iOS Deployment Technical Reference

iOS 7.1

May 2014

Contents

Introduction

This guide is for IT administrators who want to support iOS devices on their networks. It provides information about deploying and supporting iPhone, iPad, and iPod touch in a large-scale organization such as an enterprise or education institution. It explains how iOS devices provide comprehensive security, integration with your existing infrastructure, and powerful tools for deployment.

Understanding the key technologies supported in iOS will help you implement a deployment strategy that provides an optimal experience for your users. The following chapters serve as a technical reference you can use when deploying iOS devices throughout your organization:

Integration. iOS devices have built-in support for a wide range of network infrastructures. In this section, you'll learn about iOS-supported technologies and best practices for integrating with Microsoft Exchange, Wi-Fi, VPN, and other standard services.

Security. iOS is designed to securely access corporate services and protect important data. iOS provides strong encryption for data in transmission, proven authentication methods for access to corporate services, and hardware encryption for all data at rest. Read this chapter to learn more about the security-related features of iOS.

Configuration and Management. iOS supports advanced tools and technologies for ensuring that iOS devices are easily set up, configured to meet your requirements, and managed with ease in a large-scale environment. This chapter provides an overview of mobile device management (MDM).

App Distribution. There are a number of ways of deploying apps and content throughout your organization. The iOS Developer Enterprise Program enables your organization to build, and deploy apps for your internal users. Use this chapter to get an in-depth understanding on how to deploy apps built for internal use.

The following appendices provide additional technical details and requirements:

Wi-Fi Infrastructure. Details about the Wi-Fi standards that iOS supports and considerations for planning a large-scale Wi-Fi network.

Restrictions. Details about the restrictions you can use to configure iOS devices to meet your security, passcode, and other requirements.

Installing In-House Apps Wirelessly. Details and requirements for distributing your in-house apps using your own web-based portal.

Additional resources

For helpful related information, refer to the following websites:

www.apple.com/ipad/business/it

www.apple.com/iphone/business/it

www.apple.com/education/it

Chapter 1: **Integration**

iOS devices have built-in support for a wide range of network infrastructures. They include support for the following:

- Popular third-party systems like Microsoft Exchange
- Integration with standards-based mail, directory, calendar, and other systems
- Standard Wi-Fi protocols for data transmission, encryption
- Virtual private networks (VPN) including per app VPN
- Single sign on to streamline authentication to networked apps and services
- Digital certificates to authenticate users and secure communications

Because this support is built into iOS, your IT department only needs to configure a few settings to integrate iOS devices into your existing infrastructure. Read on to learn more about iOS-supported technologies and best practices for integration.

Microsoft Exchange

iOS can communicate directly with your Microsoft Exchange Server via Microsoft Exchange ActiveSync (EAS), enabling push email, calendar, contacts, notes, and tasks. Exchange ActiveSync also provides users with access to the Global Address List (GAL), and provides administrators with passcode policy enforcement and remote wipe capabilities. iOS supports both basic and certificate-based authentication for Exchange ActiveSync.

If your company currently enables Exchange ActiveSync, you have the necessary services in place to support iOS—no additional configuration is required.

Requirements

Devices with iOS 7 or later support the following versions of Microsoft Exchange:

- Exchange Server 2003 SP 2 (EAS 2.5)
- Exchange Server 2007 (utilizing EAS 2.5)
- Exchange Server 2007 SP 1 (EAS 12.1)
- Exchange Server 2007 SP 2 (EAS 12.1)
- Exchange Server 2007 SP 3 (EAS 12.1)
- Exchange Server 2010 (EAS 14.0)
- Exchange Server 2010 SP 1 (EAS 14.1)
- Exchange Server 2010 SP 2 (utilizing EAS 14.1)
- Exchange Server 2013 (utilizing EAS 14.1)
- Office 365 (utilizing EAS 14.1)

Microsoft Direct Push

Exchange Server automatically delivers email, tasks, contacts, and calendar events to iOS devices if a cellular or Wi-Fi data connection is available. iPod touch and some iPad models don’t have a cellular connection, so they receive push notifications only when connected to a Wi-Fi network.

Microsoft Exchange Autodiscovery

iOS supports the Autodiscover service of Microsoft Exchange Server 2007 and Microsoft Exchange Server 2010. When you manually configure a device, Autodiscover uses your email address and password to determine the correct Exchange Server information.

For more information on enabling the Autodiscover service, refer to Autodiscover Service.

Microsoft Exchange Global Address List

iOS devices retrieve contact information from your company's Exchange Server corporate directory. You can access the directory while searching in Contacts, and it's automatically accessed for completing email addresses as you enter them. iOS 6 or later supports GAL photos (requires Exchange Server 2010 SP 1 or later).

Unsupported Exchange ActiveSync features

The following features of Exchange are not supported:

- Opening links in email to documents stored on SharePoint servers
- Setting an out-of-office autoreply message

Identifying iOS versions via Exchange

When an iOS device connects to an Exchange Server, the device reports its iOS version. The version number is sent in the User Agent field of the request header and looks similar to Apple-iPhone2C1/705.018. The number after the delimiter (/) is the iOS build number, which is unique to each iOS release.

To view the build number on a device, go to Settings>General>About. You'll see the version number and build number, such as 4.1 (8B117A). The number in parentheses is the build number, which identifies the release the device is running.

When the build number is sent to the Exchange Server, it's converted from the format NANNNA (where N is numeric and A is an alphabetic character) to the Exchange format NNN.NNN. Numeric values are kept, but letters are converted to their positional value in the alphabet. For example, "F" is converted to "06" because it's the sixth letter in the alphabet. Numbers are padded with zeros if necessary, to fit the Exchange format.

In this example, the build number 7E18 is converted to 705.018.

The first number, 7, remains as "7." The character E is the fifth letter in the alphabet so it's converted to "05." A period (.) is inserted in the converted version, as required by the format. The next number,18, is padded with zero and converted to "018."

If the build number ends with a letter, such as 5H11A, the number is converted as described above, and the numeric value of the final character is appended to the string separated by 3 zeroes. So 5H11A becomes 508.01100001.

Remote wipe

You can remotely wipe the contents of an iOS device using features provided by Exchange. Wiping removes all data and configuration information from the device, and the device is securely erased and restored to its original factory settings. Wiping removes the encryption key to the data (encrypted using 256-bit AES encryption), which immediately makes all of the data unrecoverable.

With Microsoft Exchange Server 2007 or later, you can perform a remote wipe with the Exchange Management Console, Outlook Web Access, or the Exchange ActiveSync Mobile Administration Web Tool. With Microsoft Exchange Server 2003, you can initiate a remote wipe using the Exchange ActiveSync Mobile Administration Web Tool.

Alternatively, users can wipe their own device by going to Settings>General>Reset and choosing "Erase All Content and Settings." Devices can also be configured to be automatically wiped after a specified number of failed passcode attempts.

Standards-Based Services

With support for the IMAP mail protocol, LDAP directory services, CalDAV calendaring, and CardDAV contacts protocols, iOS can integrate with just about any standards-based environment. And if your network environment is configured to require user authentication and SSL, iOS provides a secure approach to accessing standards-based corporate email, calendar, tasks, and contacts. With SSL, iOS supports 128-bit encryption and X.509 root certificates issued by the major certificate authorities.

In a typical deployment, iOS devices establish direct access to IMAP and SMTP mail servers to send and receive email over the air, and can also wirelessly sync notes with IMAP-based servers. iOS devices can connect to your company's LDAPv3 corporate directories, giving users access to corporate contacts in the Mail, Contacts, and Messages applications. Synchronization with your CalDAV server allows users to wirelessly create and accept calendar invitations, receive calendar updates, and sync tasks with the Reminders app. And CardDAV support allows your users to maintain a set of contacts synced with your CardDAV server using the vCard format. All network servers can be located within a DMZ subnetwork, behind a corporate firewall, or both.

Wi-Fi

Out of the box, iOS devices can securely connect to corporate or guest Wi-Fi networks, making it quick and simple for users to join available wireless networks whether they're on campus or on the road.

Joining Wi-Fi

Users can set iOS devices to join available Wi-Fi networks automatically. Wi-Fi networks that require login credentials or other information can be quickly accessed without opening a separate browser session, from Wi-Fi settings or within apps such as Mail. And low-power, persistent Wi-Fi connectivity allows apps to use Wi-Fi networks to deliver push notifications.

WPA2 Enterprise

iOS supports industry-standard wireless network protocols, including WPA2 Enterprise, ensuring corporate wireless networks can be accessed securely from iOS devices. WPA2 Enterprise uses 128-bit AES encryption, a proven, block-based encryption method, providing users with the highest level of assurance that their data will remain protected.

With support for 802.1X, iOS can be integrated into a broad range of RADIUS authentication environments. 802.1X wireless authentication methods supported by iOS include EAP-TLS, EAP-TTLS, EAP-FAST, PEAPv0, PEAPv1, and LEAP.

Roaming

For roaming on large enterprise Wi-Fi networks, iOS supports 802.11k and 802.11r. 802.11k helps iOS devices transition between Wi-Fi access points by utilizing the reports from the access points, while 802.11r streamlines 802.1X authentication as a device moves from one access point to another.

For quick setup and deployment, wireless network, security, proxy, and authentication settings can be configured using configuration profiles or MDM.

Virtual Private Networks

Secure access to private corporate networks is available in iOS using established industry-standard virtual private network (VPN) protocols. Out of the box, iOS supports Cisco IPSec, L2TP over IPSec, and PPTP. If your organization supports one of these protocols, no additional network configuration or third-party apps are required to connect iOS devices to your VPN.

Additionally, iOS supports SSL VPN from popular VPN providers. Users simply download VPN client apps developed by one of these companies from the App Store to get started. Like other VPN protocols supported in iOS, SSL VPN can be configured manually on the device or via configuration profiles or MDM.

iOS supports industry-standard technologies such as IPv6, proxy servers, and split-tunneling, providing a rich VPN experience when connecting to corporate networks. And iOS works with a variety of authentication methods including password, two-factor token, and digital certificates. To streamline the connection in environments where certificate-based authentication is used, iOS features VPN On Demand, which initiates a VPN session when it's needed to connect to specified domains.

With iOS 7, individual apps can be configured to utilize a VPN connection independent from other apps on the device. This ensures that corporate data always flows over a VPN connection, and other data, such as an employee's personal apps from the App Store, does not. For details, see "Per App VPN" later in this chapter.

Supported protocols and authentication methods

SSL VPN. Supports user authentication by password, two-factor token, and certificates.

Cisco IPSec. Supports user authentication by password, two-factor token, and machine authentication by shared secret and certificates.

L2TP over IPSec. Supports user authentication by MS-CHAP v2 Password, two-factor token, and machine authentication by shared secret.

PPTP. Supports user authentication by MS-CHAP v2 Password and two-factor token.

SSL VPN clients

Several SSL VPN providers have created apps that assist in the configuration of iOS devices for use with their solutions. To configure a device for a specific solution, install the companion app and, optionally, provide a configuration profile with the necessary settings. SSL VPN solutions include:

- **Juniper Junos Pulse SSL VPN.** iOS supports Juniper Networks SA Series SSL VPN Gateway running version 6.4 or later with Juniper Networks IVE package 7.0 or later. For configuration, install the Junos Pulse app, available from the App Store.

For more information, refer to Juniper Networks application note.

- **F5 SSL VPN.** iOS supports F5 BIG-IP Edge Gateway, Access Policy Manager, and FirePass SSL VPN solutions. For configuration, install the F5 BIG-IP Edge Client app, available from the App Store.

  For more information, refer to the F5 technical brief, Secure iPhone Access to Corporate Web Applications.

- **Aruba Networks SSL VPN.** iOS supports Aruba Networks Mobility Controller. For configuration, install the Aruba Networks VIA app, available from the App Store.

  For contact information, refer to the Aruba Networks website.

- **SonicWALL SSL VPN**. iOS supports SonicWALL Aventall E-Class Secure Remote Access appliances running 10.5.4 or later, SonicWALL SRA appliances running 5.5 or later, and SonicWALL Next-Generation Firewall appliances including the TZ, NSA, E-Class NSA running SonicOS 5.8.1.0 or higher. For configuration, install the SonicWALL Mobile Connect app, available from the App Store.

  For contact information, refer to the SonicWALL website.

- **Check Point Mobile SSL VPN.** iOS supports the Check Point Security Gateway with a full Layer-3 VPN tunnel. For configuration, install the Check Point Mobile app, available from the App Store.

- **OpenVPN SSL VPN.** iOS supports OpenVPN Access Server, Private Tunnel, and OpenVPN Community. For configuration, install the OpenVPN Connect app, available from the App Store.

- **Palo Alto Networks GlobalProtect SSL VPN.** iOS supports the GlobalProtect gateway from Palo Alto Networks. For configuration, install the GlobalProtect for iOS app, available from the App Store.

- **Cisco AnyConnect SSL VPN.** iOS supports Cisco Adaptive Security Appliance (ASA) running software image 8.0(3).1 or later. For configuration, install the Cisco AnyConnect app, available from the App Store.

VPN Setup Guidelines

Cisco IPSec setup guidelines

Use these guidelines to configure your Cisco VPN server for use with iOS devices. iOS supports Cisco ASA 5500 Security Appliances and PIX Firewalls configured with 7.2.x software or later. The latest software release (8.0.x or later) is recommended. iOS also supports Cisco IOS VPN routers with IOS version 12.4(15)T or later. VPN 3000 Series Concentrators don't support iOS VPN capabilities.

Proxy setup

For all configurations, you can also specify a VPN proxy. To configure a single proxy for all connections, use the Manual setting and provide the address, port, and authentication if necessary. To provide the device with an auto-proxy configuration file using PAC or WPAD, use the Auto setting. For PAC, specify the URL of the PAC file. For WPAD, iOS queries DHCP and DNS for the appropriate settings.

Authentication methods
iOS supports the following authentication methods:

- Pre-shared key IPSec authentication with user authentication via xauth.
- Client and server certificates for IPSec authentication, with optional user authentication via xauth.
- Hybrid authentication, where the server provides a certificate and the client provides a pre-shared key for IPSec authentication. User authentication is required via xauth.
- User authentication is provided via xauth and includes the following authentication methods:
  - Username with password
  - RSA SecurID
  - CRYPTOCard

Authentication groups
The Cisco Unity protocol uses authentication groups to group users based on a common set of authentication parameters and other parameters. You should create an authentication group for iOS users. For pre-shared key and hybrid authentication, the group name must be configured on the device with the group's shared secret (pre-shared key) as the group password.

When using certificate authentication, no shared secret is used. A user's group is determined based on fields in the certificate. The Cisco server settings can be used to map fields in a certificate to user groups.

RSA-Sig should be the highest priority on the ISAKMP priority list.

Certificates
When setting up and installing certificates, make sure of the following:

The server identity certificate must contain the server's DNS name and/or IP address in the subject alternate name (SubjectAltName) field. The device uses this information to verify that the certificate belongs to the server. For more flexibility, you can specify the SubjectAltName using wildcard characters for per-segment matching, such as vpn.*.mycompany.com. You can put the DNS name in the common name field, if no SubjectAltName is specified.

The certificate of the CA that signed the server's certificate needs to be installed on the device. If it isn't a root certificate, install the rest of the trust chain so that the certificate is trusted. If you use client certificates, make sure the trusted CA certificate that signed the client's certificate is installed on the VPN server. When using certificate-based authentication, make sure the server is set up to identify the user's group based on fields in the client certificate.

The certificates and certificate authorities must be valid (for example, not expired). Sending of certificate chain by the server isn't supported and you should turn it off.

IPSec settings

Use the following IPSec settings:

- **Mode.** Tunnel Mode
- **IKE Exchange Modes.** Aggressive Mode for pre-shared key and hybrid authentication, or Main Mode for certificate authentication.
- **Encryption Algorithms.** 3DES, AES-128, AES-256.
- **Authentication Algorithms.** HMAC-MD5, HMAC-SHA1.
- **Diffie-Hellman Groups.** Group 2 is required for pre-shared key and hybrid authentication. For certificate authentication, use Group 2 with 3DES and AES-128. Use Group 2 or 5 with AES-256.
- **PFS (Perfect Forward Secrecy).** For IKE phase 2, if PFS is used, the Diffie-Hellman group must be the same as was used for IKE phase 1.
- **Mode Configuration.** Must be enabled.
- **Dead Peer Detection.** Recommended.
- **Standard NAT Transversal.** Supported and can be enabled (IPSec over TCP isn't supported).
- **Load Balancing.** Supported and can be enabled.
- **Re-keying of Phase 1.** Not currently supported. It's recommend that re-keying times on the server be set to one hour.
- **ASA Address Mask.** Make sure all device address pool masks either are not set, or are set to 255.255.255.255. For example: asa(config-webvpn)# ip local pool vpn_users 10.0.0.1-10.0.0.254 mask 255.255.255.255.

If you use the recommended address mask, some routes assumed by the VPN configuration might be ignored. To avoid this, make sure your routing table contains all necessary routes and verify that the subnet addresses are accessible before deployment.

Other supported features

- **Application Version.** The client software version is sent to the server, allowing the server to accept or reject connections based on the device's software version.
- **Banner.** The banner (if configured on the server) is displayed on the device and the user must accept it or disconnect.
- **Split Tunnel.** Split tunneling is supported.
- **Split DNS.** Split DNS is supported.
- **Default Domain.** Default domain is supported.

VPN On Demand

VPN On Demand allows iOS to automatically establish a secure connection without user action. The VPN connection is started on an as-needed basis based on rules defined in a configuration profile.

In iOS 7, VPN On Demand is configured using the OnDemandRules key in a VPN payload of a configuration profile. Rules are applied in two stages:

- **Network Detection Stage.** Defines VPN requirements that are applied when the device's primary network connection changes.
- **Connection Evaluation Stage.** Defines VPN requirements for connection requests to domain names on an as-needed basis.

For example, rules can be used to:

- Recognize when an iOS device is connected to an internal network and VPN is not necessary.
- Recognize when an unknown Wi-Fi network is being used and require VPN for all network activity.
- Require VPN when a DNS request for a specified domain name fails.

Network Detection Stage

VPN On Demand rules are evaluated when the device's primary network interface changes, such as when an iOS device changes to a different Wi-Fi network, or switches to cellular from Wi-Fi. If the primary interface is a virtual interface, such as a VPN interface, VPN On Demand rules are ignored.

The matching rules in each set (dictionary) must all match in order for their associated action to be taken; if any one of the rules does not match, evaluation falls through to the next dictionary in the array, until the OnDemandRules array is exhausted.

The last dictionary should define a "default" configuration—that is, it should have no matching rules, only an Action. This will catch all connections that haven't matched the preceding rules.

Connection Evaluation Stage

VPN can be triggered as needed based on connection requests to certain domains, instead of unilaterally disconnecting or connecting VPN based on the network interface.

On Demand Matching Rules

Specify one or more of the following matching rules:

- **InterfaceTypeMatch.** Optional. A string value of Wi-Fi or cellular. If specified, this rule will match when the primary interface hardware is of the type specified.
- **SSIDMatch.** Optional. An array of SSIDs to match against the current network. If the network is not a Wi-Fi network or if its SSID does not appear in the list, the match will fail. Omit this key and its array to ignore SSID.
- **DNSDomainMatch.** Optional. An array of search domains as strings. If the configured DNS search domain of the current primary network is included in the array, this property will match. Wildcard prefix (*) is supported; e.g., *.example.com would match anything.example.com.

- **DNSServerAddressMatch.** Optional. An array of DNS servers addresses as strings. If all of the DNS server addresses currently configured for the primary interface are in the array, this property will match. Wildcard (*) is supported; e.g., 1.2.3.* would match any DNS servers with 1.2.3. prefix.
- **URLStringProbe.** Optional. A server to probe for reachability. Redirection is not supported. The URL should be to a trusted HTTPS server. The device sends a GET request to verify that the server is reachable.

Action
This key defines VPN behavior for when all of the specified matching rules evaluate as true. This key is required. Values for the Action key are:

- **Connect.** Unconditionally initiates the VPN connection on the next network connection attempt.
- **Disconnect.** Tear down the VPN connection and do not trigger any new connections on demand.
- **Ignore.** Leave any existing VPN connection up, but do not trigger any new connections on demand.
- **Allow.** For iOS devices with iOS 6 or earlier. See "Notes on Backward Compatibility," later in this section.
- **EvaluateConnection.** Evaluate the ActionParameters for each connection attempt. When this is used, the key ActionParameters, described below, is required to specify the evaluation rules.

ActionParameters
An array of dictionaries with the keys described below, evaluated in the order in which they occur. Required when Action is EvaluateConnection.

- **Domains.** Required. An array of strings that define the domains for which this evaluation applies. Wildcard prefixes are supported, such as *.example.com.
- **DomainAction.** Required. Defines VPN behavior for the Domains. Values for the DomainAction key are:
  - **ConnectIfNeeded.** Brings up VPN if DNS resolution for the Domains fails, such as when the DNS server indicates it can't resolve the domain name, or if the DNS response is redirected, or if the connection fails or times out.
  - **NeverConnect.** Do not trigger VPN for the Domains.

When DomainAction is ConnectIfNeeded, you can also specify the following keys in the connection evaluation dictionary:

- **RequiredDNSServers.** Optional. An array of IP addresses of DNS servers to be used for resolving the Domains. These servers do not need to be part of the device's current network configuration. If these DNS servers are not reachable, VPN will be triggered. Configure an internal DNS server or a trusted external DNS server.
- **RequiredURLStringProbe.** Optional. An HTTP or HTTPS (preferred) URL to probe, using a GET request. If DNS resolution for this server succeeds, the probe must also succeed. If the probe fails, VPN will be triggered.

Notes on backward compatibility
Prior to iOS 7, domain triggering rules were configured via arrays of domains called OnDemandMatchDomainAlways, OnDemandMatchDomainOnRetry, and OnDemandMatchDomainNever. The OnRetry and Never cases are still supported in iOS 7, although deprecated in favor of the EvaluateConnection action.

To create a profile that works on both iOS 7 and earlier releases, use the new EvaluateConnection keys in addition to the OnDemandMatchDomain arrays. Earlier versions of iOS that do not recognize EvaluateConnection will use the old arrays, while iOS 7 and later releases will use EvaluateConnection.

Old configuration profiles that specify the Allow action will work on iOS 7, with the exception of OnDemandMatchDomainsAlways domains.

Per App VPN

iOS 7 adds the ability for VPN connections to be established on a per-app basis. This approach allows for more granular control over which data goes through VPN and which does not. With device-wide VPN, all data travels through the private network regardless of its origin. As more and more personally owned devices are being used within organizations, per App VPN provides secure networking for internal-use apps while preserving the privacy of personal device activity.

Per app VPN allows each app that is managed by mobile device management (MDM) to communicate with the private network via a a secure tunnel, while excluding other non-managed apps on the device from using the private network. Additionally, managed apps can be configured with different VPN connections to further safeguard data. For example, a sales quote app could use an entirely different data center than an accounts payable app, while the user's personal web browsing traffic uses the public Internet. This ability to segregate traffic at the app layer provides for separation of personal data and data belonging to the organization.

In order to use per app VPN, an app must be managed via MDM and use standard iOS networking APIs. Per app VPN is configured with an MDM configuration that specifies which apps and Safari domains are allowed to use the settings. For more information on MDM, refer to Chapter 3: Configuration and Management.

Single Sign On (SSO)

With iOS 7, apps can take advantage of your existing in-house single sign-on infrastructure via Kerberos. Single sign on can improve the user experience by requiring a user to enter their password only one time. It also increases the security of daily app use by ensuring that passwords are never transmitted over the air.

The Kerberos authentication system that is used by iOS 7 is an industry standard and the most commonly deployed single sign-on technology in the world. If you have Active Directory, eDirectory, or OpenDirectory, it's likely to already have a Kerberos system in place that iOS 7 can leverage. iOS devices need to be able to contact the Kerberos service over a network connection to authenticate users.

Supported apps

iOS provides flexible support for Kerberos single sign on (SSO) to any app that uses the NSURLConnection or NSURLSession class to manage network connections and authentication. Apple provides all developers with these high-level frameworks to make network connections seamlessly integrated within their apps. Apple also provides Safari as an example to help you get started by using SSO-enabled websites natively.

Configuring SSO

Configuration of single sign on is accomplished using configuration profiles, which may be either manually installed or managed with MDM. The SSO account payload allows for flexible configuration. SSO can be open to all apps or restricted by either app identifier, service URL, or both.

When matching URLs, simple pattern matching is used and URLs must begin with either http:// or https://. The matching is on the entire URL, so be sure that they are exactly the same. For example, a URLPrefixMatches value of https://www.example.com/ will not match https://www.example.com:443/. You may specify http:// or https:// to restrict the use of SSO to either secure or regular HTTP services. For example, using a URLPrefixMatches value of https:// will allow the SSO account to be used only with secure HTTPS services. If a URL matching pattern does not end with a slash (/), a slash (/) is appended to it.

The AppIdentifierMatches array must contain strings that match app bundle IDs. These strings may be exact matches (com.mycompany.myapp, for example) or may specify a prefix match on the bundle ID by using the wildcard character (*). The wildcard character must appear after a period character (.), and only at the end of the string (for example, com.mycompany.*). When a wildcard is given, any app whose bundle ID begins with the prefix is granted access to the account.

Digital Certificates

Digital certificates are a form of identification that enables streamlined authentication, data integrity, and encryption. A digital certificate is composed of a public key, information about the user, and the certificate authority that issued the certificate. iOS supports digital certificates, giving organizations secure, streamlined access to corporate services.

Certificates can be used in a variety of ways. Signing data with a digital certificate helps to ensure that information cannot be altered. Certificates can also be used to guarantee the identity of the author or "signer." Additionally, they can be used to encrypt configuration profiles and network communications to further protect confidential or private information.

For example, the Safari browser can check the validity of an X.509 digital certificate and set up a secure session with up to 256-bit AES encryption. This verifies that the site's identity is legitimate and that communication with the website is protected to help prevent interception of personal or confidential data.

Using certificates in iOS

Root certificates

Out of the box, iOS includes a number of preinstalled root certificates. For more information, refer to the list in this Apple Support article.

iOS can update certificates wirelessly should any of the preinstalled root certificates become compromised. To disable this, there's a restriction that prevents over-the-air certificate updates.

If you are using a root certificate that is not preinstalled, such as a self-signed root certificate created by your organization, you can distribute it using one of the methods listed below.

Supported certificate and identity formats:

- iOS supports X.509 certificates with RSA keys.
- The file extensions .cer, .crt, .der, .p12, and .pfx are recognized.

Distributing and installing certificates

Distributing certificates to iOS devices is simple. When a certificate is received, users simply tap to review the contents, then tap to add the certificate to their device. When an identity certificate is installed, users are prompted for the password that protects it. If a certificate's authenticity cannot be verified, it will be shown as untrusted, and the user can decide whether they still want to add it to their device.

Installing certificates via configuration profiles

If configuration profiles are being used to distribute settings for corporate services such as Exchange, VPN, or Wi-Fi, certificates can be added to the profile to streamline deployment. This includes the ability to distribute certificates via MDM.

Installing certificates via Mail or Safari

If a certificate is sent in an email, it will appear as an attachment. Safari can also be used to download certificates from a web page. You can host a certificate on a secured website and provide users with the URL where they can download the certificate onto their devices.

Certificate removal and revocation

To manually remove a certificate that has been installed, choose Settings>General> Profiles and choose the appropriate certificate to remove. If a user removes a certificate that is required for accessing an account or network, the device will no longer be able to connect to those services.

A mobile device management server can view all certificates on a device and remove any certificates it has installed.

Additionally, the Online Certificate Status Protocol (OCSP) and CRL (Certificate Revocation List) protocol are supported to check the status of certificates. When an OCSP- or CRL-enabled certificate is used, iOS periodically validates it to make sure that it has not been revoked.

Bonjour

Bonjour is Apple's standards-based, zero configuration network protocol that allows devices to find services on a network. iOS devices use Bonjour to discover AirPrint-compatible printers and AirPlay-compatible devices Some peer-to-peer apps also require Bonjour. You need to be sure that your network infrastructure and Bonjour are correctly configured to work together.

iOS 7.1 devices will also look for AirPlay sources over Bluetooth.

For more information on Bonjour, refer to this Apple web page.

Chapter 2: **Security**

iOS is built with multiple layers of security. This enables iOS devices to securely access network services and protect important data. iOS provides strong encryption for data in transmission, proven authentication methods for access to corporate services, and hardware encryption for all data at rest. iOS also provides secure protection through the use of passcode policies that can be delivered and enforced over the air. And if the device falls into the wrong hands, users and IT administrators can initiate a remote wipe command to erase private information.

When considering the security of iOS for enterprise use, it's helpful to understand the following:

- **Device controls.** Methods that prevent unauthorized use of the device
- **Encryption and data protection.** Protecting data at rest, even when a device is lost or stolen
- **Network security.** Networking protocols and the encryption of data in transmission
- **App security.** Enabling apps to run securely and without compromising platform integrity
- **Internet services.** Apple's network-based infrastructure for messaging, syncing, and backup

These capabilities work in concert to provide a secure mobile computing platform.

Device Security

Establishing strong policies for access to iOS devices is critical to protecting corporate information. Device passcodes are the front line of defense against unauthorized access and can be configured and enforced over the air. iOS devices use the unique passcode established by each user to generate a strong encryption key to further protect mail and sensitive application data on the device. Additionally, iOS provides secure methods to configure the device in an IT environment, where specific settings, policies, and restrictions must be in place. These methods provide flexible options for establishing a standard level of protection for authorized users.

Passcode policies

A device passcode prevents unauthorized users from accessing data or otherwise gaining access to the device. iOS allows you to select from an extensive set of passcode policies to meet your security needs, including timeout periods, passcode strength, and how often the passcode must be changed.

The following passcode policies are supported:

- Require passcode on device
- Require alphanumeric value
- Minimum passcode length
- Minimum number of complex characters
- Maximum passcode age
- Time before auto-lock
- Passcode history
- Grace period for device lock
- Maximum number of failed attempts

Policy enforcement

Policies can be distributed as part of a configuration profile for users to install. A profile can be defined so that deleting the profile is only possible with an administrative password, or you can define the profile so that it is locked to the device and cannot be removed without completely erasing all of the device contents. Additionally, passcode settings can be configured remotely using mobile device management (MDM) solutions that can push policies directly to the device. This enables policies to be enforced and updated without any action by the user.

If the device is configured to access a Microsoft Exchange account, Exchange ActiveSync policies are pushed to the device over the air. The available set of policies will vary depending on the version of Exchange ActiveSync and Exchange Server. If both Exchange and MDM policies exist, whichever policy is more stringent will be applied.

Secure device configuration

Configuration profiles are XML files that contain device security policies and restrictions, VPN configuration information, Wi-Fi settings, email and calendar accounts, and authentication credentials that permit iOS devices to work with your IT systems. The ability to establish passcode policies along with device settings in a configuration profile ensures that devices within your organization are configured correctly and according to security standards set by your IT department. And, because configuration profiles can be encrypted and locked, the settings cannot be removed, altered, or shared with others.

Configuration profiles can be both signed and encrypted. Signing a configuration profile ensures that the settings it enforces cannot be altered in any way. Encrypting a configuration profile protects the profile's contents and permits installation only on the device for which it was created. Configuration profiles are encrypted using CMS (Cryptographic Message Syntax, RFC 3852), supporting 3DES and AES 128.

The first time you distribute an encrypted configuration profile, you can install it via USB using Apple Configurator, or wirelessly using the Over-the-Air Profile Delivery and Configuration protocol, or MDM. Subsequent encrypted configuration profiles can be delivered via email attachment, hosted on a website accessible to your users, or pushed to the device using MDM solutions.

For more information, refer to Over-the-Air Profile Delivery and Configuration protocol on the iOS Developer Library site.

Device restrictions

Device restrictions determine which features your users can access on the device. Typically, these involve network-enabled applications such as Safari, YouTube, or the iTunes Store, but restrictions can also control device functionality such as app installation or use of camera. Restrictions let you configure the device to meet your requirements, while permitting users to utilize the device in ways that are consistent with your organization policies. Restrictions can be manually configured on each device, enforced using a configuration profile, or established remotely with an MDM solution. Additionally, like passcode policies, camera or web-browsing restrictions can be enforced over the air via Microsoft Exchange Server 2007 and 2010. Restrictions can also be used to prevent mail messages from being moved between accounts, or messages received in one account being forwarded to another.

See Appendix B for information about supported restrictions.

Encryption and Data Protection

Protecting data stored on iOS devices is important for any environment with sensitive information. In addition to encrypting data in transmission, iOS devices provide hardware encryption for all data stored on the device, and additional encryption of email and application data with enhanced data protection.

Encryption

iOS devices use hardware-based encryption. Hardware encryption uses 256-bit AES to protect all data on the device. Encryption is always enabled, and cannot be disabled. Additionally, data backed up in iTunes to a user's computer can be encrypted. This can be enabled by the user, or enforced by using device restriction settings in Configuration Profiles. iOS also supports S/MIME in mail, enabling users to view and send encrypted email messages.

The cryptographic modules in iOS 7 and iOS 6 have been validated to comply with U.S. Federal Information Processing Standard (FIPS) 140-2 Level 1. This validates the integrity of cryptographic operations in Apple apps and third-party apps that properly utilize iOS cryptographic services.

For more information, refer to iOS product security: Validations and guidance and iOS 7: Apple FIPS iOS Cryptographic Modules v4.0.

Data protection

Email messages and attachments stored on the device can be further secured by using data protection features built into iOS. Data protection leverages each user's unique device passcode in concert with the hardware encryption on iOS devices to generate a strong encryption key. This prevents data from being accessed when the device is locked, ensuring that critical information is secured even if the device is compromised.

To turn on the data protection feature, simply establish a passcode on the device. The effectiveness of data protection is dependent on a strong passcode, so it is important to require and enforce a passcode stronger than four digits when establishing your passcode policies. Users can verify that data protection is enabled on their device by looking at the passcode settings screen. Mobile device management solutions are able to query the device for this information as well.

Data protection APIs are also available to developers, and can be used to secure data within App Store apps or custom-developed in-house enterprise apps. As of iOS 7, data stored by applications is, by default, in the security class "Protected Until First User Authentication," which is similar to full-disk encryption on desktop computers, and protects data from attacks that involve a reboot.

Note: If a device has been upgraded from iOS 6, existing data stores are not converted to the new class. Removing then reinstalling the app will result in the app receiving the new protection class.

Touch ID

Touch ID is the fingerprint sensing system built into iPhone 5s, making highly secure access to the device faster and easier. This forward-thinking technology reads fingerprints from any angle and learns more about a user's fingerprint over time, with the sensor continuing to expand the fingerprint map as additional overlapping nodes are identified with each use.

Touch ID makes using a longer, more complex passcode far more practical because users won't have to enter it as frequently. Touch ID also overcomes the inconvenience of a passcode-based lock, not by replacing it but rather by securely providing access to the device within thoughtful boundaries and time constraints.

When Touch ID is enabled, iPhone 5s immediately locks when the Sleep/Wake button is pressed. With passcode-only security, many users set an unlocking grace period to avoid having to enter a passcode each time the device is used. With Touch ID, iPhone 5s locks every time it goes to sleep, and requires a fingerprint—or optionally the passcode—at every wake.

Touch ID works in conjunction with the Secure Enclave—a coprocessor fabricated in the Apple A7 chip. The Secure Enclave has its own protected, encrypted memory space and communicates securely with the Touch ID sensor. When iPhone 5s locks, the keys for Data Protection class Complete are protected by a key held in the encrypted memory of the Secure Enclave. The key is held for a maximum of 48 hours and is discarded if iPhone 5s is rebooted or an unrecognized fingerprint is used five times. If a fingerprint is recognized, the Secure Enclave provides the key for unwrapping the Data Protection keys and the device is unlocked.

Remote wipe

iOS supports remote wipe. If a device is lost or stolen, the administrator or device owner can issue a remote wipe command that removes all data and deactivates the device. If the device is configured with an Exchange account, the administrator can initiate a remote wipe command using the Exchange Management Console (Exchange Server 2007) or Exchange ActiveSync Mobile Administration Web Tool (Exchange Server 2003 or 2007). Users of Exchange Server 2007 can also initiate remote wipe commands directly using Outlook Web Access. Remote wipe commands can also be initiated by MDM solutions or using the Find My iPhone feature of iCloud, even if Exchange corporate services are not in use.

Local wipe

Devices can also be configured to automatically initiate a local wipe after several failed passcode attempts. This protects against brute-force attempts to gain access to the device. When a passcode is established, users have the ability to enable local wipe directly within the settings. By default, iOS will automatically wipe the device after 10 failed passcode attempts. As with other passcode policies, the maximum number of failed attempts can be established via a configuration profile, set by an MDM server, or enforced over the air via Microsoft Exchange ActiveSync policies.

Find My iPhone and Activation Lock

If a device is lost or stolen, it's important to deactivate and erase the device. With iOS 7, when Find My iPhone is turned on, the device can't be reactivated without entering the owner's Apple ID credentials. It's a good idea to either supervise institution-owned devices or have a policy in place for users to disable the feature so that Find My iPhone doesn't prevent the organization from assigning the device for use by another individual.

In iOS 7.1 or later, you can use a compatible MDM solution to enable Activation Lock when a user turns on Find My iPhone. Your MDM solution can store a bypass code when Activation Lock is enabled and later use this code to clear Activation Lock automatically when you need to erase the device and deploy it to a new user. Refer to your MDM solution documentation for details.

For more information on Find My iPhone and Activation Lock, refer to iCloud Support and Mobile Device Management and Find My Phone Activation Lock.

Network security
- Built-in Cisco IPSec, L2TP, PPTP VPN
- SSL VPN via App Store apps
- SSL/TLS with X.509 certificates
- WPA/WPA2 Enterprise with 802.1X
- Certificate-based authentication
- RSA SecurID, CRYPTOCard

VPN protocols
- Cisco IPSec
- L2TP/IPSec
- PPTP
- SSL VPN

Authentication methods
- Password (MSCHAPv2)
- RSA SecurID
- CRYPTOCard
- X.509 digital certificates
- Shared secret

802.1X authentication protocols
- EAP-TLS
- EAP-TTLS
- EAP-FAST
- EAP-SIM
- PEAP v0, v1
- LEAP

Supported certificate formats
iOS supports X.509 certificates with RSA keys. The file extensions .cer, .crt, and .der are recognized.

Network Security

Mobile users must be able to access corporate information networks from anywhere in the world, yet it's also important to ensure that users are authorized and that their data is protected during transmission. iOS provides proven technologies to accomplish these security objectives for both Wi-Fi and cellular data network connections.

In addition to your existing infrastructure, each FaceTime session and iMessage conversation is encrypted end to end. iOS creates a unique ID for each user, ensuring communications are encrypted, routed, and connected properly.

VPN
Many enterprise environments have some form of virtual private network (VPN) established. These secure network services are already deployed and typically require minimal setup and configuration to work with iOS. Out of the box, iOS integrates with a broad range of commonly used VPN technologies. See "Virtual Private Networks" in Chapter 1 for details.

SSL/TLS
iOS supports SSL v3 as well as Transport Layer Security (TLS v1.0, 1.1, and 1.2). Safari, Calendar, Mail, and other Internet applications automatically start these mechanisms to enable an encrypted communication channel between iOS and corporate services.

WPA/WPA2
iOS supports WPA2 Enterprise to provide authenticated access to your enterprise wireless network. WPA2 Enterprise uses 128-bit AES encryption, giving users the highest level of assurance that their data will remain protected when they send and receive communications over a Wi-Fi network connection. And with support for 802.1X, iPhone and iPad can be integrated into a broad range of RADIUS authentication environments.

App Security

iOS is designed with security at its core. It includes a sandboxed approach to application runtime protection and application signing to ensure that apps cannot be tampered with. iOS also has a framework that facilitates secure storage of application and network service credentials in an encrypted storage location called the keychain. For developers, it offers a Common Crypto architecture that can be used to encrypt application data stores.

Runtime protection
Apps on the device are sandboxed to restrict access to data stored by other apps. In addition, system files, resources, and the kernel are shielded from the user's application space. If an app needs to access data from another app, it can only do so using the APIs and services provided by iOS. Code generation is also prevented.

Mandatory code signing
All iOS apps must be signed. The apps provided with the device are signed by Apple. Third-party apps are signed by the developer using an Apple-issued certificate. This ensures that apps haven't been tampered with or altered. Additionally, runtime checks are made to ensure that an app hasn't become untrusted since it was last used.

The use of custom developed in-house enterprise apps can be controlled with a provisioning profile. Users must have the provisioning profile installed to execute the application. Provisioning profiles can be installed over the air using MDM solutions. Administrators can also restrict the use of an application to specific devices.

Secure authentication framework

iOS provides a secure, encrypted keychain for storing digital identities, user names, and passwords. Keychain data is partitioned so that credentials stored by third-party apps cannot be accessed by apps with a different identity. This provides the mechanism for securing authentication credentials on iOS devices across a range of applications and services within the enterprise.

Common Crypto architecture

Application developers have access to encryption APIs that they can use to further protect their application data. Data can be symmetrically encrypted using proven methods such as AES, RC4, or 3DES. In addition, iOS devices provide hardware acceleration for AES encryption and SHA1 hashing, maximizing application performance.

Application data protection

Apps can also take advantage of the built-in hardware encryption on iOS devices to further protect sensitive application data. Developers can designate specific files for data protection, instructing the system to make the contents of the file cryptographically inaccessible to both the app and any potential intruders when the device is locked.

App entitlements

By default, an iOS app has very limited privileges. Developers must explicitly add entitlements to use most features such as iCloud, background processing, or shared keychains. This ensures that apps can't grant themselves data access they weren't deployed with. Additionally, iOS apps must ask for explicit user permission before using many iOS features such as GPS location, user contacts, the camera, or stored photos.

Internet Services

Apple has built a robust set of services to help users get even more utility and productivity out of their devices, including iMessage, FaceTime, Siri, iCloud, iCloud Backup, and iCloud Keychain.

These Internet services have been built with the same security goals that iOS promotes throughout the platform. These goals include secure handling of data, whether at rest on the device or in transit over wireless networks; protection of users' personal information; and threat protection against malicious or unauthorized access to information and services. Each service uses its own powerful security architecture without compromising the overall ease of use of iOS.

iMessage

iMessage[1] is a messaging service for iOS devices and Mac computers. iMessage supports text and attachments such as photos, contacts, and locations. Messages appear on all of a user's registered devices so that a conversation can be continued from any of the user's devices. iMessage makes extensive use of the Apple Push Notification Service (APNs). iMessage uses end-to-end encryption that utilizes keys known only to the sending and receiving devices. Apple cannot decrypt messages, and messages are not logged.

FaceTime

FaceTime[2] is Apple's video and audio calling service. FaceTime calls use the Apple Push Notification service to establish an initial connection, then utilize Internet Connectivity Establishment (ICE) and Session Initiation Protocol (SIP) to create an encrypted stream.

Siri

By simply talking naturally, users can enlist Siri[3] to send messages, schedule meetings, place phone calls, and more. Siri uses speech recognition, text-to-speech, and a client-server model to respond to a broad range of requests. The tasks that Siri supports have been designed to ensure that only the absolute minimal amount of personal information is utilized and that it is fully protected. Siri requests and voice recordings are not personally identified, and whenever possible Siri functions are carried out on the device, not the server.

iCloud

iCloud[4] stores music, photos, apps, calendars, documents, and more, and automatically pushes them to all of a user's devices. iCloud also backs up information, including device settings, app data, and text and MMS messages, daily over Wi-Fi. iCloud secures your content by encrypting it when sent over the Internet, storing it in an encrypted format, and using secure tokens for authentication. Additionally, iCloud features, including Photo Stream, Document & Data, and Backup, can be disabled via a configuration profile.

For more information on iCloud security and privacy, refer to the iCloud security and privacy overview.

iCloud Backup

iCloud also backs up information—including device settings, app data, and text and MMS messages—daily over Wi-Fi. iCloud secures the content by encrypting it when sent over the Internet, storing it in an encrypted format, and using secure tokens for authentication. iCloud Backup occurs only when the device is locked, connected to a power source, and has Wi-Fi access to the Internet. Because of the encryption used in iOS, the system is designed to keep data secure while allowing incremental, unattended backup and restoration to occur.

iCloud Keychain

iCloud Keychain allows users to securely sync their passwords between iOS devices and Mac computers without exposing that information to Apple. In addition to strong privacy and security, other goals that heavily influenced the design and architecture of iCloud Keychain were ease of use and the ability to recover a keychain. iCloud Keychain consists of two services: keychain syncing and keychain recovery. In keychain syncing, devices are allowed to participate only after user approval, and each keychain item that is eligible to be synced is exchanged with per-device encryption via iCloud key value storage. The items are ephemeral and do not persist in iCloud after being synced. Keychain recovery provides a way for users to escrow their keychain with Apple, without giving Apple the ability to read the passwords and other data it contains. Even if the user has only a single device, keychain recovery provides a safety net against data loss. This is particularly important when Safari is used to generate random, strong passwords for web accounts, as the only record of those passwords is in the keychain. A cornerstone of keychain recovery is secondary authentication and a secure escrow service, created by Apple specifically to support this feature. The user's keychain is encrypted using a strong passcode, and the escrow service will only provide a copy of the keychain if a strict set of conditions are met.

Details about security are provided in the iOS Security Guide.

Chapter 3: **Configuration and Management**

iOS deployments can be streamlined through several management techniques that simplify account setup, configure institutional policies, distribute apps, and apply device restrictions. Users can then do most of the configuration work themselves through the Setup Assistant built into iOS. And after iOS devices are configured and enrolled in MDM, they can be wirelessly managed by IT.

This chapter describes how to use configuration profiles and mobile device management to support your iOS deployment.

Device Setup and Activation

Out of the box, iOS lets users activate their device, configure basic settings, and start working right away with the Setup Assistant in iOS. Beyond basic settings, users can customize their personal preferences like language, location, Siri, iCloud, and Find My iPhone. The Setup Assistant also enables the user to create a personal Apple ID if they do not have one already.

Apple ID

An Apple ID is an identity that is used to log into various Apple services such as FaceTime, iMessage, iTunes, the App Store, iCloud, and the iBooks Store. With an Apple ID, each user can install apps, books, and content from the iTunes Store, App Store, or iBooks Store. An Apple ID also allows users to sign up for an iCloud account that they can use to access and share content on multiple devices.

In order to get the most out of these services, users should use their own personal Apple ID. If they do not have one, they can create one even before they are provided a device so that configuration can be as quick as possible.

Learn how to sign up for an Apple ID at My Apple ID.

Preparing devices with Apple Configurator

For devices that are centrally managed by IT and are not set up by individual users, Apple Configurator can be used to quickly activate devices, define and apply configurations, supervise devices, install apps, and update devices to the latest iOS version. Apple Configurator is a free application for OS X, available for download from the Mac App Store. Devices need to be connected to a Mac via USB to perform these tasks. You can also restore a backup to devices, which applies device settings and Home screen layout, and installs app data.

Configuration Profiles

Configuration profiles are XML files that contain device security policies and restrictions, VPN configuration information, Wi-Fi settings, email and calendar accounts, and authentication credentials that permit iOS devices to work with your IT systems. Configuration profiles quickly load settings and authorization information onto a device. Some VPN and Wi-Fi settings can be set only by using a configuration profile, and if you're not using Microsoft Exchange, you need to use a configuration profile to set device passcode policies.

Configuration profiles can be distributed using Over-the-Air Profile Delivery or via mobile device management (MDM). You can also install configuration profiles on devices connected to a computer via USB using Apple Configurator, or you can distribute configuration profiles by email or on a web page. When users open the email attachment or download the profile using Safari on their device, they're prompted to begin the installation process. If you're using an MDM server, you can distribute an initial profile that contains just the server configuration information, then have the device obtain all other profiles wirelessly.

Configuration profiles can be encrypted and signed, which lets you restrict their use to a specific device and prevents anyone from changing the settings that a profile contains. You can also mark a profile as being locked to the device, so once installed, it can be removed only by wiping the device of all data or, optionally, by entering a passcode.

With the exception of passwords, users cannot change the settings provided in a configuration profile. Additionally, accounts that are configured by a profile, such as Exchange accounts, can only be removed by deleting the profile.

For more information, refer to Configuration Profile Key Reference on the iOS Developer Library site.

Mobile Device Management (MDM)

iOS has a built-in MDM framework that allows third-party MDM solutions to wirelessly interact with iOS devices. This lightweight framework was designed from the ground up for iOS devices, and is powerful and scalable enough to fully configure and manage all the iOS devices within an organization.

With an MDM solution in place, IT administrators can securely enroll devices in an enterprise environment, configure and update settings, monitor compliance with corporate policies, and remotely wipe or lock managed devices. iOS MDM gives IT a simple way to enable user access to network services while ensuring devices are properly configured—regardless of who owns them.

MDM solutions use the Apple Push Notification service (APNs) to maintain persistent communication with devices across both public and private networks. MDM requires multiple certificates to operate, including an APNs certificate to talk to clients and an SSL certificate to communicate securely. MDM solutions may also sign profiles with a certificate.

Most certificates, including an APNs certificate, must be renewed annually. When a certificate expires, an MDM server cannot communicate with clients until the certificate is updated. Be prepared to update all MDM certificates before they expire.

See the Apple Push Certificates Portal for more information about MDM certificates.

Enroll

Enrollment of devices enables cataloging and asset management. The enrollment process can leverage Simple Certificate Enrollment Protocol (SCEP), which allows iOS devices to create and enroll unique identity certificates for authentication to institution services.

In most cases, users decide whether or not to enroll their device in MDM, and they can disassociate from MDM at any time. Institutions should consider incentives for users to remain managed. For example, require MDM enrollment for Wi-Fi network access by using the MDM solution to automatically provide the wireless credentials. When a user leaves MDM, the device attempts to notify the MDM server.

Configuration

Once a device is enrolled, it can be dynamically configured with settings and policies by the mobile device management server, which sends configuration profiles to the device that are automatically, and silently, installed by the device.

Configuration profiles can be signed, encrypted, and locked—preventing the settings from being altered or shared—ensuring that only trusted users and devices that are configured to your specifications can access your network and services. If a user disassociates the device from MDM, all of the settings installed via MDM are removed.

Accounts

Mobile device management can help your organization's users get up and running quickly by setting up their mail and other accounts automatically. Depending on the MDM product and integration with your internal systems, account payloads can also be pre-populated with a user's name, mail address, and, where applicable, certificate identities for authentication and signing. MDM can configure the following types of accounts:

- Mail
- Calendar
- Subscribed Calendars
- Contacts
- Exchange ActiveSync
- LDAP

Managed mail and calendar accounts respect the new Managed Open In restrictions in iOS 7.

Queries

A mobile device management server has the ability to query devices for a variety of information. This includes hardware information such as serial number, device UDID, or Wi-Fi MAC address, and software information such as the iOS version and a detailed list of all apps installed on the device. This information can be used to help ensure that users maintain the appropriate set of apps.

Commands

When a device is managed, it can be administered by the mobile device management server through a set of specific actions. Management tasks include:

- **Changing configuration settings**. A command can be sent to install a new or updated configuration profile on a device. Configuration changes happen silently without user interaction.
- **Lock a device**. If a device needs to be locked immediately, a command can be sent to lock the device with the currently set passcode.
- **Remotely wiping a device**. If a device is lost or stolen, a command can be sent to erase all of the data on the device. Once a remote wipe command is received, it cannot be undone.
- **Clearing a passcode lock**. Clearing a passcode sets the device so that it immediately requires the user to enter a new passcode. This is used when a user has forgotten their passcode and wants IT to reset it for them.
- **Request AirPlay Mirroring and Stop AirPlay Mirroring**. iOS 7 adds a command to prompt a supervised iOS device to begin AirPlay mirroring to a specific destination or end a current AirPlay session.

Managed apps

Organizations often need to distribute software so their users are productive at work or in the classroom. At the same time, organizations need to control how that software connects to internal resources or how data security is handled when a user transitions out of the organization, all while coexisting alongside the user's personal apps and data. Managed apps in iOS 7 allow an organization to distribute enterprise apps over the air using MDM, while providing the right balance between institutional security and user personalization.

MDM servers can deploy both App Store apps and in-house enterprise apps to devices over the air.

Managed apps can be removed remotely by the MDM server or when the user removes their own device from MDM. Removing the app also removes the data associated with the removed app.

The iOS 7 and mobile device management add a suite of additional restrictions and capabilities to managed apps in iOS 7 to provide improved security and a better user experience:

- **Managed Open In**. Provides two useful functions for protecting an organization's app data:
  - **Allow documents created using unmanaged apps to open in managed apps**. Enforcing this restriction prevents a user's personal apps and accounts from opening documents in the organization's managed apps. For example, this restriction could prevent a user's copy of Keynote from opening a presentation PDF in an organization's PDF viewing app. This restriction could also prevent a user's personal iCloud account from opening a word processing document attachment in an organization's copy of Pages.
  - **Allow documents created using managed apps to open in unmanaged apps**. Enforcing this restriction prevents an organization's managed apps and accounts from opening documents in a user's personal apps. This restriction could prevent a confidential email attachment in the organization's managed mail account from being opened in any of the user's personal apps.
- **App Configuration**. App developers can identify app settings that can be set when installed as a managed app. These configuration settings can be installed before or after the managed app is installed.

- **App Feedback**. App developers building apps can identify app settings that can be read from a managed app using MDM. For example, a developer could specify a "DidFinishSetup" key for app feedback that an MDM server could query to determine if the app had been launched and set up.
- **Prevent Backup**. This restriction prevents managed apps from backing up data to iCloud or iTunes. Disallowing backup prevents managed app data from being recovered if the app is removed via MDM but later reinstalled by the user.

Device Supervision

By default, all iOS devices are non-supervised. To enable additional configuration options and restrictions, you may choose to supervise iOS devices owned by your organization using Apple Configurator.

After devices have been assigned to an MDM server in the program, profiles and additional features may be applied using your organization's MDM server.

These features include:

- Device supervision
- Mandatory configuration
- Lockable MDM settings
- Skipping steps in the Setup Assistant

Setup Assistant screens that can be skipped include:

- **Passcode.** Skips the passcode setup
- **Location**. Does not enable Location Services
- **Restore from backup.** Does not restore from backup
- **Apple ID.** Does not prompt you to sign in with an Apple ID
- **Terms of Service.** Skips the Terms of Service
- **Siri.** Does not enable Siri
- **Sending diagnostics.** Does not automatically send diagnostic information

Supervised devices

Supervision provides a higher level of device management for devices that are owned by your organization, allowing additional restrictions such as turning off iMessage or Game Center. It also provides additional device configurations and features such as web content filtering, or the ability to silently install apps. Supervision can be wirelessly enabled on the device using Apple Configurator.

See Appendix B for specific restrictions that can be enabled on supervised devices.

Chapter 4: App Distribution

iOS comes with a collection of apps that let people in your organization do all the everyday things they need to do—like email, managing their calendars, keeping track of contacts, and consuming content over the web. Much of the functionality users need to be productive comes from the hundreds of thousands of third-party iOS apps available from the App Store, or custom-developed in-house enterprise apps.

You can create and deploy your own in-house apps if you're a member of the iOS Developer Enterprise Program. This chapter describes the method you can use to deploy apps to your users.

In-House Apps

If you develop your own iOS apps for use by your organization, the iOS Developer Enterprise Program lets you deploy the in-house apps. The process for deploying an in-house app is:

- Register for the iOS Developer Enterprise Program.
- Prepare your app for distribution.
- Create an enterprise distribution provisioning profile that authorizes devices to use apps you've signed.
- Build the app with the provisioning profile.
- Deploy the app to your users.

Registering for app development

To develop and deploy in-house apps for iOS, first register for the iOS Developer Enterprise Program.

Once you register, you can request a developer certificate and developer provisioning profile. You use these during development to build and test your app. The development provisioning profile allows apps signed with your developer certificate to run on registered devices. You create the developer provisioning profile at the iOS Provisioning Portal. The ad hoc profile expires after three months and specifies which devices (by device ID) can run development builds of your app. You distribute your developer signed build and the development provisioning profile to your app team and testers.

Preparing apps for distribution

After you finish development and testing and are ready to deploy your app, you sign your app using your distribution certificate and package it with a provisioning profile. The designated Team Agent or the Admin for your program membership creates the certificate and profile at the iOS Provisioning Portal.

Generating the distribution certificate involves using the Certificate Assistant (which is part of the Keychain Access application on your OS X development system) to generate a Certificate Signing Request (CSR). You upload the CSR to the iOS Provisioning Portal and receive a distribution certificate in response. When you install this certificate in Keychain, you can set Xcode to use it to sign your app.

Provisioning in-house apps

The enterprise distribution provisioning profile allows your app to be installed on an unlimited number of iOS devices. You can create an enterprise distribution provisioning profile for a specific app, or for multiple apps.

Once you have both the enterprise distribution certificate and provisioning profile installed on your Mac, you use Xcode to sign and build a release/production version of your app. Your enterprise distribution certificate is valid for three years, after which you'll have to sign and build your app again using a renewed certificate. The provisioning profile for the app is good for one year, so you'll want to release new provisioning profiles annually. See "Providing Updated Apps" in Appendix C for further details.

It's very important that you limit access to your distribution certificate and its private key. Use Keychain Access on OS X to export and back up these items in p12 format. If the private key is lost, it cannot be recovered or downloaded a second time. In addition to keeping the certificate and private key safe, you should restrict access to personnel who are responsible for final acceptance of the app. Signing an app with the distribution certificate gives your company's seal of approval for the app's content, function, and adherence to the Enterprise Developer Agreement licensing terms.

Deploying Apps

There are four ways to deploy an app:

- Distribute the app for your users to install using iTunes.
- Have an IT administrator install the app on devices using Apple Configurator.
- Post the app on a secure web server; users access and perform the installation wirelessly. See "Appendix C: Installing In-House Apps Wirelessly."
- Use your MDM server to instruct managed devices to install an in-house or App Store app, if your MDM server supports it.

Installing apps using iTunes

If your users install apps on their devices with iTunes, securely distribute the app to the users and have them follow these steps:

1. In iTunes, choose File>Add to Library, and then select the file (.app, .ipa, or .mobileprovision). The user can also drag the file to the iTunes application icon.
2. Connect a device to the computer, and then select it in the Devices list in iTunes.
3. Click the Apps tab, and then select the app in the list.
4. Click Apply.

If your users' computers are managed, instead of asking them to add the files to iTunes, you can deploy the files to their computers and ask them to sync their device. iTunes automatically installs the files found in iTunes Mobile Applications and Provisioning Profiles folders.

Installing apps using Apple Configurator

Apple Configurator, a free application for OS X available for download from the Mac App Store, can be used by IT administrators to install in-house apps—or apps from the App Store.

Apps from the App Store or in-house enterprise apps can be imported directly into Apple Configurator and installed on as many devices as you choose.

Installing apps using MDM

An MDM server can manage third-party apps from the App Store, as well as in-house apps. Apps installed using MDM are called "managed apps." The MDM server can specify whether managed apps and their data remain behind when the user un-enrolls from MDM. Additionally, the server can prevent managed app data from being backed up to iTunes and iCloud. This allows IT to manage apps that may contain sensitive business information with more control than apps downloaded directly by the user.

In order to install a managed app, the MDM server sends an installation command to the device. On unsupervised devices, managed apps require a user's acceptance before they are installed.

Managed apps benefit from additional controls with iOS 7. VPN connections can now be specified at the app layer, meaning that only the network traffic for that app will be in the protected VPN tunnel. This ensures that private data remains private, and public data does not commingle with it.

Managed apps also support Managed Open In on iOS 7. This means that managed apps can be restricted from transferring data to or from the user's personal apps, which enables the enterprise to ensure sensitive data remains where it needs to be.

Caching Server

iOS makes it easy for users to access and consume digital content, and some users may request many gigabytes of apps, books and software updates while connected to an organization's wireless network. The demand for these assets comes in spikes, first with initial device deployment and then sporadically as users discover new content or as content is updated over time. The result of these content downloads can cause surges in demand for Internet bandwidth.

The Caching Server feature included in OS X Server reduces outbound Internet bandwidth on private networks (RFC1918) by storing cached copies of requested content on the local area network. Larger networks benefit from having multiple Caching Servers in place. For many deployments, configuring Caching Server is as simple as turning on the service. A NAT environment for the server and all devices that utilize it is required.

For more information, refer to OS X Server: Advanced Administration.

iOS devices running iOS 7 will automatically contact a nearby Caching Server without any additional device configuration. Here is how the Caching Server workflow behaves transparently to the iOS device:

1. When an iOS device on a network with one or more Caching Servers requests content from the iTunes Store or Software Update server, the iOS device is referred to a Caching Server.
2. The Caching Server will first check to see whether it already has the requested content in its local cache. If it does, it will begin immediately serving the content to the iOS device.
3. If the Caching Server does not have the requested asset, it attempts to download the content from another source. Caching Server 2 for OS X Mavericks includes a peer replication feature that can use other Caching Servers on the network if those servers have already downloaded the requested content.
4. As the Caching Server receives download data, it relays it immediately to any clients that have requested it and simultaneously caches a copy to disk.

The following types of cached content are supported by iOS 7:

- iOS Software Updates
- App Store apps
- App Store updates
- Books from the iBooks Store

iTunes also supports Caching Server 2. The following types of content are supported by iTunes 11.0.4 or later (both Mac and Windows):

- App Store apps
- App Store updates
- Books from the iBooks Store

Country Restrictions

Due to distribution licensing and tax laws, certain content may not be cache-able in certain countries. As of December, 2013, iTunes downloads are not cached in Brazil, Mexico, China, and Portugal, and iBooks downloads are not cached in Canada.

If the country a client is in, based on its IP address, is different from the country of the iTunes Store being used, iTunes downloads will not be cached.

For example, an iPad user in San Francisco can download an app from the German iTunes Store, but will not be able to utilize the caching service.

Appendix A: **Wi-Fi Infrastructure**

When preparing the Wi-Fi infrastructure for an iOS deployment, there are several factors to consider:

- Required coverage area
- Number and density of devices using the Wi-Fi network
- Types of devices and their Wi-Fi capabilities
- Types and amount of data being transferred
- Security requirements for accessing the wireless network
- Encryption requirements

Although this list is not exhaustive, it represents some of the most relevant Wi-Fi network design factors.

Reminder: This chapter focuses on Wi-Fi network design in the United States. This design may differ for other countries.

Planning for coverage and density

Although it is critical to provide Wi-Fi coverage where iOS devices will be used, it is also essential to plan for the density of devices in a given area.

Most modern, enterprise-class access points are capable of handling up to 50 Wi-Fi clients, although the user experience would likely be disappointing if a single access point had that many devices associated to it. The experience on each device depends on the available wireless bandwidth on the channel in use and the number of devices sharing the overall bandwidth. As more and more devices use the same access point, the relative network speed for those devices decreases. You should consider the expected usage pattern of the iOS devices as part of your Wi-Fi network design.

2.4GHz vs. 5GHz

Wi-Fi networks operating at 2.4GHz allow for 11 channels in the United States. However, due to channel interference considerations, only channels 1, 6, and 11 should be used in a network design.

5GHz signals do not penetrate walls and other barriers as well as 2.4GHz signals, which results in a smaller coverage area. Therefore, 5GHz networks may be preferred when you design for a high density of devices in an enclosed space, such as in classrooms. The number of channels available in the 5GHz band varies among vendors of access points and from country to country, but at least eight channels will always be available.

5GHz channels are non-overlapping, which is a significant departure from the three non-overlapping channels available in the 2.4GHz band. When designing a Wi-Fi network for a high density of iOS devices, the additional channels provided at 5GHz become a strategic planning consideration.

Designing for coverage

The physical layout of the building may have an impact on your Wi-Fi network design. For example, in a business environment, users may meet with other employees in conference rooms or in offices. As a result, users move around the building throughout the day. In this scenario, the majority of network access comes from checking email, calendars, and Internet browsing so Wi-Fi coverage is the highest priority. A Wi-Fi design could include two or three access points on each floor to provide coverage for the offices, and one access point in each conference room.

Designing for density

Contrast the scenario above with a school that has 1000 students and 30 teachers in a two-story building. Every student has been issued an iPad, and every teacher has been issued both a MacBook Air and an iPad. Each classroom holds approximately 35 students, and classrooms are adjacent to each other. Throughout the day, students conduct research on the Internet, watch curriculum videos, and copy files to and from a file server on the LAN.

The Wi-Fi network design for this scenario is more complex due to the higher density of mobile devices. Because each classroom has approximately 35 students, one access point per classroom could be deployed. Multiple access points should be considered for the common areas to provide adequate coverage. The actual number of access points for the common areas will vary, depending on the density of Wi-Fi devices in those spaces.

If devices that only support the 802.11b or 802.11g standards are required to participate on the network, one option is to simply enable 802.11b/g if dual-band access points are being deployed. Another option is to provision one SSID using 802.11n at 5GHz for newer devices and a second SSID at 2.4GHz to support 802.11b and 802.11g devices. However, care should be taken to avoid creating too many SSIDs.

The use of hidden SSIDs should be avoided in either design scenario. It is harder for a Wi-Fi device to rejoin a hidden SSID than a broadcast SSID, and there's very little security benefit in hiding the SSID. Users tend to frequently change location along with their iOS devices, so hidden SSIDs may delay network association time.

Wi-Fi standards in Apple products

Support in Apple products for the various Wi-Fi specifications are detailed in the list that follows, which includes this information:

- **802.11 compatibility.** 802.11b/g, 802.11a, 802.11n
- **Frequency band.** 2.4GHz or 5GHz
- **MCS index.** The Modulation and Coding Scheme (MCS) index defines the maximum transmit rate at which 802.11n devices can communicate.
- **Channel bonding.** HD20 or HD40

- **Guard interval (GI).** The guard interval is the space (time) between symbols transmitted from one device to another. The 802.11n standard defines a short guard interval of 400ns that allows for faster overall throughput, but devices may utilize a long guard interval of 800ns.

iPhone 5s
802.11n @ 2.4GHz and 5GHz
802.11a/b/g
MCS Index 7 / HT40 / 400ns GI

iPhone 5c
802.11n @ 2.4GHz and 5GHz
802.11a/b/g
MCS Index 7 / HT40 / 400ns GI

iPhone 5
802.11n @ 2.4GHz and 5GHz
802.11a/b/g
MCS Index 7 / HD40 / 400ns GI

iPhone 4s
802.11n @ 2.4GHz
802.11b/g
MCS Index 7 / HD20 / 800ns GI

iPhone 4
802.11n @ 2.4GHz
802.11b/g
MCS Index 7 / HD20 / 800ns GI

iPad Air and iPad mini with Retina display
802.11n @ 2.4GHz and 5GHz
802.11 a/b/g
MCS Index 15 / HT40 / 400ns GI

iPad (4th Generation) and iPad mini
802.11n @ 2.4GHz and 5GHz
802.11a/b/g
MCS Index 7 / HD40 / 400ns GI

iPad (1st, 2nd, and 3rd Generation)
802.11n @ 2.4GHz and 5GHz
802.11a/b/g
MCS Index 7 / HD20 / 800ns GI

iPod touch (5th Generation)
802.11n @ 2.4GHz and 5GHz
802.11a/b/g
MCS Index 7 / HD40 / 400ns GI

iPod touch (4th Generation)
802.11n @ 2.4GHz
802.11b/g
MCS Index 7 / HD20 / 800ns GI

Appendix B: **Restrictions**

iOS supports the following policies and restrictions, all of which can be configured to meet the needs of your organization.

Device functionality

- Allow installing apps
- Allow Siri
- Allow Siri while locked
- Allow use of camera
- Allow FaceTime
- Allow screen capture
- Allow automatic syncing while roaming
- Allow syncing of Mail recents
- Allow voice dialing
- Allow In-App Purchase
- Require store password for all purchases
- Allow multiplayer gaming
- Allow adding Game Center friends
- Set allowed content ratings
- Allow Touch ID
- Allow ControlCenter access from Lock screen
- Allow Notification Center access from Lock screen
- Allow Today view from Lock screen
- Allow Passbook notifications in Lock Screen

Applications

- Allow use of iTunes Store
- Allow use of Safari
- Set Safari security preferences

iCloud

- Allow backup
- Allow document sync and keychain sync
- Allow My Photo Stream
- Allow iCloud photo sharing

Security and privacy

- Allow diagnostic data to be sent to Apple
- Allow user to accept untrusted certificates
- Force encrypted backups
- Allow Open from Unmanaged to Managed apps
- Allow Open from Managed to Unmanaged apps
- Require passcode on first AirPlay pairing
- Allow over-the-air PKI updates
- Require Limit Ad Tracking

Supervised-only restrictions

- Single App Mode only
- Accessibility settings
- Allow iMessage
- Allow Game Center
- Allow removal of apps
- Allow iBooks Store
- Allow erotica from iBooks Store
- Enable Siri Profanity Filter
- Allow manual install of configuration profiles
- Global network proxy for HTTP
- Allow pairing to computers for content sync
- Restrict AirPlay connections with whitelist and optional connection passcodes
- Allow AirDrop
- Allow account modification
- Allow cellular data settings modification
- Allow Find My Friends
- Allow host pairing (iTunes)
- Allow Activation Lock

Appendix C: **Installing In-House Apps Wirelessly**

iOS supports over-the-air installation of custom-developed in-house apps without using iTunes or the App Store.

Requirements:

- A secure web server that authenticated users can access
- An iOS app in .ipa format, built for release/production with an enterprise provisioning profile
- An XML manifest file, described in this appendix
- A network configuration that allows the devices to access an iTunes server at Apple

Installing the app is simple. Users download the manifest file from your website to their iOS device. The manifest file instructs the device to download and install the apps referenced in the manifest file.

You can distribute the URL for downloading the manifest file by SMS or email, or by embedding it in another enterprise app you create.

It's up to you to design and host the website used to distribute apps. Make sure that users are authenticated, perhaps using basic auth or directory-based authentication, and that the website is accessible via your intranet or the Internet. You can place the app and manifest file in a hidden directory, or in any other location that's readable using HTTPS.

If you create a self-service portal, consider adding a Web Clip to the user's Home screen so it's easy to direct them back to the portal for future deployment information such as new configuration profiles, recommended App Store apps, and enrollment in a mobile device management solution.

Preparing an in-house app for wireless distribution

To prepare your in-house app for wireless distribution, you build an archived version (an .ipa file), and a manifest file that enables wireless distribution and installation of the app.

You use Xcode to create an app archive. Sign the app using your distribution certificate and include your enterprise deployment provisioning profile in the archive. For more information about building and archiving apps, visit the iOS Dev Center or refer to the *Xcode User Guide,* available from the Help menu in Xcode.

About the wireless manifest file

The manifest file is an XML plist. It's used by an iOS device to find, download, and install apps from your web server. The manifest file is created by Xcode, using information you provide when you share an archived app for enterprise distribution. See the previous section about preparing apps for distribution.

The following fields are required:

Item	Description
URL	The fully qualified HTTPS URL of the app (.ipa) file.
display-image	A 57-by-57-pixel PNG image that's displayed during download and installation. Specify the image's fully qualified URL.
full-size-image	A 512-by-512-pixel PNG image that represents the app in iTunes.
bundle-identifier	Your app's bundle identifier, exactly as specified in your Xcode project.
bundle-version	Your app's bundle version, as specified in your Xcode project.
title	The name of the app, which is displayed during download and installation.

For Newsstand apps only, the following fields are required:

Item	Description
newsstand-image	A full-size PNG image for display on the Newsstand shelf.
UINewsstandBindingEdge UINewsstandBindingType	These keys must match those in your Newsstand app's info.plist.
UINewsstandApp	Indicates that the app is a Newsstand app.

Optional keys you can use are described in the sample manifest file. For example, you can use the MD5 keys if your app file is large and you want to ensure download integrity beyond the error checking normally done for TCP communications.

You can install more than one app with a single manifest file by specifying additional members of the items array.

A sample manifest file is included at the end of this appendix.

Constructing your website

Upload these items to an area of your website that your authenticated users can access:

• The app (.ipa) file

• The manifest (.plist) file

Your website design can be as simple as a single page that links to the manifest file. When a user taps a web link, the manifest file is downloaded, which triggers the downloading and installation of the apps it describes.

Here's a sample link: <a href="itms-services://?action=download-manifest&url=http://example.com/manifest.plist">Install App</a>

Don't add a web link to the archived app (.ipa). The .ipa file is downloaded by the device when the manifest file is loaded. Although the protocol portion of the URL is itms-services, the iTunes Store isn't involved in this process.

Also ensure that your .ipa file is accessible over HTTPS and that your site is signed with a certificate that's trusted by iOS. Installation will fail if a self-signed certificate doesn't have a trusted anchor and can't be validated by the iOS device.

Setting server MIME types

You may need to configure your web server so the manifest file and app file are transmitted correctly.

For OS X Server, add the following MIME types to the web service's MIME Types settings:

application/octet-stream ipa
text/xml plist

For IIS, use IIS Manager to add the MIME type in the Properties page of the server:

.ipa application/octet-stream
.plist text/xml

Troubleshooting wireless app distribution

If wireless app distribution fails with an "unable to download" message, check the following:

- Make sure the app is signed correctly. Test it by installing it on a device using Apple Configurator, and see if any errors occur.
- Make sure the link to the manifest file is correct and the manifest file is accessible to web users.
- Make sure the URL to the .ipa file (in the manifest file) is correct and the .ipa file is accessible to web users over HTTPS.

Network configuration requirements

If the devices are connected to a closed internal network, you should let iOS devices access the following:

URL	Reason
ax.init.itunes.apple.com	The device obtains the current file-size limit for downloading apps over the cellular network. If this site isn't reachable, installation may fail.
ocsp.apple.com	The device contacts this site to check the status of the distribution certificate used to sign the provisioning profile. See "Certificate Validation" below.

Providing updated apps

Apps you distribute yourself aren't automatically updated. When you have a new version for users to install, notify them of the update and instruct them to install the app. Consider having the app check for updates and notify the user when it opens. If you're using wireless app distribution, the notification can provide a link to the manifest file of the updated app.

If you want users to keep the app's data stored on their device, make sure the new version uses the same bundle-identifier as the one it's replacing, and tell users not to delete their old version before installing the new one. The new version will replace the old one and keep data stored on the device, if the bundle-identifiers match.

Distribution provisioning profiles expire 12 months after they're issued. After the expiration date, the profile is removed and the app won't launch.

Before a provisioning profile expires, use the iOS Development Portal to create a new profile for the app. Create a new app archive (.ipa) with the new provisioning profile, for users who are installing the app for the first time.

For users who already have the app, you may want to time your next released version so that it includes the new provisioning profile. If not, you can distribute just the new .mobileprovision file so users won't have to install the app again. The new provisioning profile will override the one that's already in the app archive.

Provisioning profiles can be installed and managed using MDM, downloaded and installed by users from a secure website that you provide, or distributed to users as an email attachment to open and install.

When your distribution certificate expires, the app won't launch. Your distribution certificate is valid for three years from when it was issued, or until your Enterprise Developer Program membership expires, whichever comes first. To prevent the premature expiration of your certificate, be sure to renew your membership before it expires. For information about how the distribution certificate is checked, see "Certificate Validation" below.

You can have two distribution certificates active at the same time; each is independent from the other. The second certificate is intended to provide an overlapping period during which you can update your apps before the first certificate expires. When requesting your second distribution certificate from the iOS Dev Center, be sure you don't revoke your first certificate.

Certificate validation

The first time a user opens an app, the distribution certificate is validated by contacting Apple's OCSP server. Unless the certificate has been revoked, the app is allowed to run. Inability to contact or get a response from the OCSP server isn't interpreted as a revocation. To verify the status, the device must be able to reach ocsp.apple.com. See "Network Configuration Requirements," earlier in this appendix.

The OCSP response is cached on the device for the period of time specified by the OCSP server—currently, between three and seven days. The validity of the certificate isn't checked again until the device has restarted and the cached response has expired.

If a revocation is received at that time, the app is prevented from running.

Revoking a distribution certificate invalidates all of the apps you've signed with it. You should revoke a certificate only as a last resort—if you're sure the private key is lost or the certificate is believed to be compromised.

Sample app manifest file

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN""http://www.apple.com/
DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
 <!-- array of downloads. -->
 <key>items</key>
 <array>
  <dict>
   <!-- an array of assets to download -->
   <key>assets</key>
   <array>
    <!-- software-package: the ipa to install. -->
    <dict>
     <!-- required. the asset kind. -->
     <key>kind</key>
     <string>software-package</string>
     <!-- optional. md5 every n bytes. will restart a chunk if md5 fails. -->
     <key>md5-size</key>
     <integer>10485760</integer>
     <!-- optional. array of md5 hashes for each "md5-size" sized chunk. -->
     <key>md5s</key>
     <array>
      <string>41fa64bb7a7cae5a46bfb45821ac8bba</string>
      <string>51fa64bb7a7cae5a46bfb45821ac8bba</string>
     </array>
     <!-- required. the URL of the file to download. -->
     <key>url</key>
     <string>http://www.example.com/apps/foo.ipa</string>
    </dict>
    <!-- display-image: the icon to display during download.-->
    <dict>
     <key>kind</key>
     <string>display-image</string>
     <!-- optional. indicates if icon needs shine effect applied. -->
     <key>needs-shine</key>
     <true/>
     <key>url</key>
     <string>http://www.example.com/image.57x57.png</string>
    </dict>
    <!-- full-size-image: the large 512x512 icon used by iTunes. -->
    <dict>
     <key>kind</key>
     <string>full-size-image</string>
     <!-- optional. one md5 hash for the entire file. -->
     <key>md5</key>
     <string>61fa64bb7a7cae5a46bfb45821ac8bba</string>
     <key>needs-shine</key>
     <true/>
     <key>url</key><string>http://www.example.com/image.512x512.jpg</
string>
```

```
      </dict>
       </array><key>metadata</key>
       <dict>
         <!-- required -->
         <key>bundle-identifier</key>
         <string>com.example.fooapp</string>
         <!-- optional (software only) -->
         <key>bundle-version</key>
         <string>1.0</string>
         <!-- required. the download kind. -->
         <key>kind</key>
         <string>software</string>
         <!-- optional. displayed during download; typically company name -->
         <key>subtitle</key>
         <string>Apple</string>
         <!-- required. the title to display during the download. -->
         <key>title</key>
         <string>Example Corporate App</string>
       </dict>
     </dict>
   </array>
</dict>
</plist>
```

[1]Normal carrier data rates may apply. Messages may be sent as SMS when iMessage is unavailable; carrier messaging fees apply. [2]FaceTime calling requires a FaceTime-enabled device for the caller and recipient and a Wi-Fi connection. FaceTime over a cellular network requires iPhone 4s or later, iPad with Retina display, or iPad mini with cellular data capability. Availability over a cellular network depends on carrier policies; data charges may apply. [3]Siri may not be available in all languages or in all areas, and features may vary by area. Internet access required. Cellular data charges may apply. [4]Some features require a Wi-Fi connection. Some features are not available in all countries. Access to some services is limited to 10 devices.