



Performing Device Management With Funambol

Funambol provides a Device Management (DM) server and client API for building DM applications. The DM server is based on the Open Mobile Alliance (OMA) DM specification. This document offers an overview of DM, presents background concepts and provides a technical introduction to the Funambol DM server and client API.

Overview

Configuring a mobile device can be a difficult task due to the many variables involved. Network parameters change from phone to phone, visual interfaces can be difficult to use and mobile operators have different configurations. Moreover, mobile devices are becoming smarter every day with increased functionality and applications that are pre-installed or downloaded and installed by the user, making the task of configuring a mobile device more complex. Mobile operators, service providers, device manufacturers and others are looking for a platform that allows remote management of devices in a convenient and effective way.

To address this need, some companies are starting to offer mobile device management solutions. However, available mobile DM solutions are based on different proprietary implementations. A proprietary platform exposes users to a serious risk of insufficient interoperability and associated cost implications. In the long run, it means being locked into a vendor and technology, while the market is moving towards standardization.

Device management from Funambol is based on open standards, enabling an external party to manage, set parameters, troubleshoot, service problems and install or upgrade software remotely. In broad terms, device management consists of three areas:

- Management operations: the protocol used between a management server and mobile device
- Data model: the data made available for remote manipulation, for example browser and mail settings
- Policy: the policy specifying who can update a particular parameter or object in the device

OMA DM Protocol

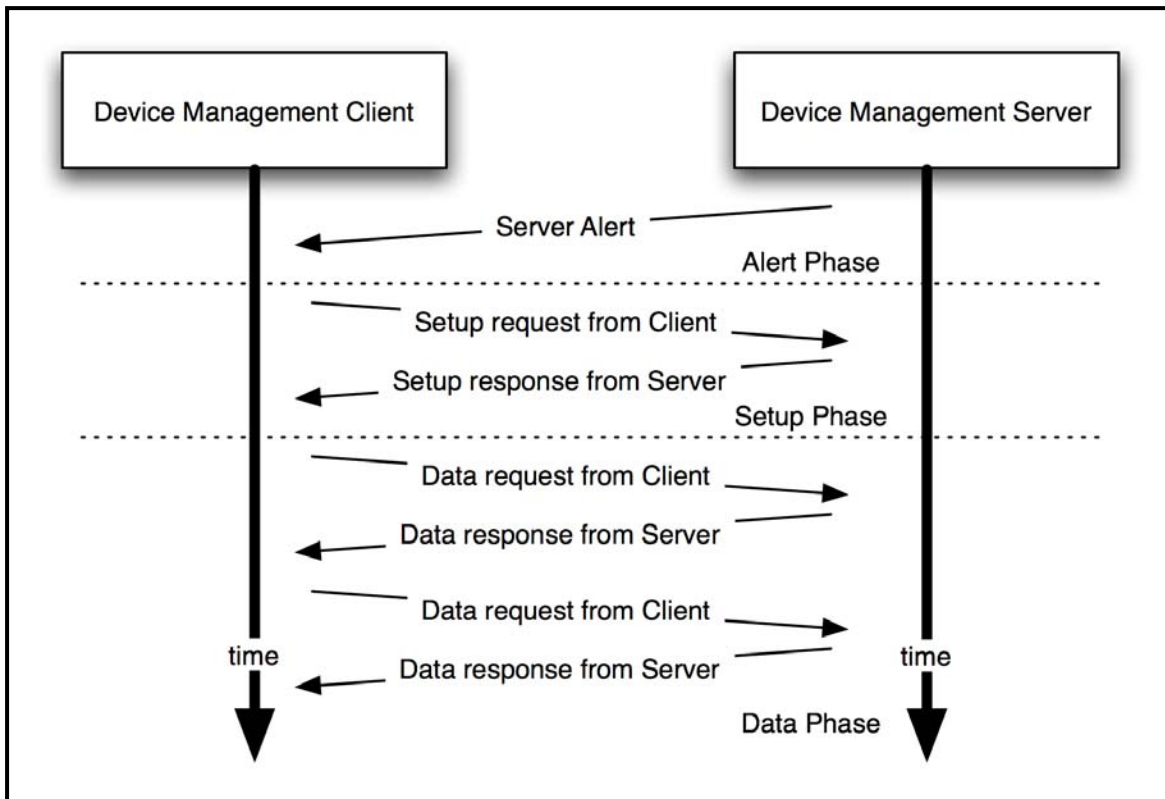
The OMA DM protocol is an open industry standard for remote DM of networked devices. DM is the generic term for a technology that allows third parties to carry out the difficult process of configuring mobile devices on behalf of the end user. DM users would typically be mobile operators, service providers and device manufacturers.

In a wireless environment, the crucial element for a device management protocol is the need to efficiently and effectively address the characteristics of mobile devices, including low bandwidth and high latency. The OMA DM protocol has been built with these requirements. The scope of OMA DM includes:

- device configuration – i.e. modify or read operating parameters
- software maintenance
- inventory – i.e. read from a device its current operating parameters, list installed or running software, determine hardware configurations
- diagnostics – i.e. listen for alerts sent from a device, invoke local diagnostics on a device

The OMA DM Protocol uses a messaging sequence that consists of three parts (see figure below):

- Alert Phase – used only for server initiated management sessions
- Setup Phase – authentication and device information exchange
- Data (Management) Phase



Alert Phase: This phase is optional and data flows only from server to client. The server sends a notification package to the client requesting it to start a new management session. This is usually done with a WAP push message sent by a Push Proxy Gateway on behalf of the DM server which acts as the WAP Push initiator agent.

Setup Phase: This phase is required. The client issues a setup request, with data flowing from client to server, followed by a server response. The initial client request contains 3 primary pieces of information:

1. Device information e.g. device id, manufacturer, model tag, phone language and DM protocol version
2. Client credentials used for authentication purposes
3. Session alert specifies whether the incoming session is client or server initiated

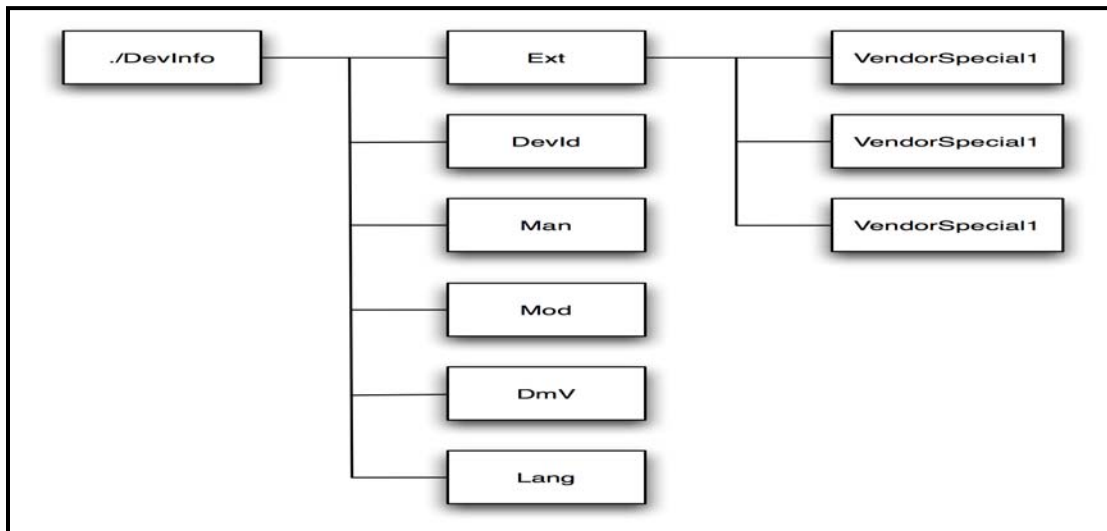
The server responds with its credentials, to identify the server to the client for authentication and identification purposes. The server also sends user interaction and the first data command.

Data (management) phase: These message exchanges comprise the session's core. In this phase, the device's 'Data request from client' consists of status information and results that it provides in response to commands from the server. The first 'Data request from Client' is the response to commands included in the 'Setup response from Server'. Afterward, the server can issue new commands in another 'Data request from server' or it can simply indicate there are no more operations. In that case, the client will stop sending a 'Data request from client' and closes the session.

Device Management Tree

Device configuration data is organized in a hierarchical structure called the device management tree. Subtrees are called device management nodes and a leaf, usually a single configuration parameter, is called a manageable object. Device objects can be anything from a single parameter to a splash screen GIF to an entire application. The DM tree is essentially mapped to permanent or dynamic objects as an addressing schema to manipulate these objects. Permanent objects are objects that are built into the device when it was manufactured and typically cannot be deleted. Dynamic objects are items that can be added or deleted (e.g. ringtones or wallpaper).

The initial request from the client always contains device information whereby the data is retrieved from the `./DevInfo` subtree. The `./DevInfo` node is only part of the overall DM tree structure and it maps to device parameters that allow initial operations and inspection of the device by support personnel. The `./DevInfo` object is a standard object such that any compliant device exposes the same basic information in the same structure.



The figure provides a good example of how management objects are organized. `./DevInfo` is the main node, which includes subtrees (other management nodes) and leafs (manageable objects). `Ext`, for example, is a subtree where any vendor implementation can expose vendor specific information. The other entities represent manageable objects: `DevId` contains the device id (such as the IMEI code for a phone device); `Man` is the device manufacturer identifier; `Mod` represents the device model identifier; `DmV` specifies the OMA DM client version; `Lang` is the device's current language setting.

In addition to the manageable object value, each manageable object has properties. These define metadata information about an object to allow things such as access control, etc. These properties are:

- Access Control List (ACL): Defines who can manipulate the underlying object (required)
- Format: Specifies how an object should be interpreted. For example, if the underlying object is a URL for a particular management server, the Format may be defined as `chr` or character (required)
- Name: Name of the object in the tree (required)
- Size: Size of the object in bytes (required for Leaf Objects, not applicable for Interior Nodes)
- Title: User-friendly name of the object (optional)
- Tstamp: Time stamp of the last modification (optional)
- Type: MIME type of the object (required for Leaf Objects, optional for Interior Nodes)
- VerNo: Version number of the object (optional)

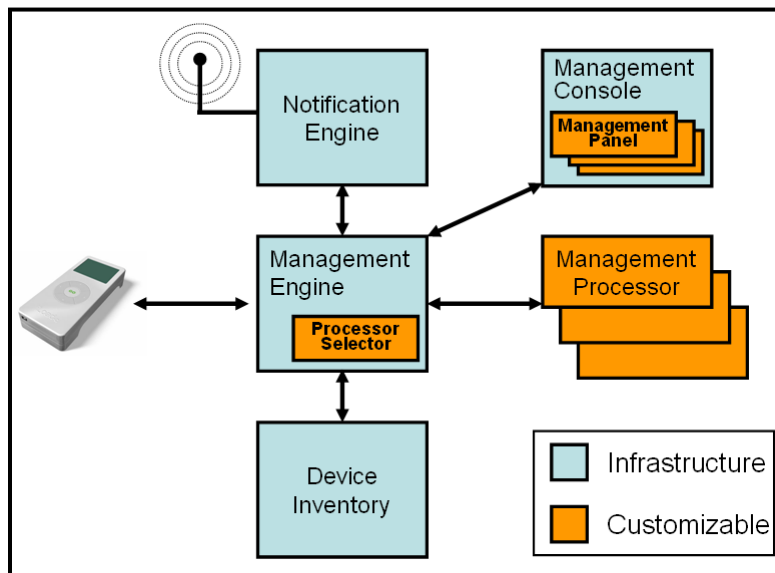
Management objects can be manipulated via SyncML messages with these commands:

- Add: Add an Object (Node) to a tree
- Get: Returns a Node name based on the URI passed with the GET request
- Replace: Replaces an Object on the tree
- Delete: Deletes an Object on the tree
- Copy: Copies an Object on the tree
- Exec: Execute a device-defined command on an Object on the tree

Funambol DM Server

Funambol DM includes two components, the Funambol DM Server and the Funambol DM Client API, which are described below.

The Funambol DM Server implements OMA DM. It provides a framework that implements DM functionality and that supports extensions. In the diagram below, the orange blocks represent functionality that developers can plug into the framework.



At the core of the DM Server is a management engine that is responsible for understanding and interpreting the OMA DM protocol. The management engine delegates the real DM logic to an external and customizable Management Processor component. Multiple management processor components can be used and a management processor is selected via the Processor Selector that is invoked at runtime from the management console.

A Notification Engine notifies mobile devices to start a new DM session. For example, in the case of WAP Push enabled devices, it builds a binary WAP push message which is delivered to the device as specified by OMA specifications. The Notification Engine supports both bootstrap and DM notification messages. The former are used to store in the device initial connection and authentication data; the latter is used to trigger a server initiated OMA DM session.

The repository of Device Descriptor Framework (DDF) descriptors is in the Device Inventory. The DDF framework is part of the OMA DM specification and is a dynamic discovery mechanism that a server can use to discover and gather information about the device's management tree objects.

The Management Console is used by staff to interact with the DM server and with the devices.



The Management Processor contains code that implements DM logic. When the server receives an OMA DM message, a customizable Processor Selector chooses the Management Processor best suited for the request. Policy examples are selecting a Management Processor based on:

- device details e.g. device id, manufacturer, model
- state of the current DM session
- external configuration/interaction/events

Management Processors can be simple or complex. To simplify the work of DM developers, an easy to use Management Processor based on a scripting language is provided. With it, writing a Management Processor is as simple as writing a BeanShell (<http://www.beanshell.org>) script.

Each Management Processor can have its own Management Panel. The panel implements the user interface through which staff interact with the device, using a particular Management Processor object.

Funambol DM Client API

Client APIs support the development of OMA DM applications that can be remotely managed by an OMA DM server. The API includes:

- Database Layer: Represents the device's local database.
- SyncPlatform Store (SPS) Layer: Abstracts database access. It can be used by the framework to access the local database in a generic way. SPS is optional in a Funambol application.
- SyncPlatform Data Synchronization (SPDS) Layer: Abstracts the complexity associated with the OMA DS protocol from extension developers, who simply need to call a few methods of the synchronization manager to kick off the synchronization process.
- SyncPlatform Device Management (SPDM) Layer: Core layer of the OMA DM implementation. The main component is called device manager, which is responsible for storing and reading the management trees and nodes and for exposing to the application code hooks to start and handle the DM session. All protocol details and complexity are abstracted for the mobile application developer.

Summary

The Funambol DM Server is an OMA DM-based framework that provides extensive infrastructure for building DM applications. It enables developers to add the specific DM functionality that they need for their environment, without having to reinvent the DM wheel. For more information, you can:

- visit the OMA DM website at http://www.openmobilealliance.org/tech/wg_committees/dm.html
- download the Funambol DM Server software and documentation from the Funambol website
- contact your Funambol sales representative

About Funambol, Inc.

Funambol is the leading provider of mobile open source cloud sync and push email software. Funambol open source software has been downloaded over three million times by 50,000 developers in more than 200 countries. The commercial version of Funambol has been deployed at mobile operators, portals, device manufacturers, service providers and ISVs including customers such as AOL, 1&1, EarthLink and CA, Inc. Funambol is headquartered in Redwood City, California with an R&D center in Italy. For more information, please visit www.funambol.com.