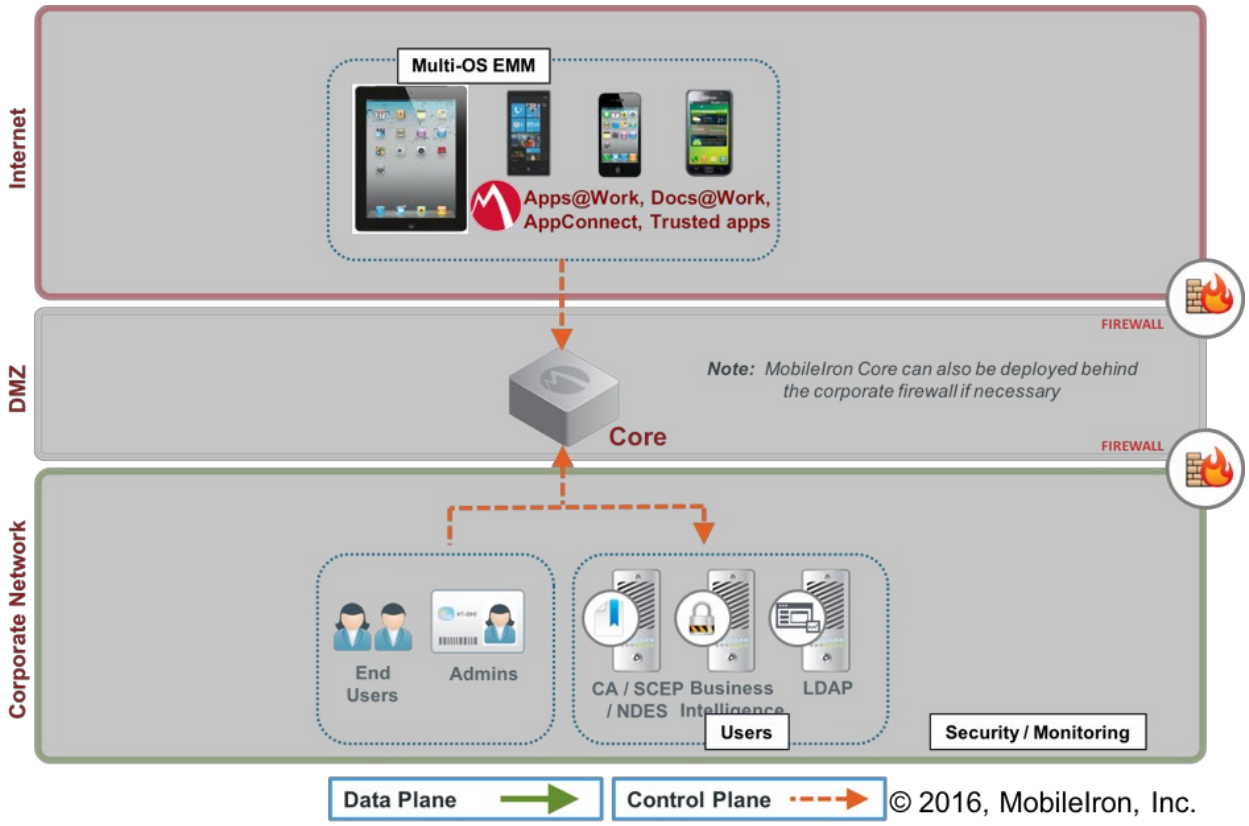
LIST OF FIGURES

Page

1. MOBILEIRON MDM SOFTWARE SECURITY AND CONFIGURATION INFORMATION

1.1 MobileIron MDM Architecture

Figure 1-1: MobileIron Core MDM Architecture



1.2 MobileIron MDM Software Components

Table 1-1: MobileIron Core Components

Component	Description
Mobile@Work for Android	MobileIron MDM Agent for Android
MobileIron Core	MobileIron MDM Server

1.3 MobileIron MDM Required Firewall Ports

Table 1-2: Required Ports and Services

From	To	Port (TCP)	Description
Administrators	MDM Server	22	SSH
Mobile Devices	MDM Server	80	HTTP (for CRLs)

From	To	Port (TCP)	Description
Mobile Devices	MDM Server	443	HTTPS
Administrators	MDM Server	8443	HTTPS-alt
Mobile Devices	MAS (component of Core)	7443	HTTPS-alt

1.4 PKI Considerations

In order to implement over-the-air (OTA) provisioning of a DoD mobile device, an authenticated and encrypted tunnel can be set up between the mobile device and the mobile device management (MDM) server. The mobile device and MDM server must support the same root certificate authority to set up a mutually authenticated trusted tunnel between both endpoints. In order for the mobile device to support the current DoD root Certificate Authority, DoD Root CA 3, the mobile device needs to natively, out-of-the-box, trust the current DoD root Certificate Authority, or the certificate will need to be side-loaded on the mobile device, which is not scalable in an Enterprise environment. Unfortunately, few, if any, mobile devices natively trust this root CA. Alternately, since there is a path of trust between DoD Root CA 3 and the Federal Common Policy Certificate Authority (FCPCA), a mobile device that natively trusts the FCPCA can authenticate the MDM if either the MDM server or web service used by the MDM (for example IIS, Apache) pushes down a path to the FCPCA to the mobile device during the TLS handshake.

The MobileIron MDM's web service is provided by Apache. A Local Admin on the MDM can manage these certificates through the Web UI's System Manager, by navigating to the “Security” tab and selecting "Certificate Mgmt". They can then upload a PKCS12 file containing the server's certificate and all CA certificates in the path from the DoD PKI Issuing CA (e.g., DoD ID SW CA 37) to Federal Common Policy.

1.5 Provisioning Derived Credentials

The need to provision derived credentials benefits from some MDM features that are not required to support other functionality. This section describes these features for Android and iOS.

1.5.1 Android

Starting with Android P, some key management APIs require key management applications to be configured as a device owner, profile owner, device owner delegate, or profile owner delegate. To support the full range of options, MDMs should support the setCertInstallerPackage interface of the DevicePolicyManager class.

1.5.2 Apple iOS

On iOS, to enable third-party apps to use derived credentials, the key sharing interface of the Purebred application should be leveraged. The key sharing interface is a use of Apple's document provider extensions to share PKCS 12 objects between a key management application and an application desired to use keys. Sample code is available at https://github.com/purebred.

For iOS 12, depending on the MDM vendor and the use of the iOS provided mail client for work email, a managed Exchange payload with the following settings set to "True" could be leveraged to allow users to select Purebred-issued credentials for signed and encrypted email:

SMIMESigningUserOverrideable;
SMIMESigningCertificateUUIDUserOverrideable;
SMIMEEncryptByDefaultUserOverrideable;
SMIMEEncryptionCertificateUUIDUserOverrideable