

# Universal Manager: Seamless Management of Enterprise Mobile and Non-mobile Devices

Sandeep Adwankar, Sangita Mohan‡, Venu Vasudevan

Mobile Services and Platforms Department,  
Motorola Labs,

1301, E. Algonquin Road,  
ILO2-2240, Schaumburg, IL 60196

[Sandeep.adwankar@motorola.com](mailto:Sandeep.adwankar@motorola.com), [venuv@labs.mot.com](mailto:venuv@labs.mot.com)

‡University of Illinois, Chicago, [smohan1@uic.edu](mailto:smohan1@uic.edu)

## Abstract

Seamless management of resource constrained wireless devices along with enterprise hosts is complex because of the multitude of management protocols and architectures. An atomic and coordinated management action that spans the scale of the enterprise is needed. In this paper we present the architecture and our experiences in building a Universal Manager that manages all mobile and non-mobile devices in the enterprise. We came up with a multi-protocol gateway that blends intermittently connected mobile devices seamlessly with enterprise hosts. Our implementation of SyncML based mobile devices integrated with SNMP based enterprise manager is deployed in the enterprise and performance results are presented.

## 1 Introduction

Mobile handheld devices such as cellphones and PDA's are becoming an increasingly important part of the ecosystem. Although enterprise ownership of these devices grew rapidly through the 1990's, they were largely viewed separately from the enterprise infrastructure. Cellphones were managed and serviced by telco operators, and therefore of no concern to enterprise IT staff. PDA's, while tracked by enterprises, were essentially either self-managed and non-networked, or managed in limited fashion by the ISP

who provided connectivity services. As mobile devices were viewed as computing devices with fixed "burned in" functionality, there wasn't a strong motivation for sophisticated enterprise-driven management of mobile devices.

Increased mobile device complexity, raised user expectations of device functionality, and growth in the richness of "service bundles" supported by mobile devices motivate a more sophisticated approach to enterprise-driven mobile device management. As mobile devices integrate PDA and phone functionality, and add capabilities such as GPS and cameras, it becomes important to incrementally upgrade the core software and application palettes of a mobile, much like IT staff upgrade desktop platforms. Enterprises are using the computing capabilities of mobile devices to support applications customized to the enterprise or business unit, and perhaps customized to the backend enterprise information that the mobile provides access to. As converged (voice and data) mobile devices begin to be used for fleet and workgroup oriented functions, uniformity of software versions across mobiles in a fleet becomes vital to uniform access and exchange of data amongst enterprise users.

While mobile devices have gotten computationally capable, they are nevertheless considerably different from desktop computers. They are based on a variety of wireless communications systems [1][2] such as 3G, CDMA, GSM, paging protocols that differ widely. They have different underlying operating systems, application development environments and hardware capabilities. These mobile devices have low CPU power, memory; storage capabilities compared to other enterprise

devices/hosts. As part of the enterprise network, these devices however will be at parity with other enterprise hosts in terms of need for reliable operation, need for zero downtime, rapid fault detection with correction and performance that satisfies Service Level Agreements (SLA). With mobility, these devices pose more security threats than non-mobile hosts. Enterprises need assurance of security of corporate data along with security of business processes.

With evolving technologies of seamless vertical handover [3] between Wide Area Network (WAN), Wireless Local Area Network (WLAN) and Personal Area Network (PAN), the boundary between mobile and non-mobile devices is slowly blurring. With convergence of IP voice and data with WLAN, PAN and Cellular, a user has seamless mobility [4] across work, home, car and hot spots. Thus a cellular mobile device can be part of the operator controlled cellular radio network, the enterprise controlled WLAN network or a user controlled bluetooth based PAN network. These technologies pose a number of new requirements for Operations, Administration, Maintenance and Provisioning (OMAP) on management system. A unified management system should provide consolidated Authorization, Authentication and Accounting (AAA) functions and consolidated OMAP functions across heterogeneous networks and devices.

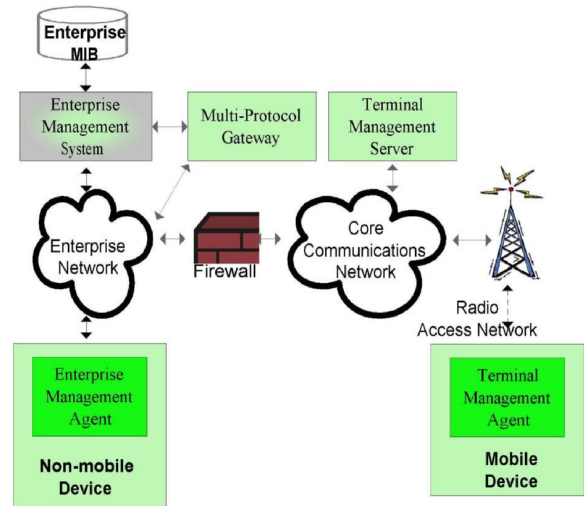
An “SNMP [5] everywhere” approach might seem a technically feasible way of unifying the fixed and mobile enterprise. However, SNMP has not gained traction in the mobile enterprise for a variety of reasons such as complexity, size, underlying protocols that need to be supported. Instead, a number of wireless management approaches have developed that operates effectively in the thin-client, low-bandwidth (high latency), intermittently connected world of mobile devices. Having multiple management paradigms for different enterprise devices is technically feasible but deficient and expensive. Management stations are expensive and require a steep learning curve on the part of IT staff. Consequently, having multiple management platforms requires a compelling rationale. Even if the cost and training issues were addressed, IT administrators will only get partial, fractured views of the enterprise that they will then have to manually or visually integrate. It is hard to detect, for instance, that the desktop and mobile versions of document editors are compatible for all employees in a business unit. Supporting such an operation requires an integrated view of management data on a single management platform.

The solution being proposed in this paper is to use a conventional enterprise management protocol such as

SNMP as the unified management platform, while at the same time supporting a diversity of management protocols. This allows a unified view of the enterprise without expecting enterprise management agents to be supported on wireless devices. The solution proposed uses a multi-protocol gateway that translates between the enterprise management operations dispatched from a management station, and the specific wireless management protocol that enacts it.

The rest of the paper is organized as follows. Section 2 introduces the architecture of the Universal Manager for seamless management of all enterprise devices. Section 3 takes a look at diversity of management protocols and outlines some widely used protocols and frameworks. Section 4 describes all elements of the Universal Manager in detail. Section 5 presents seamless management flow for managing SyncML based mobile devices. Section 6 presents performance results and Section 7 presents conclusions of the deployed system.

## 2 Architecture



**Figure 1 The Universal Manager Architecture**

As shown in Figure 1, the universal manager spans the network controlled by the enterprise and a core communications network such as the Internet to manage devices connected to each of these networks. A user controlled PAN in this case is considered part of the enterprise network. Enterprises maintain Management Information Base (MIB) [6] that consists of collection of managed objects for each type of enterprise device. The

enterprise management system (EMS) provides a complete view of the entire enterprise including all devices, hosts and routers in the enterprise.

Enterprise devices (supporting the enterprise management protocol) are located through discovery mechanisms [7] and the enterprise management agent directly executes the requested management action. For mobile devices, legacy discovery mechanisms will not work. In our architecture, mobile devices advertise their presence with their capabilities data to the multi-protocol gateway. The multi-protocol gateway acts as a proxy and represents the mobile device in the enterprise network. Thus an enterprise management action on a mobile device will be routed through multi-protocol gateway.

A mobile device has a terminal management (TM) agent that is customized for its capabilities. The terminal management agent has access to firmware specific details and can perform management operations such as getting or setting configuration data, software image installation, and fault and security management. The TM agent talks with a TM server using a protocol that the device can support. For example a mobile device that supports only Short Message Service (SMS), management actions and results/responses are sent over SMS between TM agent and TM server.

The multi-protocol gateway maps the destination of a management action to a particular mobile device. For example the multi-protocol gateway has advertised a GSM terminal with International Mobile Equipment Identity (IMEI) "000000011234564" as IP address 10.1.100.212 in the enterprise network. For management action of mass software upgrade for all devices in 10.1.100 subnet, the management action destined for 10.1.100.212 is routed to the multi-protocol gateway. The multi-protocol gateway then maps it to the corresponding IMEI and from the capabilities data of the device, it converts management action Protocol Data Unit (PDU) to terminal management protocol specific PDU. The multi-protocol gateway then schedules the management action on the corresponding terminal management server. The terminal management server actually participates in management protocol exchanges with terminal management agent to perform management actions such as software upgrade. Enterprises need atomicity of management actions as expressed in Table 1. The multi-protocol gateway converts enterprise atomicity requirements to individual device protocol levels.

**Table 1 Atomicity Requirements**

Entity	Atomic Operation
--------	------------------

Device	Device software upgrade with change in corresponding configuration.
Group	Changes in group protocol such as multicast.
Enterprise	Business process change.

The following characteristics make this architecture truly scalable:

- Support for diverse terminal management protocols
- Distributed terminal management servers based on protocols supported
- Scheduling management actions on terminal management server based on availability of resources
- Terminal management agent specific to a device based on hardware, software capabilities
- Using layer 2 routing to advertise presence of intermittently connected devices

### 3 Survey of Management Protocols

A management protocol consists of components such as data model, data modeling language, data representation, protocol operations and protocol transports. The degree of fitness of each of these components differs for different devices and networks. This resulted in diverse set of management protocols, each suitable for a certain set of platforms. Following is a brief introduction to these management protocols.

#### 3.1 Simple Network Management Protocol (SNMP)

SNMP is the IETF standard for operations and maintenance protocol for the Internet. SNMP is based on the manager/agent model consisting of a manager, an agent, a database of management information, managed objects and the network protocol. The manager and agent use MIB and a relatively small set of commands or protocol operations (e.g. GET, SET) to exchange information through SNMP PDUs. The MIB is organized as a tree structure with individual variables, such as point status or description, being represented as leaves on the branches. The SNMP data model defines an object identifier (OID) that is used to distinguish each variable uniquely in the MIB and in SNMP messages. For example, OID 1.3.6.1.2.1.1.1.0 is associated with MIB variable system description. The SNMP data modeling language is a subset of Abstract Syntax Notation Number One (ASN.1), defined as SMIV2. SNMP uses ASN.1 BER encoding as the data representation. SNMP is

commonly implemented over Transport Control Protocol (TCP) or User Datagram Protocol (UDP) transport layers.

### 3.2 Web-Based Enterprise Management (WBEM)

WBEM is a set of Distributed Management Task Force (DMTF) standards [8] using data model called Common Information Model (CIM) and a specification for CIM operations over HTTP. CIM provides a comprehensive object-oriented information model in the form of a hierarchy of classes and associations. Microsoft System Management Server is WBEM based system for managing enterprise's inventory of windows based hardware, software, distribution and installation of software updates, diagnostics and control of remote clients and applications.

### 3.3 XML Based Protocols

**BEEP** [9] is an application protocol framework that can be used to construct a variety of application protocols. It uses XML and MIME for its own protocol elements, and can transfer arbitrary MIME content.

**SOAP** [10] is an application messaging framework that can be used to construct a variety of XML applications. It uses XML for its own protocol elements, and can transfer arbitrary XML content. The SOAP has an envelope structure for conveying arbitrary XML data, data encoding and protocol (e.g. HTTP) binding framework.

**JXTA** [11] is a network overlay framework for peer-to-peer applications. It defines a set of protocols for Peer discovery, information, binding, resolving, endpoint routing and membership.

**NetConf** [12] is a proposed protocol for network device management. An example of such a protocol is JUNOScript [13], which is an RPC (Remote Procedure Call) over SSH, SSL, or telnet and uses structured data represented in XML.

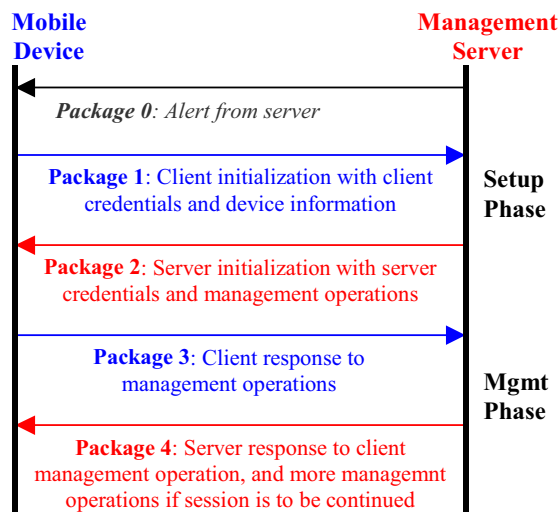
### 3.4 Wireless Application Protocol Provisioning

Wireless Application Protocol (WAP) Provisioning defines the architecture [14] and a protocol to configure mobile devices using WAP by either over the air (OTA) provisioning or provisioning by means of smart cards. It is a two-step process consisting of bootstrapping to create a trusted relationship between the device and the infrastructure. The bootstrapped information is then used to access remote access servers, WAP proxies or application servers for configuration information.

### 3.5 SyncML Protocol

The SyncML standard [15] [16] was developed as an XML-based information synchronization standard designed specifically for the needs of the wireless industry. Thus, the SyncML protocol is *lightweight* to cater to device limitations, *language-neutral* and *protocol-neutral*. To suit device limitations, SyncML language constructs are kept fairly minimal, with the protocol not using some features (e.g., server sockets) that are yet to become pervasive on mobile devices. Because SyncML is XML-based, it is inherently language-neutral, and supports OEM-specific extensibility. In addition, the SyncML "protocol" can run over a number of underlying transport protocols including HTTP, WSP, and OBEX.

SyncML's popularity in information synchronization led to its scope being expanded via SyncML-DM to include device management. SyncML-DM allows management actions to be performed on management objects, where a management object might represent a device configuration or the run-time software application environment. Actions taken against the former might include reading and setting parameter keys and values, while actions taken against the latter might include installing, upgrading, or uninstalling software elements. The signatures of the Get and Set methods on a management object are type-specific and may vary substantially in complexity. For instance, a management action for setting the device clock accepts a simple textual MIME type (text/plain) while an action to change the WAP browser settings requires new WAP provisioning "blobs" to be transmitted as part of the Set operation. Software upgrades present another example of a management action with significant payload complexity.



**Figure 2 SyncML Protocol Phases**

As shown in Figure 2 the SyncML-DM is a 2-phase protocol consisting of a *setup* phase for authentication and device information exchange, followed by a *management* phase that can be repeated multiple times to support complex manager-to-mobile sessions. A management session may also start with Packet 0 (the trigger), where the trigger may be out-of-band depending on the environment.

Table 2 details comparison of some of these widely used management protocols.

**Table 2 Comparison of commonly used management protocols**

Protocol	Simple Network Management Protocol (SNMP)	SyncML Device Management (SyncML DM)	Web-based Enterprise Management (WBEM)
Data Model	Management Information Base	Management Tree	CIM Schemas
Data Modeling Language	Structure of Management Information (subset of ASN.1)	Data Description Framework	Managed Object Format
Data	ASN.1	XML,	xmlCIM

Representation	BER encoding	WBXML	Encoding
Identification	Object Identifier	URI	Key
Protocol Commands	Get, Getnext, Getbulk, Set, Trap, Inform	Get, Replace, Add, Delete, Exec, Copy, Alert	GetClass, GetInstance, CreateClass, CreateInstance, GetProperty, DeleteClass, DeleteInstance
Protocol Transport	UDP	HTTP, OBEX, WSP	HTTP
Standards Organization	IETF	OMA	DMTF
Use	Routers, End hosts	Mobile devices	End hosts

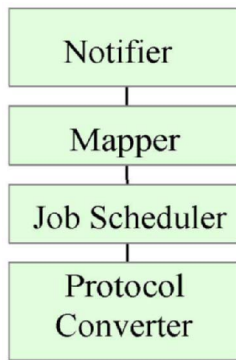
## 4 The Universal Manager

The management protocol survey in the previous section showed that management protocols differ in a number of attributes. We envision that the multi-protocol gateway is an important architectural element in seamless management of these diverse devices.

### 4.1 Multi-Protocol Gateway

The multi-protocol gateway is a software entity that represents a terminal (having its own terminal specific management protocol) in the enterprise management system. The multi-protocol gateway makes the enterprise management system believe that the terminal is like any other manageable entity in the enterprise. At the same time it uses terminal specific management protocols for management of terminals.

Following are the main components of the Multi-protocol gateway as shown in Figure 3:



**Figure 3 Multi-Protocol Gateway**

**Protocol Converter:** This component is responsible for the conversion of the enterprise protocol to the terminal management protocol. This includes mapping atomicity requirements on to the terminal specific protocol.

**Mapper:** The Multi-protocol gateway maintains a list of the wireless devices that can be reached. This list is populated by a handshake or a discovery mechanism that is exchanged between the Multi-Protocol Gateway and the device. The Mapper maps the address of the device on the enterprise address space. For example, in the case of a GSM terminal, it allows the mapping of a terminal IMEI to the enterprise local private IP address space. The Mapper advertises reach-ability to this local private IP address space from the enterprise address space. It then provides translation from the local private IP address space to the terminal IMEI.

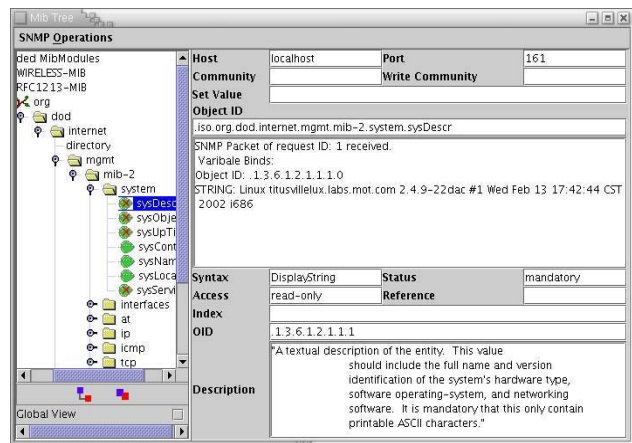
**Job Scheduler:** This component is responsible for the load balancing of terminal management activities. It schedules TM operations depending on the terminal management server availability and availability of scarce wireless resources.

**Notifier:** Since the time taken for terminal management is more than that expected in the enterprise management, the Notifier allows notifying the enterprise management system after the terminal management task is complete.

## 4.2 Enterprise Management System

Since SNMP is commonly used as the enterprise management protocol, we have used an SNMP-based enterprise management system that manages IP based devices, applications and software systems in the enterprise. Most hosts and routers in our enterprise support SNMP, thus the enterprise management agent in

our case is simply an SNMP agent. Enterprise manager is the SNMP manager that uses SNMP PDUs to get the local system's information directly from the SNMP agent on the host. The Enterprise Management Server is used to query parameters from both enterprise devices and wireless devices with the help of multiple MIBS specific to both wired and wireless devices. We wrote a new GUI using Java SNMP API [17] as shown in Figure 4. The figure shows the result of SNMP GET operation on Object ID iso.org.dod.internet.mgmt.mib-2.system.sysDescr. This management operation fetches the local system's information directly from the SNMP agent on the host.



**Figure 4 Management of non-mobile devices using the Universal Manager**

## 4.3 Terminal Management System

Most wireless terminals do not have enough device and network capability to house a SNMP agent. The terminal management standard such as SyncML based device management is specifically addressed for resource-constrained terminals. We built the SyncML based terminal management system for managing resource-constrained wireless mobile devices. Thus the multi-protocol gateway transforms SNMP PDU's to SyncML packages. In this transformation process, SNMP security credentials are converted to SyncML based security credentials [20] required by terminal management server.

### 4.3.1 Terminal Management Agent

The terminal management agent is based on the SYMPLE [19] platform that extends SyncML device management specification. The SYMPLE platform brings

mechanism of script construction and delegation to SyncML-based wireless terminal management. It defines a SyncML-derived scripting language called *Symple* (SYncML Programming Language) that extends SyncML semantics in a lightweight manner suitable for resource-constrained devices. The Symple architecture also extends the SyncML runtime framework with support for the scheduling, evaluation and lifecycle management of Symple scripts. A Synclet is an executable script consisting of Symple commands. A Synclet specification comprises of two parts: a policy and an action routine. The Synclet policy specifies non-functional aspects of the Synclet such as if, when and how often it should be executed. The action routine is the management script that makes up the Synclet and is composed of a set of instructions that allow periodic, coordinated and policy based management action. The policy, action routine separation allows the same functional Synclet to be reused in different circumstances.

The terminal management agent occupies only 110 KB space on a terminal. Shown in Figure 5 is a GSM terminal used for the implementation. We chose J2ME [18] as the platform-independent application development environment, as it provides User Interface and HTTP connection APIs. We chose Tiny XML [21] as the XML parser as it is around 10 K in size. It is also SAX-based [22] and hence, memory efficient.

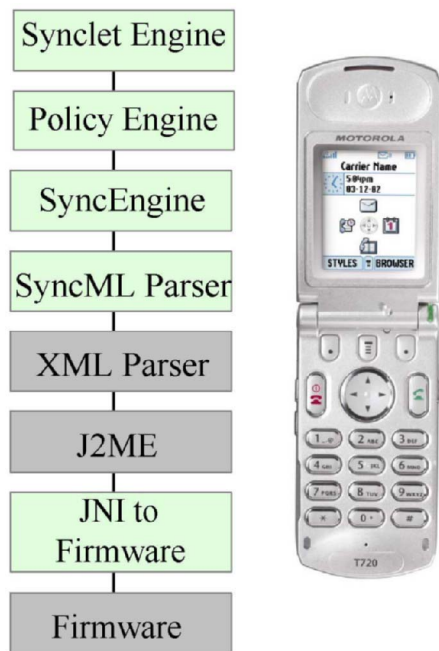


Figure 5 Terminal Management agent

The SyncML parser was written on top of the XML parser. The SyncML parser parses the SyncML package received by the J2ME Midlet over HTTP. The result of parsing is a set of management actions that need to be performed on the device. Since we had access to the firmware code of a GSM terminal, we wrote closed classes that in turn made Java Native method calls to firmware. The terminal management agent has two types of mechanisms for starting a TM Midlet. In one mechanism the TM agent has a socket connection in receive mode. On receiving a UDP message, it will check the body of the UDP message and will launch the TM Midlet. For phones, where this mechanism was not possible, we used SMS mechanism to trigger the TM Midlet.

### 4.3.2 Terminal Management Server

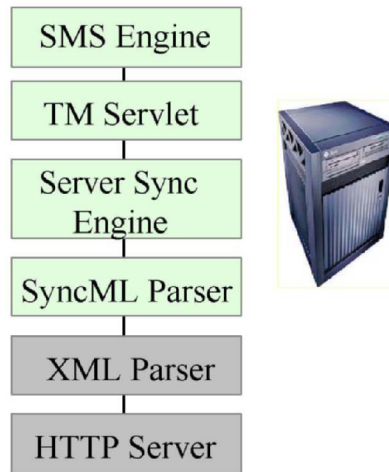


Figure 6 Terminal Management Server

The Terminal management server shown in Figure 6 is a conventional HTTP Web Server with a Servlet Engine [23]. It receives SyncML messages sent over HTTP, parses them and extracts a set of results and status indicators of the management operations. These results are then sent to the multi-protocol gateway. The SMS Engine is basically a Java mail based mechanism to create a SMTP message. This mechanism was used to trigger the TM Midlet of the phone. For devices other than phones, the server, upon receiving a management action, sends periodic UDP messages to TM agent. Along with security information this UDP packet contains the IP address that the TM Agent will connect back for SyncML exchanges.

## 5 Seamless Management Flow

Figure 7 shows a sequence of exchanges between various components of the Universal Manager. These sequences of steps are explained below.

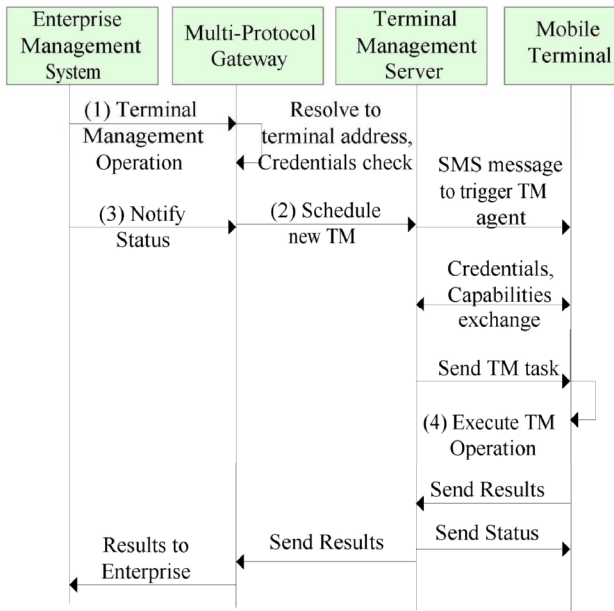


Figure 7 Flow of operations

### (1) Terminal Management Operation

For a terminal management action, such as the querying of the manufacturer name of a GSM terminal, the enterprise management system is unaware of the presence of such a terminal and either of the following operations takes place.

1. The enterprise management system based on additional information such as IMEI decides to route management action to multi-protocol gateway, or
2. Multi-protocol gateway advertises addressability to that IMEI as a certain IP address and terminal management action takes place on this proxy IP address.

In either case the enterprise management system routes the SNMP PDU for the corresponding management action to the Multi-Protocol Gateway instead of sending the SNMP PDU directly to the wireless device. The SNMP PDU's host and port are set to the host and port (in our case 15003) that the Multi-Protocol Gateway is listening on as opposed to the host and port of the wireless device. The SNMP Packet thus sent contains the Operation Name (e.g. Get, Get Next, Set), the OID to be

queried, (optionally IMEI of the GSM terminal and the value of the OID if it is a Set Operation), SNMP credentials and a request ID of the SNMP Packet.

Figure 8 shows the result of querying the Object ID “./DevInfo/Man” of a mobile device from the enterprise management system. The result that is the manufacturer name is shown in the result panel.

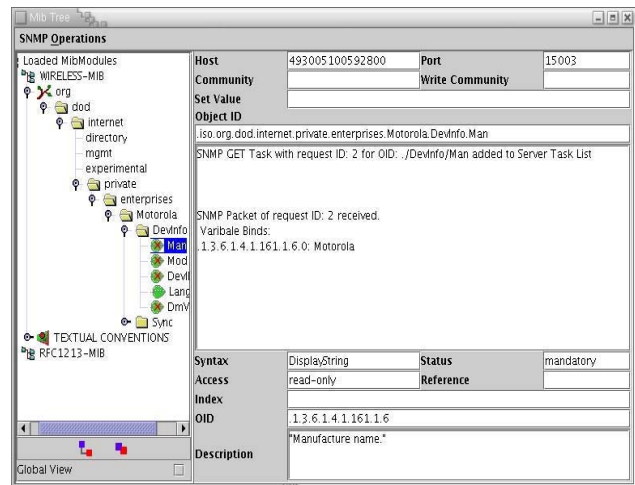


Figure 8 Management of mobile device using the Universal Manager

### (2) Protocol Conversion

On receiving a SNMP Packet, the Multi-Protocol Gateway extracts the required information from the SNMP PDU. The Multi-Protocol gateway performs a credentials check and maps the SNMP OID to the corresponding SyncML management tree node. In case the IMEI is not sent, it maps the destination IP address to the IMEI of the device. The Protocol Converter after fetching the corresponding SyncML parameter hands the same over to the Scheduler that schedules the task on the TM Servlet. The Scheduler uses HTTP Get to send the request to the TM Servlet. The parameters sent as part of the HTTP Get Request to the TM Servlet include the Operation Name i.e. (Get, Get Next, Set), SyncML management tree node, the value of the SyncML parameter if it is a Set Operation, IMEI of the phone to be queried and the Request ID (associated with each SNMP PDU) of the Operation.

Figure 9 shows a part of the SyncML command actually sent to the mobile device to retrieve the manufacturer name from the management tree of the device.

```

<Get>
  <CmdID>2</CmdID>
  <Meta>
    <Format xmlns="syncml:metinf">
      application/vnd.syncml-devinf-
        devicemanagement+xml
    </Format>
  </Meta>
  <Item>
    <Target>
      <LocURI>./DevInfo/Man</LocURI>
    </Target>
  </Item>
</Get>

```

**Figure 9 SyncML Command to get manufacturer name**

### (3) Status Notification

The Notifier component of the Multi-Protocol Gateway now sends back a status to the enterprise management server informing it that the task has been added to the TM SyncML Server and hence prevents time outs and retransmission of SNMP PDUs. The status information is sent to the port 15001, where the enterprise management server is listening for information from the Multi-Protocol Gateway.

### (4) SyncML Protocol exchanges

The SyncML Servlet maintains all the requests coming from the Multi-Protocol Gateway and depending on the schedule time, it sends a trigger message to the mobile terminal. The trigger message activates the TM agent and it connects to the TM Server. There is a credentials and capabilities exchange followed by sending of the management action to the TM agent as part of the SyncML package. The TM agent at the mobile terminal executes the requested operation and sends back the results and status to the TM Server.

### (5) Notification of results

The Notifier component of the Multi-Protocol Gateway consists of a Poller component that polls the TM Server via HTTP Get to check if the values have been received

from the mobile terminal. The Poller sends the HTTP Get querying the TM Server for results. If the values have been received from the phone, the TM Server sends them via HTTP to the Multi-Protocol Gateway. The Poller mechanism was used as the enterprise management system was behind a firewall.

### (6) Results at the enterprise manager

The Protocol Converter at the Multi-Protocol Gateway converts results from the TM Server to SNMP PDUs and sends the SNMP Packet to the enterprise management server listening on port 15002 for response SNMP PDUs. The response PDU also contains information on the requested Command either Get/Get Next/Set, the request ID associated with the request SNMP Packet, SNMP OID, value associated with the OID and the status if the operation executed on the phone was successful or not. On receiving the response SNMP Packet from the Multi-Protocol Gateway the packet information is extracted and the results are displayed on the enterprise management server GUI.

Figure 10 shows a part of the wireless device MIB that must be part of the enterprise MIB to enable management actions.

```

WIRELESS-MIB DEFINITIONS ::= BEGIN
IMPORTS MODULE-IDENTITY,
        OBJECT-TYPE, Integer32,
        enterprises
FROM SNMPv2-SMI
DisplayString, PhysAddress
FROM SNMPv2-TC;

WIRELESS-MIB MODULE-IDENTITY
LAST-UPDATED "200310200000Z"
ORGANIZATION "Motorola"
CONTACT-INFO "Sandeep Adwankar..."
DESCRIPTION
  "This MIB module defines a MIB which
  provides mechanisms to perform SNMP
  management operations on mobile devices."
 ::= { enterprises xxx }

DevInfo OBJECT IDENTIFIER
 ::= { WIRELESS-MIB 1 }
DevDetail OBJECT IDENTIFIER
 ::= { WIRELESS-MIB 2 }
Ext OBJECT IDENTIFIER ::= { DevDetail 1 }

```

```

WAP OBJECT IDENTIFIER ::= { Ext 2 }

-- the DevInfo group

Man OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (0..255))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "Manufacture name."
    ::= { DevInfo 1 }

Mod OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (0..255))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "Model name."
    ::= { DevInfo 2 }

DevID OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (0..255))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "Device serial number."
    ::= { DevInfo 3 }

-- the DevDetail group

OEM OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (0..255))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "Specify OEM of the
                 device (e.g., Motorola)."
    ::= { DevDetail 1 }

FwV OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (0..255))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "Firmware version."
    ::= { DevDetail 2 }

-- the WAP group

Defaultwebsession OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (0..255))
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION "Set default web session on phone."
    ::= { WAP 1 }

...
END

```

**Figure 10 Part of Wireless MIB for mobile devices**

## 6 Performance

The Universal Manager is deployed in our enterprise and is used to perform management actions on mobile and non-mobile devices. The time taken for a management action on a non-mobile device is observed to be very small (in milliseconds) compared to the time taken on a wireless mobile device. Hence the performance result for the management action on a mobile device is presented. **Table 3** details our experience in running a management operation from the enterprise manager on a set of wireless mobile terminals over a GSM network. The management operation in this case is changing the default web session on a mobile device to a value that is at index 3. The SyncML management command is shown in Figure 11.

```

<Replace>
  <CmdID>1</CmdID>
  <Meta>
    <Format xmlns="syncml:metinf">
      application/vnd.syncml-devinf-
        devicemanagement+xml
    </Format>
  </Meta>
  <Item>
    <Target>
      <LocURI>
        ./Sync/DM/WAP/DEFAULTWEBSSESSION
      </LocURI>
    </Target>
    <Data> 3 </Data>
  </Item>
</Replace>

```

**Figure 11 SyncML Command to replace default web session**

The results shown in **Table 3** are for ASCII representation of SyncML messages sent over GSM as a wireless bearer. The table shows universal manager operation flows, size of data sent in bytes and time taken to send data over GSM network in seconds. (Time taken in milliseconds range is ignored.) These results should improve with binary representation (WBXML) of SyncML messages. Higher data rate wireless networks should improve the performance of the system.

**Table 3. Performance results of management of GSM mobile device from Universal Manager.**

	<b>Universal Manager Operation flow</b>	<b>Size</b>	<b>Time</b>
0.	<b>Operator initiates management action on enterprise manager</b>		
1.	<b>Mapping mobile device destination</b> The enterprise manager maps the destination address to determine the management action destined for a mobile device.		
2.	<b>Routing SNMP PDU</b> The enterprise manager sends SNMP PDU to the Multi-protocol gateway.	<b>30</b>	
3.	<b>Protocol conversion and Scheduling</b> The Multi-protocol gateway decides the exact protocol to be used and accordingly routes and schedules the management action on the corresponding terminal management server.	<b>23</b>	
4.	<b>Creation of SyncML package 0</b> The TM server based on the schedule information sends a SMS message or a UDP packet to trigger the mobile device.		
5.	<b>Creation of SyncML package 1</b> The SyncML Engine on the mobile device on receiving package 0 creates a SyncML package with capabilities data such as Manufacturer name, Model name etc. and credentials.	<b>1077</b>	<b>3</b>
6.	<b>TM Server sending management action</b> The Mobile device sends the SyncML package 1 over HTTP to the TM Server. The TM Server parses the package, checks for credentials and management commands that need to be sent to the device. It creates a new SyncML package and sends it as HTTP reply.	<b>1656</b>	<b>16.1</b>
7.	<b>Creation of SyncML package</b>	<b>645</b>	<b>7.4</b>

	<b>2</b> The syncEngine on a mobile device parses incoming message and determines the management action to perform. The mobile device then performs the management action by accessing firmware. The Sync Engine composes the results and status of management action in a SyncML package and sends it to the TM server.		
8.	<b>TM Server sending status</b> The TM Server on receiving the result/status responds with its own status (and/or any more management actions) in a SyncML package.	<b>525</b>	<b>10.6</b>
9.	<b>Mobile device processing status</b> Mobile device parses SyncML message (and processing steps 3 and 4 in case of any more management action)		<b>2</b>
10.	<b>Notification of results to the enterprise manager</b> The TM Server notifies the results to the Multi-protocol gateway. The Notifier component of the Multi-protocol gateway sends the results to the enterprise manager		
11.	<b>Enterprise manager displays the results</b>		

## 7 Conclusions

The growth of mobile devices in the enterprise and their use in the enterprise business processes is giving rise to new questions of security and management of enterprise resources. With evolving technologies that enable seamless mobility across multiple networks, enterprises need a uniform solution for all of their enterprise devices. A uniform protocol across all mobile and non-mobile devices is infeasible as management protocols are optimized for a certain family of devices. Our solution is to maintain a diversity of management protocols along with a diversity of devices while providing an architecture that provides seamless management of all enterprise elements. Such a Universal Manager enables atomic transactions that are executed across a large scale

of diverse devices. It allows distributed diagnostics across heterogeneous hosts and protocols. The Universal Manager allows maintaining existing investment in the enterprise management system while providing multi-device coordination and management transaction control.

Our experience with SyncML DM protocol for resource-constrained wireless handhelds like GSM phones shows a promising management protocol for those set of devices. The SYMPLE platform that we built is small, flexible, and extendable and is able to perform a wide set of configuration, software distribution and diagnostics tasks. The multi-protocol gateway seamlessly integrated SyncML based mobile devices with the conventional SNMP based enterprise manager. The Universal Manager is deployed in our enterprise over a period of time and the system performance is reliable, scalable and consistent. The system performance studies show that most of the system performance delays can be attributed to low data rate of underlying wireless network. The Universal Manager allowed blending of heterogeneous devices to form a scalable enterprise management infrastructure.

## Acknowledgements

We would like to thank Hung Tsang and Kevin Cutts for their help with the Motorola GSM platform. We would also like to thank Prof. Ashfaq Khokhar for useful discussions.

## References

1. 3<sup>rd</sup> Generation Partnership Project Specifications, <http://www.3gpp.org/specs/specs.htm>
2. 3<sup>rd</sup> Generation Partnership Project 2 Specifications, [http://www.3gpp2.org/Public\\_html/specs/](http://www.3gpp2.org/Public_html/specs/)
3. M. Lott, M. Weckerle, W. Zirwas, H. Li, E. Schulz. Hierarchical Cellular Multihop Networks, EPMCC 2003.
4. Context Transfer, Handoff Candidate Discovery, and Dormant Mode Host Alerting Charter, <http://www.ietf.org/html.charters/seamoby-charter.html>
5. J. Case, M. Fedor, M. Schoffstall, J. Davin. A Simple Network Management Protocol. RFC 1157, Internet Engineering Task Force, May 1990.
6. K. McCloghrie, M. Rose. Management Information Base for Network Management of TCP/IP-based internets:MIB-II. RFC 1213, Internet Engineering Task Force, March 1991.
7. W. Stallings. SNMP, SNMPv2, SNMPv3, and RMON 1 and 2. 3rd ed. Addison-Wesley, 2000.
8. Web-Based Enterprise Management (WBEM) Initiative, [http://www.dmtf.org/standards/standard\\_wbem.php](http://www.dmtf.org/standards/standard_wbem.php)
9. M. Rose The Blocks Extensible Exchange Protocol Core, <http://www.ietf.org/rfc/rfc3080.txt>
10. D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. K. Nielsen, S. Thatte, D. Winer, Simple Object Access Protocol (SOAP) 1.1, <http://www.w3.org/TR/SOAP/>
11. Project JXTA, <http://www.jxta.org/>.
12. Network Configuration, NetConf charter, <http://www.ietf.org/html.charters/netconf-charter.html>
13. JUNOScript, Example of a JUNOScript Session, <http://www.juniper.net/techpubs/software/junos53/junoscript53-guide/html/session-guide37.html>
14. Provisioning Architecture Overview Version 1.1, 2002, <http://www.openmobilealliance.com>
15. The SyncML Device Management Protocol, version 1.1.2, Open Mobile Alliance Ltd. June 2003
16. The SyncML Representation Protocol Device Management Usage, version 1.1.2, Open Mobile Alliance Ltd. June 2003
17. AdventNet SNMP Agent Toolkit, <http://www.adventnet.com>.
18. Mobile Information Device Profile (MIDP) <http://java.sun.com/products/midp/>
19. V. Vasudevan, S. Adwankar, N. Narsimhan. MobiMan: Bringing Scripted Agents to Wireless Terminal Management. 14<sup>th</sup> IFIP/IEEE International Workshop on Distributed Systems: Operations & Management, October 2003
20. The SyncML Device Management Security, version 1.1.2, Open Mobile Alliance Ltd. June 2003
21. Tiny XML Parser, <http://www.gibardunn.srac.org/tiny/index.shtml>
22. SAX, Simple API for XML, <http://www.saxproject.org/>
23. The Apache Software Foundation, <http://www.apache.org>