RESEARCH ARTICLE

OMA DM v1.x compliant Lightweight Device Management for Constrained M2M devices

Nhon Chu¹,²∗, Djelal Raouf¹, Bruno Corlay¹, Mohamed Ammari¹, Nenad Gligoric², Srdjan Krco², Nemanja Ognjanovic³, Aleksandar Obradovic³

¹ Sierra Wireless France, 5 bd Gallieni, 92442 Issy-les-Moulineaux, France
² Ericsson d.o.o Milentija Popovia 5a/v, 11070 Belgrade, Serbia
³ Telekom Serbia, Takovska 2, Belgrade, Serbia

ABSTRACT

This paper presents ELFO (Exalted Lightweight For OMA DM v1.x), an alternative lightweight solution that is fully compatible with the current OMA DM protocol and with all existing OMA Management Objects (MOs). This solution aims to enable operators and service providers to leverage their existing OMA DM servers to continue supporting current OMA DM enabled mobile devices and to incrementally support new constrained M2M devices through an added proxy server. ELFO reduces the size of OMA DM messages by 89% while optimizing the processing performance and resource consumption.

Copyright © 2013 John Wiley & Sons, Ltd.

*Correspondence
Nhon Chu, Sierra Wireless France, 5 bd Gallieni, 92442 Issy-les-Moulineaux, France.
E-mail: nchu@sierrawireless.com

1. INTRODUCTION

Open Mobile Alliance Device Management (OMA DM) is well established in the mobile value chain. OMA DM helps operators, enterprises and device vendors to manage access capabilities, diagnose problems, fix and update devices over the network. In February 2012, OMA confirmed the deployment of its device management (DM) mobile service enablers in 1.4 billion mobile phones and connected devices worldwide [1].

The growing number of M2M devices is expected to be much larger than the number of mobile devices. Machina Research forecasts that the installed base of M2M Connected Consumer Electronics devices will exceed 4.2 billion by 2020 [2]. Cisco created an infographic [3] that depicts the increasing number of things connected to the Internet. Similar to Ericsson predictions [4], it states that by 2020, there will be 50 billion ‘things’ connected to the Internet. The Internet of Things (IoT) covers everything from body, car, alarm clock and even cows. Unlike mobile devices, these devices are mostly constrained: small, low-cost and have limited resources, e.g. battery-operated, 8-bit microcontroller, with a small amount of memory.

OMA Device Management protocol v1.2 [5] or v1.3 [6] is being widely used to manage mobile devices. However it is not suitable for constrained devices and their networks, due to the verbosity of messages, encoding and protocol complexity. Therefore, new standards for M2M technology are being developed in order to sustain the exponential growth of M2M wireless communications market with lightweight device management protocol being one of the key focuses of standardization efforts. These initiatives and trends are briefly described in section 2.

This paper investigates a lightweight device management method using servers that implement OMA DM protocol v1.2 or v1.3. Operators can leverage this novel method to reuse their existing OMA DM servers to manage new M2M devices while continuing to manage the current OMA DM enabled mobile devices. All OMA DM Management Objects can be leveraged (e.g. FUMO, GwMO, DiagMO, SCOMO, etc.) to support additional services. M2M Devices residing in a capillary network and having no IP address can be managed through a M2M Gateway using OMA Gateway Management Object enabler (GwMO) [7]. The proposed method, described in section 4, essentially aims to reduce both the size of OMA DM message payload and the encoding/decoding complexity. Consequently, this solution enables communication cost savings and more importantly it helps preserving the M2M device battery life as the data transmission time is significantly shortened along with low processing complexity. Additionally, the amount of required memory and the CPU
processing capability would both be reduced in the device, thus lowering the device cost in general. The proposed solution is detailed in section 4 and an implementation option is described in section 5. Performance results and benchmarks are provided in section 6.

2. STATE OF THE ART

This section provides an overview of the most important standardization efforts on M2M constrained device management as well as other scientific works on M2M/IoT simplification.

2.1. OMA Lightweight M2M

OMA Device Management technology is evolving from traditional mobile devices networks to heterogeneous networks that support both mobile and M2M devices. In June 2011, OMA has approved a new work item (WID 246) "Lightweight Machine to Machine" [8] to define a lightweight M2M solution in order to address the following challenges:

1. Support capability constrained devices. Most M2M devices being deployed in M2M solutions have limited capabilities (e.g. 8-bit microcontrollers, small amount of RAM, battery operated and lightweight operating system). Due to the huge number of devices to be deployed, stakeholders tend to be very sensitive to the cost of devices. In addition, M2M services are mainly focused on data collection and remote controlling without heavy processing and user interactions, therefore M2M devices do not need high capability. The lightweight M2M protocol will also support various security models to adjust to different needs. Lightweight security models can lead to further device resource savings.

2. Preserve battery consumption. Constrained devices are mostly battery-operated. Thus, in addition to energy efficiency, the solution must also enable efficient communication with devices that are likely to have long sleeping cycle to preserve battery life.

3. Optimize network resources. Due to the exponential growth of M2M devices, a very large number of devices may be connected to the communication network simultaneously.

4. Unique Device identifier. This is necessary for identification in the service layer. The identifier is translated to a network number, which is addressable based on the available access technology (IP, Non-IP).

Interesting orientations have been adopted at the early stage of the work to reduce the payload and to simplify the encoding scheme; for instance, binary based addressing scheme has been adopted instead of the URI and flat data model for efficient data access.

2.2. OMA DM NG (Next Generation)

OMA has also approved a work item called DM NG (Next Generation) v1.1 [9]. It is targeted to greatly improve the efficiency of the existing DM protocol. Numerous work areas have been identified for this work item. In the previous DM NG v1.0, there were interesting contributions which aimed at reducing the complexity of the DM protocol (v1.2 and v1.3) and reducing the message size:

1. Some contributions suggested the use of JSON to encode DM NG messages. In one study, eMail management objects returned by the device have been represented with JSON encoding.

2. Other contributions suggested protocol simplification by skipping some parts of the OMA DM message. This simplification is applicable to periodic messages. For instance, authentication information can be substituted by using a shorter token. When a device is reporting data to a server on a regular basis, some parts of the message, which can be assumed to be unchanged within the session (e.g. target, meta), can be skipped in the subsequent reports. Another improvement suggestion can be applied to the Generic Alert command. This simplification, if applicable, consists in reporting the output data in the Alert body. An extra round-trip can therefore be saved as the server does not need to query the output data in the next request. The protocol and payload size can be further simplified by not sending the DevInfo management object on every request.

3. An earlier contribution suggested that OMA DM protocol can be transitioned to REST, as they are both resource oriented, and many OMA DM commands/ features can be mapped to the REST methods. A later contribution [10] introduced CoAP, Constrained Application Protocol (defined by IETFs CoRE working group [11]), with the intention to jumpstart the discussion on adapting DM NG for M2M Applications. As CoAP [11] is designed for use with constrained devices and networks for M2M applications, this contribution has also been proposed to the DM Lightweight M2M work item.

2.3. Compressing SyncML messages

An alternative approach to reduce the verbosity of OMA-DM packages consists in converting SyncML messages (XML documents) into a binary representation format. The conversion is reversible, namely, the binary representation can be converted back to the original SyncML document without loss of information. WBXML [12], EXI [13] and Fast Infoset [14] specifications aim to optimize both the document size and processing performance.

Wap Binary XML (WBXML) [12] is a XML compression scheme recommended by OMA. This binary encoding scheme was initially developed by WAP Forum
and is now maintained by OMA (Open Mobile Alliance). WBXML is used by mobile phones to transmit XML documents (e.g. OMA DM packages, OMA DRM, Wireless Markup Language Sync ML) in a compact manner over mobile networks. WBXML has also been proposed as an addition to the W3C WAP.

Fast Infoset [14] (or FI) is an international standard that specifies a binary encoding format for XML text documents. FI specification is defined by both ITU-T and ISO standards bodies.

Efficient XML Interchange (EXI) [13] is a data format proposed by the Efficient XML Interchange Working Group (EXI WG) of the World Wide Web Consortium (W3C). An advantage of EXI over Fast Infoset is that EXI can use the schema-informed mode to further compress XML documents. However, this option requires the server and devices to share the same schema for encoding and decoding.

Other XML compression techniques and studies have been analyzed by various papers [15], [16], [17], [18].

2.4. Other scientific works

Other scientific papers related to M2M/IoT have been considered [19], [20], [21], [22], [23], [24]. In [19], lightweight data synchronization framework is proposed to enable concurrent read and update data operations on RFID devices. The paper [20] proposes MINDiT, a framework that provides a common abstract interface towards the communication support with different entities. A lightweight TLV-based (Type Length Value) interface, derived from MIH (Media Independent Handover introduced by IEEE 801.21), is used to communicate with IoT devices. The payload includes: the Source identifier TLV, the Destination identifier TLV and MIH service specific TLVs. The header contains, amongst others, a service identifier, an operation code and an Action identifier. Action value supports sequence of parameters, which can be adapted to create DM commands. The paper [21] is comparing REST versus SOAP-based web services from a developer point of view. In [22], guidelines are given to leverage clustering techniques along with software agents and data synchronization to enhance the energy efficiency, scalability and robustness. The paper [23] describes an auto-regulation monitoring system that makes use of Zigbee to transmit data from device to the Gateway and then from Gateway to remote server. Paper [24] recommends a layered architecture framework to manage RFID readers and tags, using SNMP protocol RFID Managing Protocol (RFID-MP).

2.5. Observations

OMA Lightweight DM M2M work item (2.1) [8] has started and requirements are being defined. Initial orientations to simplify the protocol and reduce the payload are so far promising (e.g. binary based addressing scheme and flat data model for efficient data access). However, this solution is not backward compatible with OMA DM v1.x.

Improvements suggested by OMA DM NG (2.2) certainly lead to simplification of the DM protocol along with the message payload reduction enabling a significant network bandwidth optimization. However, the proposed protocol simplification implies important changes on the existing OMA DM servers in order to achieve the backward compatibility initially required in the DM NG v1.0.

Solutions leveraging compression of SyncML messages (2.3) share the same objectives as ELFOMA. They aim to minimize the payload size with efficient processing while maintaining compatibility with OMA DM v1.x. EXI is an efficient XML compression method and can generate very compact OMA-DM messages. ELFOMA performances will be compared with these solutions (2.3).

Scientific papers (2.4) are mostly framework proposals. Device management related procedures such as bootstrapping, firmware update, lock and wipe, etc are not covered. Moreover, the proposed frameworks and protocols are not suitable to achieve the aim of this paper, which consists in proposing a lightweight solution that reuses existing OMA DM servers and enables. The TLV-based encoding, used in [20], is a good option to represent data model with small footprint, the information needs to be packed as structure with well-defined data order. However, this is not optimized for fine-granularity data access.

3. SYSTEM DESCRIPTION

The system depicted in Figure 1 is reusing the current OMA DM v1.x server (a) “as is” to support both the existing OMA DM v1.x enabled mobile devices (b) and the new constrained M2M devices (c). An OMA DM adapter proxy (d) is introduced between the constrained M2M devices (c) and the server (a). M2M devices (c) only support ELFOMA Lightweight messages. Device originated messages are converted by the proxy (d) to OMA DM protocol messages prior to being redirected to the server (a). In the opposite direction, server originated OMA DM messages are converted back to ELFOMA by the proxy and sent to the device.

In section 4, a step-by-step approach shows how OMA DM messages can be mapped onto ELFOMA Lightweight messages while achieving a significant payload reduction. Section 5 describes an implementation option of this message translation mechanism in the OMA DM Adapter Proxy. This latter entity is acting as an OMA DM client and an ELFOMA server to perform 2-ways OMA-ELFOMA DM message translation in order to maintain the envisaged compatibility. ELFOMA client is introduced. The proxy configuration is also covered in this section.

The proposed concept is fully compliant with the existing OMA DM v1.x server, namely:

DOI: 10.1002/ett
Prepared using etauth.cls
OMA DM v1.x compliant Lightweight Device Management for Constrained M2M devices

- All OMA DM Management Objects can be reused without any change
- OMA DM Setup phase and Management phase remain unchanged
- No protocol simplification is needed or introduced
- Fully compliant with the SyncML security framework: server and client mutual authentication, authentication challenge, messages integrity check using server nonce, client nonce and hashed secret, protection against replay attacks using MessageID, inclusion of encrypted data in OMA DM package, etc.

The key advantage of this solution is to reuse the existing OMA DM v1.x servers "as is" to incrementally support new constrained M2M devices. The management of fleet of existing mobile devices is not affected.

4. PAYLOAD REDUCTION

OMA DM v1.x messages are SyncML (Synchronization Markup Language) documents [25], which are based upon standardized XML communication and thus can be implemented on any platform. However, the message payloads are verbose. In addition, potentially long URIs and tree-based data model further inflate the payload size. To achieve the payload reduction, a possible approach is to use the OMA DM adapter proxy to compress SyncML-based messages to binary-based messages. The candidate XML compression methods are described in section 2.3.

4.1. Message Translation

While the aforementioned XML compression methods can be applied to all types of XML document, the proposed ELFOMA encoding aims to further optimize both the OMA DM v1.x message payload size and the processing performance. Unlike the above XML compression methods, space efficiency is not inversely proportional to processing efficiency. Space efficiency refers to the memory requirements of a processor implementing a new format with respect to that of a processor implementing XML. Processing efficiency refers to the speed at which a new format can be generated and/or consumed for processing with respect to that of XML. ELFOMA is defined with the following approaches:

1. ELFOMA message content is a plain text format. Further compression can therefore be applied on top if needed, e.g. HTTP compression.
2. The SyncML Representation protocol [25] has specified a Content Model for each SyncML element type. OMA DM messages are then serialized to SyncML based on a DTD which reflects the aforementioned content models. To achieve a full compatibility with OMA DM v1.x messages, the lightweight approach consists in using the same content models to serialize lightweight messages using a simple encoding scheme. Figure 2 depicts the concept.

For instance, Figure 3 shows the Content Model for SyncHdr element type. This model specifies all child elements that are mandatory or optional (marked with "?") to define the SyncHdr element type in an OMA DM message. Figure 4 is an excerpt of an OMA DM payload highlighting the SyncHdr element type. Please note that the RespURI and NoResp elements are optional and are not specified in the payload. In this example, the payload size is 519 bytes. SyncML element types having more than 1 child element are represented in the ELFOMA using CSV (comma separated value) based syntax. SyncHdr element, Command elements (e.g. Get, Replace, Alert), Response Command elements (e.g. Status, Results) are mapped to a CSV based syntax. Figure 5 shows ELFOMA syntax for the SyncHdr element type. Please note that all child elements defined by the content model in Figure 3 are taken into account. This replication is necessary to achieve ELFOMA full compatibility with OMA DM.

3. Values of child elements (e.g. VerDTD, SessionID, MsgID) are directly placed into CSV fields. Figure 6 shows how the OMA DM payload (Figure 4) is translated to ELFOMA using CSV (comma separated value) based syntax. SyncHdr element, Command elements (e.g. Get, Replace, Alert), Response Command elements (e.g. Status, Results) are mapped to a CSV based syntax. Figure 5 shows ELFOMA syntax for the SyncHdr element type. Please note that all child elements defined by the content model in Figure 3 are taken into account.

4. A CSV field could be left empty if no value is assigned to the element type. In Figure 6, the
RespURI and NoResp fields are left empty, as they are not specified. When converting ELFOMA back to a SyncML message, optional child element types being left blank do not appear in the SyncML message. However, the OMA DM adapter proxy could assign a default value to the mandatory CSV field being left empty.

5. Optional element types positioned in last CSV field(s) could be omitted.

6. Curly brackets can be used to pack complex child element types into a CSV field. As shown in Figure 5, Cred element is composed of a Meta element and a Data element. As the Meta element has at least 3 child elements (Type, Format and application data), curly brackets are used to separate a Meta field from the next element. Data. Figure 6 shows the use of curly brackets to map the Cred element values. Note that the content model of the Meta element type has been extended to include Type and Format child elements.

7. "SyncML" and "SyncBody" message container elements are not used. All commands listed after the SyncHdr are considered as part of the SyncBody. When converting ELFOMA back to OMA DM v1.x, the OMA DM adapter proxy automatically inserts the "SyncML" element with the proper
Figure 7. OMA DM payload Replace 1170 bytes

value (e.g. Syncml:Syncml1.2). Likewise, the “SyncBody” element is automatically inserted by the proxy.

8. Namespaces are essentially used in the Meta elements to define custom variables. Constrained devices are likely to deal with variables and values (key/value) only, while namespace may be dropped. For this reason, the mapping of the namespaces is handled by the OMA DM adapter

proxy and variable name collision is avoided. The namespace definition (xmlns=syncml:metinf) has therefore been discarded in ELFOMA, as shown in Figure 6. When converting ELFOMA back to the SyncML messages, namespaces will be inserted based on the namespace mapping tables kept by the OMA DM adapter.

9. The curly brackets “{ ” ”}” are used to specify a list of child elements of the same type (e.g. Item+). Figure 7 depicts an excerpt of an OMA DM message with a Replace command applying to 5 variables. The counterpart Lightweight version is shown in Figure 8.

10. Square brackets ”[ ” ”]” can be used to factorize common elements. Fields to be factorized are positioned using CSV syntax. For instance, the following 2-fields expression:

../DevInfo/DevID;soft/v1./DevInfo/Man;soft/v2

can be factorized as follows:

[../DevInfo/;soft]:DevID;v1Man;v2.

11. Nested child commands can be specified using curly brackets. High level commands such as Atomic and Sequence can include several commands. The Atomic command below contains one Replace and two Exec commands:

Atomic=CmdID;NoResp?;Meta?;
Replace=CId2;nores?;cred?;meta?;item1item2
Exec=CId2;nores?;cred?;meta?;correlator?;item
Exec=CId3;nores?;cred?;meta?;correlator?;item

12. The extended Meta element type has “Type” and “Format” fields which can be respectively set by default to “chr” and “text/plain” by the OMA adapter proxy. Hence, the lightweight payload in Figure 8 is further reduced to 154 bytes:
13. If the Data element type value contains special characters pertaining to ELFOMA syntax domain, i.e. "," ";" "[" "]" ""]:" "", then the value shall be quoted, for example "IMEI:493005100592800"

14. The SourceRef and TargetRef element types specify a Source or a Target referenced by the Status or Results element type. If the SourceRef is the same as the Source in the SyncHdr or TargetRef the same as the Target in the SyncHdr, then they can be omitted to reduce the payload. In this latter case, the OMA DM adapter will be using the Source and the Target in the SyncHdr to specify the TargetRef and the SourceRef.

4.2. Further optimizations

The proposed lightweight encoding can be further optimized with the following 3 approaches:

The SyncML command element types can be represented in the ELFOMA using the token codes defined for WBXML usage. This token definition can be found in the SyncML Representation protocol document [25]. The "Replace" protocol command element in Figure 8 can be replaced by token 20. The "SyncHdr" message container element in Figure 5 can be replaced by token 2C. Likewise, token codes can be used to specify the SyncML command (Cmd) referenced by a Status element type.

OMA Management Objects are tree-based, e.g. /x/DownloadAndUpdate/PkgURL (a FUMO node). This model implies the constrained device to maintain and parse the tree. This model also contributes to inflate the payload size. The OMA adapter proxy can handle a mapping between a tree-based data model and a flat data model. The use of the flat data model saves resources on the device and reduces the payload size. Tree-based nodes, referenced by a URI, can be mapped to node codes which reflect the flat data variables. The node codes are specified in the Source or Target element types within the Item element type of a given SyncML command. Constrained devices can use 2 hexadecimal characters (8-bit) node code, to map up to 256 flat data variables. More hexadecimal characters can be used to address more flat data variable without any backward compatibility issues. URI referenced by the RespURI element type or by a Target or a Source element type within a SyncHdr can be long. OMA DM adapter proxy can handle a mapping between the original long URI and a URI short code or an alias. OMA DM Adapter proxy must be provisioned with the node codes and the URI short codes mapping.

For further details, refer to configuration in section 5. The original tree based data and the long URI will be used as is, if no mapping is provided in the proxy.

Values for the Type and Format elements e.g. "chr;text/plain", "auth-basic;b64" can be mapped to tokens. This contributes to reduction of the payload pertaining to the Meta element type. In Figure 9, the Type "auth-basic" has been replaced by an 8-bits token (1). The Format "b64" has also been replaced by a token. By default, the pair of Type/Format tokens (0;0) represents (chr; text/plain). In paragraph 4.1.8, namespaces are removed. They are taken into account to define tokens. Therefore an element Type belonging to the namespace Syncml will not share the same token as an element Type of a different namespace. Applying the aforementioned optimizations, Lightweight payload of Figure 6 and 8 is further reduced as shown in Figure 9 and 10 above. The source element type value in Figure 6 "http://www/syncml.org/mgmt-server" has been replaced by an alias "mgmt-server" in Figure 9. A shorter alias or a token could be used. The MaxMsgSize meta-information has been replaced by a meta-token, 1D. The payload size has been reduced: 146 bytes down to 95 bytes. Its size is now 5.46 times smaller than the counterpart OMA DM payload (519 bytes), Figure 4. In Figure 10, the Replace command element in Figure 8 has been replaced by a token (20). The tree-based DevInfo management objects (/DevInfo/DevId, /DevInfo/Man, etc) have been replaced by node codes (2A, 2B). The factorization in Figure 8 is no longer required, as the /DevInfo path prefix is integrated into the node code, and the Meta information uses the default values (chr; text/plain, as stated in section 4.1.12). The payload size in Figure 8, 180 bytes, has been reduced to 122 bytes. The size of the OMA DM counterpart payload in Figure 7 was 1170 bytes: 9.6 times larger.

Finally, less frequently used elements can be placed at the end of the content models. Therefore, comma signs used to mark empty or default fields can be saved. This requires a reordering of the content elements in the models used by ELFOMA. The concept of Figure 2 has to be extended to handle two Content Models, as shown in Figure 11. Conversion between the ELFOMA elements Content models and the SyncML elements content models is handled by the OMA DM adapter proxy. For example: SyncML Content Model for Item element type:

```
Item: (Target?, Source?, SourceParent?, TargetParent?, Meta?, Data?)
```

is mapped to ELFOMA Content Model:

```
Replace=3; ;[;/DevInfos;;;;];
;DevId;;;;"IMEI:493005100592800"
;Man;;;;Device Factory, Inc.
;Mod;;;;SmartPhone2000
;DmV;;;;1.0.0.1;Lang;;;;en-US
```

```
2C=1.2;DM/1.2;/2;mgmt-server;"IMEI:493005100592800";
;1;[;/1;J3Y2uG
k9oQmVoYXZ1i;[;1D=5000]
```

```
Replace=3; ;[;/DevInfos;;;;];
;DevId;;;;"IMEI:493005100592800"
;Man;;;;Device Factory, Inc.
;Mod;;;;SmartPhone2000
;DmV;;;;1.0.0.1;Lang;;;;en-US
```

```
2C=1.2;DM/1.2;/2;mgmt-server;"IMEI:493005100592800"
;1;[;/1;J3Y2uG
k9oQmVoYXZ1i;[;1D=5000]
```
SyncML Content Model for Replace cmd element type:

Replace: (CmdID, NoResp?, Cred?, Meta?, Item+)

is mapped to ELFOMA Content Model:

Replace: (CmdID, Item+, Meta?, Cred?, NoResp?)

The exception to handling the Meta content model differently, as described in 4.2.6, can now be handled:

SyncML Content Model for Meta element type:

Meta: (#PCDATA)

is mapped to ELFOMA Content Model:

Meta: (Type, Format, #PCDATA2).

Where #PCDATA2 does not include Type and Format elements.

These changes have been applied to ELFOMA content models for the following element types: SyncHdr, Meta, Alert, Replace, Item, Status, Get, Atomic, Sequence and Exec element types as well. Reordering should be applied to other element types as well. The payloads in Figure 9 and Figure 10 now have more compact forms as shown in Figure 12 and Figure 13. The size of the replace command is now 11.25 times smaller than its OMA DM counterpart payload, i.e. 104 bytes vs. 1170 bytes.

5. IMPLEMENTATION & CONFIGURATION

Entities (a) and (b) in Figure 1 remain unchanged, where (a) is the existing OMA DM v1.x server and (b) represents a fleet of existing mobile devices embedding OMA DM v1.x client. The OMA DM adapter proxy (d) enables the OMA DM server (a) to manage new M2M constrained devices and gateways (c). Lightweight ELFOMA messages, as detailed in the previous section, are exchanged between the entities (c) and (d).

The adapter proxy functional diagram is shown in Figure 14. It can support more than one OMA DM server. The settings pertaining to each OMA DM server can be persistently stored in the proxy as Management Objects (MOs) tree, e.g. Figure 15, and thus manageable by OMA DM servers over the OMA DM protocol. An OMA DM server may add, update or delete new tokens or aliases into its dedicated MOs tree (e.g. OMA-DM-server1). Access to the proxy configuration MOs is controlled by the credentials. As such, an OMA DM server does not have access to configuration of another server.

Once the proxy is configured, it is ready to serve M2M devices. However, prior being able to connect to the proxy, M2M devices need the following information: APN configuration, proxys URL, associated OMA DM server alias... Such information is sent to managed devices via SMS by OMA DM Server. This bootstrapping procedure is defined by OMA [27].

As shown in Figure 14, ELFOMA client is embedded on constrained devices and gateways (c). To maintain full compatibility with the OMA DM v1.x servers, ELFOMA client must implement the OMA DM protocol stack. However, SyncML is not used in the message representation layer. As XML parser is no longer needed, the following savings can be achieved in the device: (i) device memory can be saved as the XML parser code/library footprint is quite large (depending on the target platform and development language, the library could reach over 4Mb in size [26]), (ii) reduction of the CPU and memory resource consumption, refer to results in section 6 (iii) preserve battery life as the transmission time is shortened due to the payload reduction, (iv) tree representation of the data model is no longer required, access to local variable is simplified. Low-resolution
variables (tokens) can be used to represent the pre-stored DevInfo MO (52 bytes), DevDetail MO (62 bytes) and other MOs. Message translation is not required in the client; it is handled by the proxy. The client device creates and processes directly lightweight ELFOMA messages as depicted in section 4.

Upon bootstrapping, the device has necessary information to connect to the associated adapter proxy. The device initiates a DM session by sending a request to the adapter proxy. The message translation is performed by the proxy as follow. In step 1 of Figure 14, the incoming ELFOMA message is parsed conforming to the ELFOMA content model defined in section IV; for each protocol element (e.g. SyncHdr, Status, Replace, Get, Add, Exec) found in the message, the proxy creates an instance of the corresponding software object class. In case of an error (bad syntax, missing SyncHdr), an immediate response with appropriate error code is returned to the device to terminate the session. Upon successful parsing in step 1, the incoming ELFOMA message is then transformed onto a list of protocol element objects in step 2. Each protocol object encapsulates a set of child elements objects, as defined by the associated content model. For example, the instance of a Replace protocol command object class contains CmdID, NoResp, Cred, Meta and Item child object instances. These objects are then converted onto SyncML object representation using SyncML content model [25]. The Target element is used to look up the proper recipient OMA DM server. All ELFOMA tokens are also mapped onto the corresponding high-resolution identifiers, as previously configured in the placeholder MO tree of the recipient OMA DM server. OMA DM server uses High-resolution identifiers to reference the physical data schema in database.

Finally, the serializeOMA function of each protocol elements objects (listed in step 2) is sequentially invoked in order to build the OMA DM SyncML payload. This newly converted payload is then sent to the recipient OMA DM server as a request.

On the downstream flow, the proxy handles the SyncML to ELFOMA translation as follow. Upon receiving the response from an OMA DM server, the proxy first validates the received payload in step 4. Once the conformance to the SyncML DTD [25] is validated, the proxy releases all resources allocated in steps 1 to 3. For each protocol element [25] (e.g. SyncHdr, Status, Replace, Get, Add, Exec) found in the OMA DM message, the proxy creates an instance of the corresponding software object class. Consequently, a list of protocol element objects is obtained in step 5; these objects are then converted to ELFOMA representation objects, defined in section 4. Known attributes of child elements, within all protocol element objects, are mapped to the corresponding tokens to further reduce signaling in step 6. The Source child element of the SyncHdr protocol element defines which configuration should be used for the token mapping.
Finally, the serializeELFOMA function of each protocol elements objects (listed in step 5) is sequentially invoked in order to generate ELFOMA payload (refer to section IV). This newly converted payload is then returned to the device as a response to the initial request, processed in step 1. All resources allocated in steps 4 to 6 are then released.

Complexity can be reduced using polymorphism in object oriented design, steps 1-3 and steps 4-6 can share the same code, except for creating the list of protocol element objects.

As described above, the adapter proxy main function is limited to a low complexity message translation. Unlike the OMA DM server, the proxy does not implement the following resource demanding operations: (i) device management logic defined by device management authorities and driven by various overlying enablers e.g. FUMO, DiagMO, SCOMO, etc, (ii) Execution of device management commands. (iii) device data persistency in database, such as searching, fetching and updating device attributes in large and distributed database. Therefore the proxy has a lower workload than OMA DM server. It is suggested to deploy the proxy onto a cloud infrastructure, the number of proxy instances is automatically scaled up or down to maintain the system availability and latency depending on the number of end devices connecting concurrently to the proxy.

A distributed system approach is considered in section 7 to address more complex scenarios or to enhance the scalability. An implementation is also suggested.

### 6. RESULTS

The proposed ELFOMA method has been evaluated and the following performance indicators have been measured: payload reduction, processing time and memory consumption. A smart grid use case implementing an energy consumption controller scenario is used to perform the assessment. This scenario can be used by an electricity supplier to avoid importing electricity from abroad or starting a fossil fuelled thermal power plant in case of energy consumption peaks that may occur in cold winter, particularly between 6PM to 10PM timeframe. This scenario consists in monitoring the global energy consumption. When it exceeds a predefined threshold, the system will be sending power cut orders to selected heaters. Cutting heaters for 10 or 15 minutes does not affect households comfort level. Heaters could be powered off based on a round robin basis so that households are evenly affected in order to prevent the consumption peak to happen. An alternative option is to lower the heating temperature instead of turning the heaters off. The corresponding message flow is depicted in Figure 16. OMA DM v1.2 payloads are firstly generated for the 8 messages of this scenario. They are converted to ELFOMA messages using directives listed in 4.1 and 4.2.

Respective payload sizes are measured and plotted in Figure 17. ELFOMA is also benchmarked against the XML compression methods mentioned in section 2.3, such as EXI and WBXML. OMA DM payloads are encoded with EXIfficient [28], an open source implementation of the W3C Efficient XML Interchange format specification, and with kXML2 [29]. GZIP is also included in the benchmark. Figure 17 depicts payload sizes of 8 messages used in the evaluation scenario. The compactness of different encoding schemes is shown in Figure 18. The baseline for these payload reduction ratios is the OMA DM payload size. The average reduction performances are shown in Figure 19.

The overall ELFOMA payload size (message 1 through 8) represents only 11.2% (810 bytes) of the cumulated SyncML payload weight (7246 bytes). In other words, ELFOMA payloads are 9 times smaller than the OMA DM payloads. Device battery consumption is significantly reduced as the transmission time over a radio access network (2G/3G) is shortened by a factor of 9. This achievement has also a positive impact on the network scalability; the number of M2M devices can be increased by a magnitude of 9.

Furthermore, ELFOMA over-performs EXI, WBXML and GZIP: the average compactness of the EXI schema-less encoding is 35.8%, EXI schema-informed mode achieves 18.5% compactness. WBXML yields less compact payloads, representing 68.4% of the SyncML size. GZIP reaches 54.5% average compactness. To enable the EXI schema-informed mode, the SyncML DTD, as defined in SyncML Repro [25], has been converted to XML schema definition (XSD). In practice, constrained devices and the adapter proxy have to share the same XSD for decoding and encoding. The XSD is large, 14394 bytes and it occupies more space in memory than the actual payloads. Findings on EXI are conform to the compactness results as stated in the EXI evaluation [30].

In addition to the payload reduction, the evaluation also measures processing performance and memory consumption with respect to parsing ELFOMA versus
OMA DM payloads. The respective 8 message payloads are thus parsed using java-based testbeds.

The average time required to parse ELFOMA payloads is 44.75ms, while 148.5ms are required to parse the counterpart OMA DM payloads, Figure 21. Therefore, parsing ELFOMA payloads is 3.3 times faster and less CPU demanding (Figure 20), thus less battery consuming, than parsing OMA-DM payloads. Moreover, parsing ELFOMA payloads is 22.7 times less memory demanding than parsing OMA DM payloads, Figure 22. ELFOMA parser uses 55KB (average over 8 messages) while OMA DM parser is making use of 1247KB. The amount of memory may appear to be very high. VisualVM tool (part of Java Development Kit) has been used to measure memory usage of the testbed within the Java Virtual Machine. It should be noted that the amount of memory being measured actually includes java runtime, libraries, code footprint, instances and memory allocation to process the said operations. The java-based implementation used in the testbed may not be leveraged in actual devices, which would rather use lightweight development framework suited for microcontrollers. However, the finding on the ratio would remain valid in term of magnitude. Regardless the type of embedded system and programming language, the device has to allocate memories to store the entire payload (9 times larger for OMA DM) and increasing working buffers as parsing complexity is getting higher. ELFOMA is memory efficient, hence helps to reduce the cost of the device.
7. CONSIDERING A DISTRIBUTED SYSTEM

The proposed solution can be extended and oriented to a distributed system in order to support a very large number of devices. The approach consists in deploying devices in a 2 or more levels hierarchical network structure. Gateways can manage directly end-devices and cluster heads, which in turn manage a cluster of end-devices. Device Groups can be defined; each group could contain list of cluster heads (transparency include child devices) or a list of devices (for fine granularity). In the aforementioned smart grid scenario, on behalf of the OMA DM Server, the proxy can issue one command (e.g. turn off heater, or set heater to low) to a gateway, which will forward it to all devices belonging to the targeted device Group. The command is forwarded to cluster heads, which in turn will relay it to subsequent child devices. The gateway then collects all responses from low-level devices prior sending the aggregated responses at once back to the proxy and ultimately to OMA DM server. The scalability is significantly enhanced, as the number of server transaction is reduced to one per group, regardless the number of devices in the group.

Standardized OMA GwMO enabler [7] can be used to implement this distributed approach as ELFOMA can leverage all existing OMA MOs. GwMO enabler has defined several MOs residing in the management tree of the gateway and cluster head (extended). A brief description is given below, for further information please refer to [7]. DeviceInventoryMO maintains a list of devices in the sensor network that are managed by the gateway. ConfigMO is used to configure groups of devices, alert, bootstrap, etc. The FanoutMO maintains information regarding the handling of DM command fan-out and response aggregation. The gateway leverages these MOs to keep track of managed devices, to identify targeted devices, to relay server commands to devices and to aggregate device responses.

8. POSSIBLE CONTRIBUTIONS

The proposed Lightweight method could be applied to other Lightweight related M2M standardization activities. The tree-based to flat data model mapping principle could help to leverage the existing OMA MOs in the Lightweight M2M (2.1) and the DM Next Generation (2.2) Work Items. This method combined with the envisaged protocol simplifications in the DM Next Generation (DM NG) will further reduce payloads. The adapter Proxy could be avoided if the derived methods are implemented within the new OMA standards. Although OMA DM v1.x protocol is a stateful, namely, and connection oriented, the ELFOMA concept can be applied to the connectionless oriented protocols, such as CoAP [11] which is considered by the DM NG.

9. CONCLUSION

Future OMA DM standards, namely OMA Lightweight M2M and OMA DM Next Generation, are not backward compatible with the OMA DM v1.x. On the other hand, ELFOMA (Exalted Lightweight For OMA DM) provides an alternative lightweight device management solution to operators who wish to reuse their existing OMA DM v1.x servers to continue managing OMA DM v1.x mobile devices and to incrementally manage new M2M constrained devices. ELFOMA converts SyncML messages into lightweight messages that are 9 times smaller. It over-performs EXI and WBXML. Moreover, ELFOMA messages are plain text, hence they can be further compressed using GZIP (e.g. HTTP compression, with Content-Encoding: gzip) to further compress them.

In addition to the size efficiency, ELFOMA is also data access efficient as it enables constrained devices to use flat data model, while it is mapped to a tree-based data model in the back-end OMA DM server. The efficient encoding scheme is based on a CSV representation using well-defined and straightforward Content Models as a pattern. Parsing ELFOMA payloads is 3.3 times faster, less CPU demanding and 22.7 times less memory demanding than parsing OMA DM payloads. The following key benefits can obviously be derived from the measured performances:

(i) battery life of constrained devices is preserved as the payload transmission time is reduced by a factor of 9 and parsing ELFOMA is less CPU demanding with low processing time (ii) better utilization of the network bandwidth as the number of devices can be increased by a factor of 9, (iii) the cost of devices can be reduced as hardware requirements are lowered (lower memory capacity).

Finally, the proposed adapter proxy has low complexity and workload. The solution can support multiple OMA DM servers and be can be extended to a distributed system.

ACKNOWLEDGEMENTS

This paper describes work undertaken in the context of the EXALTED project - Expanding LTE for Devices (http://www.ict-exalted.eu). The research leading to these results has received funding from the European Community’s Seventh Framework Programme under grant agreement n ICT-5-258512.

REFERENCES


7. OMA-DM Gateway Management Objects. GwMO. http://www.openmobilealliance.org/Technical/release_program/GwMO_V1_0.aspx
22. Toms Sánchez Lpez, Alexandra Brintrup,Marc-Andr Isenberg, Jeanette Mansfeld , emphResource Management in the Internet of Things: Clustering, Synchronisation and Software Agents, Architecting the Internet of Things 2011, pp 159-193
AUTHORS’ BIOGRAPHIES

Nhon Chu graduated from Institut National des Télécommunications, in France. He held different positions for the last 10 years: software engineer, architect, software team leader, project manager. Nhon works on various M2M projects at Sierra Wireless and serves as device management task leader in FP7 ICT-Exalted project.

Djelal Raouf received Eng. Degree in 2000 from the Ecole Nationale Supérieure de Techniques Avancées (ENSTA). He began to work as an embedded system consultant at Altran Technology. In 2005, he joined Sagemcom Energy & Telecom where he worked for the GSM-R activity and then for the M2M activity. In particular, he has been in charge of the FP7 ICT-EXALTED project management. Since August 2012 and the acquisition of the M2M activity of Sagemcom Energy & Telecom by Sierra Wireless, he works as a software architect and cellular network expert.

Bruno Corlay graduated in Physics in Université de Bretagne Occidentale as well as in digital signal processing in Université de Valenciennes. Now working in Sierra Wireless, he has participated in various telecommunication projects such as GSMR, Wimax and, lately, in the Sierra Wireless M2M research and development department with a notable involvement in a European Commission funded project named Exalted (“Expanding LTE for devices”) and in software architecture for embedded systems.

Mohamed Ammari Graduated in microelectronics and automation (Eng. & Master) in 1999 from Polytech’ Montpellier. His early works were on laser telemetry for nuclear plant visual inspection before he join the mobile phone R&D hardware engineer for several companies (Philips, Sagem ). Since 2003 has been working in M2M and he is in charge of the hardware of the main M2M module projects in Sagemcom then Sierra wireless and for the Exalted European project.

Nenad Gligoric is a Ph.D. candidate at the Faculty of Organizational Sciences, University of Belgrade in Serbia and is advised by Srdjan Krco. His research interests are M2M, Internet of Things and Pervasive computing. As a research consultant in Ericsson, he is engaged for FP7 projects (HOBNET, EXALTED and SmartSantander). His work has been published in academic venues including PerCom, DCOSS and CyberC.

Dr Srdjan Krco is with Ericsson since 2000 where he held a number of positions (senior research engineer, system manager, project manager) and has worked in and managed a number of product development and research projects. For the last 5 years Srdjan leads a research and innovation team focusing on M2M, Internet of Things and their application in various domains including the smart cities. He has participated in several FP7 projects (SENSEI, SmartSantander, EXALTED, IOT-i) and is active in the Future Internet Assembly. Currently, Srdjan is chairing Technology working group of the IoT Forum. In 2007 he received the Innovation engineer of the Year Award in Ireland from the Institute of Engineers of Ireland.

Nemanja Ognjanovic was born in Belgrade, Serbia. He received B.Sc. degree and M.Sc. degree in Telecommunications from the Faculty of Electrical Engineering, Belgrade, in 2005 and 2011, respectively. Since then, he has been with Telekom Srbija, Belgrade, the incumbent service provider in Serbia, where he is currently the Head of Section for Development and Evaluation of Technologies.

Aleksandar Obradovic received his M.Sc. degree in telecommunication engineering from University of Belgrade, Serbia in 2008. He joined Telekom Srbija a.d., Belgrade, Serbia, in 2009 where he started in mobile PS (Packet Switch) core department. He currently works in Mobile Network Service Architecture Department. He is involved in projects like E2E QoS traffic management, mobile multimedia, WiFi offload, FP7 projects, etc. His main areas of research interest are VAS and IP based service platforms.