2012

Intrepidus Group, Inc.

By David Schuetz
Senior Consultant

IOS MDM PROTOCOL

SIMPLE COMMAND REFERENCE

This document is a follow-up to a white paper released at Black Hat USA 2011. Essentially, it's an updated Appendix A, which lists (in greatly simplified form) the descriptions of the various commands used by Apple's iOS MDM system.

This update includes descriptions of changes implemented with iOS version 5.x, including:

- Checking out of MDM
- Installing applications
- Listing managed applications
- Removing managed applications
- Configuring settings

Please see the white paper for further details. Note also that this document is far from comprehensive, does not list all the possible device or server responses, and is mostly an attempt to publicly document otherwise undocumented Apple Private APIs. Accuracy therefore cannot be 100% guaranteed, and commands, parameters, features, etc., are subject to change in future iOS versions.

Intended for research and experimentation / testing only. Do not use this to create an actual, commercial MDM product.

Appendix A - Command Listing

(commands new to iOS 5.0 shown in ***bold/italics***)

Control Commands	• Device Lock • Erase Device • Clear Passcode
Device Queries	• Security Information • Installed Application List • Device Information • Certificate List • Profile List • Provisioning Profile List • Restrictions List • ***List Managed Applications***
Device Configuration	• Install Profile • Remove Profile • Install Provisioning Profile • Remove Provisioning Profile • ***Install Application*** • ***Remove Application*** • ***Settings***
Device to Server Commands	• Authenticate • Token Update • ***Check Out***

Overall Format

All commands are sent as Apple Property List (`.plist`) files. Each includes a top-level key called "`CommandUUID`", containing a UUID string to uniquely identify the command instance, and a top-level key "`Command`", which is a dict containing additional information.

```
<plist version="1.0">
<dict>
	<key>Command</key>
	<dict>
		<key>RequestType</key>
		<string>[command name]</string>
		[... additional parameters as needed ...]
	</dict>
	<key>CommandUUID</key>
	<string></string>
</dict>
</plist>
```

Each command is listed below with a short description, and the required parameters.

Responses

The device responds to many commands with a simple acknowledgment:

```
<plist version="1.0">
<dict>
    <key>CommandUUID</key>
    <string></string>
    <key>Status</key>
    <string>Acknowledged</string>
    <key>UDID</key>
    <string>[device UUID]</string>
</dict>
</plist>
```

The "Status" field may contain "Acknowledged", "Error", "CommandFormatError", or "NotNow" (see below for details on the error fields).

Where commands elicit a more extended response (such as for DeviceInformation queries), details of those responses are given below. Typically, these commands add a top-level field (such as InstalledApplicationList) which has as its value the extended data, stored as a string, dict, or array of other elements. For example:

```
<plist version="1.0">
<dict>
    <key>CommandUUID</key>
    <string></string>
    <key>SecurityInfo</key>
    <dict>
        <key>HardwareEncryptionCaps</key>
        <integer>3</integer>
        <key>PasscodeCompliant</key>
        <true/>
        <key>PasscodeCompliantWithProfiles</key>
        <true/>
        <key>PasscodePresent</key>
        <false/>
    </dict>
    <key>Status</key>
    <string>Acknowledged</string>
    <key>UDID</key>
    <string>[device UUID]</string>
</dict>
</plist>
```

Error Messages

The general format for an error message is the same as the acknowledgement, with "`Status`" changed to "`Error`" and an additional array of dicts added as "`ErrorChain`":

ErrorCode	(integer) A unique identifying error code
ErrorDomain	(string) Category of error
LocalizedDescription	(string) Error message, translated to a localized language
USEnglishDescription	(string) Standardized version of error message

A special error message is "`NotNow`", which is seen when a command cannot be requested because the device is locked with a passcode (such as requesting Security Information or installing a profile). When that occurs, the device will attempt to re-connect with the MDM server as soon as the device is unlocked, in order to retry the command.

A command sent with invalid or missing parameters returns the "`CommandFormatError`" status.

Device Lock

Immediately locks the device. If a passcode is present, that passcode will be required to unlock the device.

RequestType	DeviceLock

Erase Device

Immediately wipes the device memory and resets it to a "clean from factory" state. Requires connection to iTunes to restore from backup or configure as new.

RequestType	EraseDevice

Clear Passcode

If a passcode is present on the device, this command will clear that passcode. If a passcode is required by other configuration controls, the user will be given a grace period in which to set a new passcode.

RequestType	ClearPasscode
UnlockToken	(data) UnlockToken data, base-64 encoded

Security Information

Lists specified security-related settings for the device, including hardware encryption capabilities, and whether a passcode is present (and if so, whether it is compliant with configuration). If the passcode is present, the device must be unlocked for this command to execute.

RequestType	SecurityInfo
Queries	(array of strings): “HardwareEncryptionCaps”, “PasscodePresent”, “PasscodeCompliant”, “PasscodeCompliantWithProfiles”

The response is based on the general acknowledgement response, with an additional dictionary named “`SecurityInfo`”:

HardwareEncryptionCaps	integer
PasscodePresent	boolean
PasscodeCompliant	boolean
PasscodeCompliantWithProfiles	boolean

Installed Application List

Lists all the applications currently installed on the device. Includes the overall persistent storage used by the application, expressed in bytes, along with the application’s name, version, and bundle identifier. Does not list applications installed via jailbreaking methods.

RequestType	InstalledApplicationList

Additional response information, in key “`InstalledApplicationList`”, is an array of dict items:

BundleSize	integer
DynamicSize	integer
Identifier	string
Name	string
Version	string

Device Information

Retrieves specified general information about the device, including MAC addresses, IMEI, phone number, software version, model name and number, serial number.

RequestType	DeviceInformation
Queries	(array of strings): "AvailableDeviceCapacity", "BluetoothMAC", "BuildVersion", "CarrierSettingsVersion", "CurrentCarrierNetwork", "CurrentMCC", "CurrentMNC", "DataRoamingEnabled", "DeviceCapacity", "DeviceName", "ICCID", "IMEI", "IsRoaming", "Model", "ModelName", "ModemFirmwareVersion", "OSVersion", "PhoneNumber", "Product", "ProductName", "SIMCarrierNetwork", "SIMMCC", "SIMMNC", "SerialNumber", "UDID", "WiFiMAC", "UDID"

The response is a dict named "`QueryResponses`" including the above-listed items as keys. Responses that would be null (for example, the `PhoneNumber` field from an iPod Touch) are simply omitted. `AvailableDeviceCapacity` and `DeviceCapacity` are real number fields, while `DataRomingEnabled` and `IsRoaming` are boolean values. All the rest are returned as strings.

Certificate list

Lists all certificates currently installed on the device.

RequestType	CertificateList

The response includes a "`CertificateList`" array of dict values:

CommonName	string
Data	base-64 cert information
IsIdentity	boolean

Profile List

Lists configuration profiles installed on the device. Includes Common name, whether a remove passcode is required, whether removal is disallowed, unique identifiers, and other similar information.

RequestType	ProfileList

The response key "`ProfileList`" contains an array of dict items:

HasRemovalPasscode	boolean
IsEncrypted	boolean
PayloadDisplayName	string
PayloadIdentifier	string
PayloadRemovalDisallowed	boolean
PayloadUUID	string
PayloadVersion	integer
SignerCertificates	array of data items, each with base-64 cert info
PayloadContent	array of dicts, each with PayloadDisplayName, PayloadIdentifier, PayloadType, and PayloadVersion keys.

Provisioning Profile List

Lists provisioning profiles installed on the device (similar to the Profile list).

RequestType	ProvisioningProfileList

The response includes a "`ProvisioningProfileList`" key, which contains an array of dict values:

ExpiryDate	date
Name	string
UUID	string

Restrictions List

Lists restrictions currently in effect on the device. For example, lists disabled applications, whether backup encryption is forced on, etc.

RequestType	RestrictionsList

The response includes "`GlobalRestrictions`", which is a dict containing detailed list of restrictions, most presented as boolean values. The exact content and structure depends on the restrictions in place on the device.

List Managed Applications

Lists all the applications currently installed on the device *which are managed by the MDM server*. These are applications which were installed by MDM.

RequestType	ManagedApplicationList

Additional response information, in key "`ManagedApplicationList`", is an array of dict items, one per application (each dict has the applications' bundle ID as its key):

Status	"Managed"
ManagementFlags	integer

An example response (in JSON format):

```
{'Status': 'Acknowledged',
 'CommandUUID': 'd3a8ac67-6662-4f43-8f28-27b1ce1ab7d5',
 'UDID': '-- redacted --',
 'ManagedApplicationList':
      {'com.apple.movietrailers':
            {'Status': 'Managed',
            'ManagementFlags': 1}
      }
}
```

The ManagementFlags setting controls what happens if the device is removed from MDM control: If the least-significant bit is set (1), then the application and its data will be deleted if the device is removed from MDM control. If that bit is not set, then the application will not be deleted.

Install Profile

Given a base-64 encoding of a `.mobileconfig` profile (as created by the IPCU or other tools), installs the profile on the device.

RequestType	InstallProfile
Payload	(data) IPCU .mobileconfig file, base-64 encoded

Remove Profile

Given a payload identifier (which is typically shown as a reverse-DNS identifier such as "`com.example.cfg.restrictions`"), removes the profile from the device.

RequestType	RemoveProfile
Identifier	(string) Profile identifier

Install Provisioning Profile

Given a base-64 encoding of a `.mobileprovision` profile (as created by the IPCU or other tools), installs the profile on the device.

RequestType	InstallProvisioningProfile
Payload	(data) IPCU .mobileprovision file, base-64 encoded

Remove Provisioning Profile

This command removes the provisioning profile from the device, given the profile's UUID.

RequestType	RemoveProvisioningProfile
UUID	(string) Provisioning profile UUID

Install Applications

Two different forms of this command are supported: One for installing an application from the App Store, the other for installing a custom-built application.

The first form takes an iTunesStoreID as an argument, and causes the device to prompt the user for their AppleID and Password. The ID is the same as is seen in a web-based App Store page (for example, "http://itunes.apple.com/us/app/apple-store/id471966214?mt=8"), where 471966214 would cause the iTunes Movie Trailers app to be installed.

RequestType	InstallApplication
ManagementFlags	integer (see List Managed Applications)
iTunesStoreID	integer

The other form installs a custom-developed app. You may need to install a related provisioning profile first. The ManifestURL key points to a Manifest.plist file (detailed below).

RequestType	InstallApplication
ManagementFlags	integer (see List Managed Applications)
ManifestURL	url

The Manifest.plist file provides information about the application, as well as a link to an Xcode .ipa file to download the app.

```
<plist version="1.0">
  <dict>
    <key>items</key>
    <array>
      <dict>
        <key>assets</key>
        <array>
          <dict>
            <key>kind</key>
            <string>software-package</string>
            <key>url</key>
            <string>https://*** SERVER_IP ***:port/MyApp</string>
          </dict>
        </array>
        <key>metadata</key>
        <dict>
          <key>bundle-identifier</key>
          <string>*** BUNDLE ID (com.example.myapp) ***</string>
          <key>bundle-version</key>
          <string>1.0.0</string>
          <key>kind</key>
          <string>software</string>
          <key>subtitle</key>
          <string></string>
          <key>title</key>
          <string>*** APP NAME ***</string>
        </dict>
      </dict>
    </array>
  </dict>
</plist>
```

Remove Application

Given an application bundle ID, removes the application and its data from the device. Returns an error (and doesn't delete the app) if the application requested for removal is not managed by MDM.

RequestType	RemoveApplication
Identifier	(string) Bundle identifier

```
<plist version="1.0">
  <dict>
    <key>items</key>
    <array>
      <dict>
        <key>assets</key>
        <array>
          <dict>
            <key>kind</key>
            <string>software-package</string>
            <key>url</key>
            <string>https://*** SERVER_IP ***:port/MyApp</string>
          </dict>
        </array>
        <key>metadata</key>
        <dict>
          <key>bundle-identifier</key>
          <string>*** BUNDLE ID (com.example.myapp) ***</string>
          <key>bundle-version</key>
          <string>1.0.0</string>
          <key>kind</key>
          <string>software</string>
          <key>subtitle</key>
          <string></string>
          <key>title</key>
          <string>*** APP NAME ***</string>
        </dict>
      </dict>
    </array>
  </dict>
</plist>
```

Settings

Used to configure certain settings on the device. Currently supports changing DataRoaming and VoiceRoaming. Hints in the code indicate that some form of Wallpaper control may be present as well.

RequestType	Settings
Settings	(array of dicts)

The "Settings" key is an array of dicts, each of which represents a setting to be changed:

Item	(string) (“DataRoaming” or “VoiceRoaming”)
Enabled	(boolean)

Example of the full command:

```
{'CommandUUID': 'ce0c8b34-9ac5-44f6-a25b-1c9cfffce666',
 'Command': {
      'RequestType': 'Settings',
      'Settings': [
            {'Item': 'DataRoaming', 'Enabled': False},
            {'Item': 'VoiceRoaming', 'Enabled': True}]}}
```

When “VoiceRoaming” is changed on a device without voice services (such as an iPad), this returns an error for that item alone, but other items may still be successfully processed. Example of such a response:

```
{'Status': 'Acknowledged',
 'CommandUUID': 'ce0c8b34-9ac5-44f6-a25b-1c9cfffce666',
 'UDID': '-- redacted --',
 'Settings': [
      {'Status': 'Acknowledged', 'Item': 'DataRoaming'},
      {'Status': 'CommandFormatError', 'Item': 'VoiceRoaming'}]}
```

Authenticate

This is a client command, sent by the client to initiate enrollment. Can be used by the server to permit or deny enrollment based on the device’s UDID. NOTE - Does not follow same format as server-to-client commands. Has no `CommandUUID` field nor the `Command` dict structure -- all parameters are top-level items in the main property list dict.

MessageType	Authenticate
Topic	(string) Subject Name: User ID on APNS push certificate used by server
UDID	(string) Device UDID

Token Update

This is a client message, sent by the client during enrollment. Provides the server with tokens used to contact device via APNS, as well as a key to unlock the device through the Clear Passcode command. NOTE - Does not follow same format as server-to-client commands. Has no `CommandUUID` field nor the `Command` dict structure -- all parameters are top-level items in the main property list dict.

MessageType	Token Update
PushMagic	(string) UUID-like string
Token	(data) 32-byte APNS device token, base-64 encoded
Topic	(string) Subject Name: User ID on APNS push certificate used by server
UDID	(string) Device UDID
UnlockToken	(data) Device unlock key, base-64 encoded

Check Out

This is a client command, sent by the client to alert the server that the device is about to remove itself from MDM control. The device does not wait for a response, and the server cannot refuse the removal. NOTE - Does not follow same format as server-to-client commands. Has no `CommandUUID` field nor the `Command` dict structure -- all parameters are top-level items in the main property list dict.

MessageType	CheckOut
Topic	(string) Subject Name: User ID on APNS push certificate used by server
UDID	(string) Device UDID