European Commission
Information Society and Media

SEVENTH FRAMEWORK PROGRAMME

Large Scale Integrating Project

EXALTED

Expanding LTE for Devices

FP7 Contract Number: 258512

WP4 – End-to-End (E2E) M2M System

D4.3
Device Management

Contractual Date of Delivery to the CEC:	31 October 2012
Actual Date of Delivery to the CEC:	31 October 2012
Responsible Beneficiary:	SWIR
Contributing Beneficiaries:	SWIR, GTO, UNIS, EYU, TKS, CTTC
Estimated Person Months:	45
Security:	Public
Nature	Report
Version:	1.0

Document Information

Document ID:	EXALTED_WP4_D4.3.doc
Version Date:	31 October 2012
Total Number of Pages:	149
Abstract	EXALTED Device Management Solutions
Keywords	Device Management, OMA-DM, Lightweight

Authors

Name	Organisation	Email
Nhon CHU (Editor)	Sierra Wireless	nchu@sierrawireless.com
Serdar Vural	University of Surrey, UK	s.vural@surrey.ac.uk
Jerome d'Annoville	Gemalto	jerome.d-annoville@gemalto.com
Nenad Gligoric	Ericsson	nenad.gligoric@ericsson.com

Approvals

	Name	Organisation	Date	Visa
Internal Reviewer 1	Bojana Jakovljevic	TKS	31/10/2012	OK
Internal Reviewer 2	Pethros Bithas	UPRC	31/10/2012	OK
Technical Manager	Pirabakaran Navaratnam	UNIS	31/10/2012	OK
Project Manager	Djelal Raouf	SWIR	31/10/2012	OK

Executive Summary

This report presents EXALTED Lightweight Device Management (DM) solutions for M2M applications that require the transmission of small volumes of data by a large number of power-constraint and low complexity M2M devices.

Scalable solutions
Two EXALTED Lightweight DM approaches are proposed along with security considerations. ELFOMA (EXALTED Lightweight device management For OMA) solution is spectrum efficient as the size of DM messages is reduced by 85% compared to OMA-DM v1.x messages. A proxy adapter is introduced to perform a 2-way message conversion, namely, to convert OMA-DM v1.x packages to ELFOMA packages and vice versa. This solution is compliant with existing OMA-DM servers which remain unchanged. Text based or ASN.1 encoded messages can be exchanged in ELFOMA.
Stakeholders are not tied with existing OMA-DM servers and can use the CoAP based solution which is also spectrum efficient. Messages are exchanged using CoAP over UDP to update management objects. The update is utilized by sending compressed executable code modules over UDP, which enables updates on all system levels for simple sensor devices and reduces energy consumption and dissemination time. A study on CoAP over SMS is also disclosed in this document.
In the proposed EXALTED Lightweight DM, depending on the reactivity required by the M2M application, non-urgent upstream data sent by end devices towards M2M applications is temporarily stored at the gateway. Based on a configurable data collection strategy, the gateway then forwards the aggregated data to the application. Therefore, this approach avoids having a large number of concurrent connections over the radio access network, hence addresses the scalability issues. Furthermore, the EXALTED Lightweight DM supports multicasting and broadcasting capabilities. This reduces the number of required concurrent connections which are established by the DM server to send downstream DM commands to end devices deployed behind the gateway. Upon receiving DM data from the DM server, the gateway multicasts DM commands to all end devices belonging to the targeted group, or broadcast commands to end devices pertaining to the targeted top level of sub-tree hierarchy. DM commands are relayed by subsequent levels of Cluster Heads (CHs) until delivery to the intended recipient end devices is complete. Gateway and CHs may have multiple air interfaces, such as ZigBee, Bluetooth, or KNX; the appropriate air interface is used to communicate with end devices.
Other aspects of scalability are also considered in this report, such as infrastructure, architecture and automation. Since M2M devices are unattended, it is critical to have automation processes to monitor the devices so as to increase reliability and autonomy, which provides an additional lever to further scales up system performance. Having distributed service logic rather than a centralized one will further automate the system and offload M2M applications. This concept is enabled by providing scripting capability to devices in order to execute service rules, and by enabling device-to-device messaging.

Energy efficient solutions
As end devices are mostly battery operated, energy efficiency is another essential objective for M2M applications. DM solutions presented in this report address this issue in different ways. First, the aforementioned DM payload reduction has a significant contribution in reducing the energy consumption of devices, as the transmission time is shortened. Second, an energy-efficient capillary network structuring solution is also required. As such, end devices collaborate over the Distributed Clustering Protocol (DisC Protocol) to elect CHs, which is also presented in this report. When the battery level gets low in CHs, new ones are automatically elected. CHs have essential routing and data aggregation roles within the capillary network, hence they are more energy demanding. By preserving the battery of CHs this approach also increases the reliability of the capillary network. Third, DM sessions can

be initiated by DM Server using SMS, which enables devices to have a longer idle cycle, hence yielding further energy savings. Since DM commands that are generated by the DM server are propagated from gateway to CHs, they can be retained at these nodes until they are delivered to intended recipient end devices. The Time-to- Live (TTL) parameter, which determines how long the commands are retained, can be adjusted to the duty cycle of end devices.

Cost effective solutions
EXALTED Lightweight DM also contributes to reduce the cost of devices through the simplifications and low resource features it provides. For instance, ELFOMA solution enables operators to reuse existing OMA-DM v1.x servers to incrementally manage new constrained M2M devices. The capital expenditure is limited to ELFOMA proxy adapter which converts OMA-DM v1.x SyncML based messages (XML) to ELFOMA's Comma Separated Value (CSV) messages. Parsing CSV message is much less complex than parsing XML messages. Furthermore, an ELFOMA message is six times smaller than an OMA-DM message. EXALTED Lightweight DM is used as underlying protocol to support other service capabilities, such as data collection, device-to-device messaging and device addressing. Using this approach, an M2M device does not have to support additional protocols for different services, e.g., File Transfer Protocol (FTP) for data collection/download and MQTT protocol for messaging. Posting sensor data to M2M applications, downloading device firmware, and sending and receiving messages are performed using the same underlying EXALTED Lightweight DM protocol. This concept not only is low resource demanding, but also simplifies device design and maintenance. For example, there is no need to upgrade multiple protocols and, multiplexing, synchronizing or managing protocols concurrent access to the common air interface is not required.

Secure system
EXALTED Lightweight DM solutions are based on existing DM standards (OMA-DM and CoAP), thus they inherit proven security mechanisms such as, security at the transport level, mutual authentication, challenge, replay attack, and integrity check, etc. Besides the standardized security mechanisms, end-to-end security enables end device to encrypt sensor data prior to conveying to the associated M2M application. The encrypted data may be routed and relayed by different entities before reaching the intended recipient. These entities participating to the message relaying, i.e. devices, CHs, gateway, or M2M server, do not have the proper key to decrypt the encrypted sensor data. Only the intended recipient has the right key to access the data. Binary encrypted data can be base64 encoded and encapsulated in ELFOMA or CoAP messages. Group keys are also defined for multicast and broadcast capabilities.

Open architecture
Besides addressing the business and technical objectives, a DM architectural concept is also unveiled in this report along with its components and interfaces. With the enabler approach, new DM functions and new service capabilities can be added to the system. DM architecture can be mapped onto the EXALTED system architecture [2]. As a result, a high level mapping of algorithms developed by EXALTED's other Work Packages onto the DM architecture is provided. This gives an idea on how EXALTED Lightweight DM can leverage multiple EXALTED algorithms to collaborate and to achieve EXALTED's objectives. This DM architecture is being implemented in a DM testbed [10] along with ELFOMA to showcase various DM functionalities and service capabilities. This testbed will also be used to numerically assess some of EXALTED's Key Performance Indicators (KPIs), i.e. signalling reduction, transmission payload size (K30), actual payload size (K32). A preliminary list of KPIs can be found in deliverable D7.2 [10].
The DM components and interfaces presented in this report can be mapped onto the ETSI M2M architecture. The EXALTED Lightweight DM Server is the counterpart of ETSI's

Network REmote Management entity (NREM). The data retention server can be mapped to ETSI's NDHDR (Network Data History and Data Retention). DM functions can be implemented into ETSI's Management Function entity. In the first release, ETSI M2M recommends OMA-DM v1.x and TR-069 protocols for device management in REM service capabilities. Furthermore, an implementation option of ELFOMA onto OpenMTC platform is described in this report. OpenMTC is a cooperative development of Fraunhofer FOKUS and Technische Universität Berlin (TUB). Its commercial M2M platform is based on the ETSI M2M Architecture.

Achieving EXALTED objectives

In conclusion, the novel DM solutions provided in this report, address EXALTED's five high level objectives: (i) scalability, (ii) energy efficiency, (iii) cost efficiency, (iv) reliability, and (v) security. The proposed DM solutions can be mapped onto the ETSI M2M architecture and the EXALTED system architecture. They can also be combined with other algorithms developed by other EXALTED Work Packages to further achieve project objectives by combining the benefits yielded at different levels of the EXALTED system.

Table of Contents

1. Introduction

Business needs
EXALTED aims to develop a new system to support the anticipated massive number of M2M devices [11][12][13] utilized in M2M applications. These applications are mostly cost constrained, mainly due to the large number of devices that needs to be deployed in large areas. As a consequence of this requirement for low-cost devices, a M2M device must have the following key characteristics: It must be (i) small size with limited capability and low design complexity in order to reduce material costs, (ii) wireless, hence mostly battery operated, in order to reduce installation costs, and (iii) unattended devices in order to reduce operational costs. To save energy, M2M devices mostly optimize duty cycle, exchange small volumes of data, and communicate with neighbour devices using energy efficient wireless communication technologies. Sending technicians to the deployment field to physically operate on devices is not suitable for the aforementioned cost constraints. Hence, Device Management (DM) is a must have capability in order to extend the lifetime of M2M devices.

Challenges
The key challenge is to support a huge number of M2M devices concurrently exchanging small amount of data with M2M applications over the radio access network. LTE is appropriate to support mobile devices exchanging video, email or any other user oriented data. However, LTE is not efficient for M2M Applications as the signalling overhead is likely larger than the actual size of M2M data. In addition, the increase in number of M2M devices is expected to exceed mobile device growth figures. EXALTED has specified an extension of the existing 3GPP LTE, referred as LTE-M system [3], to satisfy this need.
From the end-to-end point of view, DM must also address this challenge while providing the ability to manage four different types of M2M devices and different communication scenarios that have been defined to achieve ubiquitous connectivity and coverage in large areas.
This report discloses DM solutions that satisfy the following EXALTED high level objectives whose scopes are described in the Executive Summary: (i) Scalability, (ii) Energy efficiency, (iii) Cost efficiency, (iv) Reliability, (v) Security.

Considering different needs
Section 2. details key considerations that have been taken into account in order to determine the required DM functionalities and related services. EXALTED scenarios as envisioned in deliverable D2.1 [1] have been analysed to define high level DM needs. These specific DM needs are then consolidated with high level needs coming from other EXALTED tasks. As a result of this consolidation, a list of technical requirements have been defined and listed in deliverable D2.1 [1]. It should be noted that a technical requirement may be addressed by one or more solutions defined by one or multiple EXALTED tasks. The required DM functionalities and related services are identified, they are necessary to meet a set of technical requirements. EXALTED system architecture [2] is then considered. In addition to the definition of entities in network and device domains, further technical objectives can be derived from the system architecture, particularly, the need to: (i) manage 4 different types of M2M devices (M2M Gateway, LTE-M enabled devices, Non-LTE-M cluster heads (CH) and Non-LTE-M end devices), (ii) structure Non-LTE-M devices within a multi-levels hierarchical heterogeneous capillary network, (iii) handle different types of communication scenarios.

Defining DM architecture and solutions
DM functions and related services capabilities identified in section 2. are presented in section 3. DM architecture is revealed along with key components and interfaces realizing the identified DM functions or services. The proposed DM architecture can be mapped onto EXALTED system architecture. As such, components developed by other EXALTED tasks can be integrated and leveraged to further achieve EXALTED objectives. General aspects of

security are considered. Different scalability levers are disclosed to address challenges in managing huge number of M2M devices.

Sections 4. to 8. , detail key enabling solutions, components and interfaces as described in section 3. in order to fulfil technical requirements and technical objectives as listed in section 2. Solutions design has been guided by EXALTED high level objectives as listed in the "challenge" paragraph:

Adopting and Extending Standards to build DM solutions
State of art DM protocols are described in section 4. to identify which existing DM solutions or protocols can be used in EXALTED. Two DM solutions leveraging existing standards are proposed and disclosed in this report. This report provides an answer to the key question, how the DM solution manages M2M devices organized in a multi-level hierarchical structure in heterogeneous capillary networks? The first solution is based on the Constrained Application Protocol (CoAP) [33], while the other is based on OMA-DM v1.x [35]. Why based on this latter specification which is designed for managing mobile devices and OMA-DM is specifying a new lightweight protocol [38] dedicated to M2M devices? The reason for this need is threefold: (i) the new OMA-DM lightweight M2M protocol is not available yet, (ii) it will not be backward compatible with the current V1.x specifications, (iii) for cost saving reasons, operators and service providers may wish to use their existing and proven OMA-DM v1.x servers to continue managing existing types of mobile devices and to incrementally support newly emerging M2M devices. This last point is the main motivation for proposing an OMA-DM v1.x compliant lightweight DM solution, referred as ELFOMA (EXALTED Lightweight adapted From OMA-DM). Besides this cost driven advantage, this solution can leverage all existing OMA-DM Management Object (MO) enablers, thus inheriting all proven and reliable DM functions, procedures and security. As such, the OMA Gateway MO enabler is used in this solution to manage devices behind the gateway. However, OMA GwMO enabler does not specify how to manage devices organized in a multi-level hierarchical structure in heterogeneous capillary networks. For instance, for a 2-level hierarchical structure, devices are located behind a Cluster Head (CH), which is managed by the gateway. This aspect is addressed in this report.
The added values over the standard DM protocol are signalling reduction and low complexity payload encoding. These approaches contribute directly to enhance the scalability, the cost efficiency and the energy efficiency of the system. The CoAP [33] based DM solution is a scalable connectionless oriented protocol. Managing devices over SMS is also considered.

Addressing M2M Devices
Section 5. describes the device naming and addressing approach used in EXALTED Lightweight DM solutions. Addressing related to LTE-M devices and Non-LTE-M devices is considered. DM Server exchanges DM messages with M2M Gateway or LTE-M enabled devices over IPv4 or IPv6. ELFOMA uses device unique identifier to address all M2M devices. The number of addressable devices is therefore not limited. Device addressing mechanism is thus scalable.

Implementing DM Functions and Service Capabilities
Having detailed EXALTED Lightweight DM protocols in section 4. and how DM messages can be addressed to M2M devices in section 5. , section 6. reveals how DM functions can be actually implemented. Self-diagnostic and self-healing components disclosed in deliverable D6.3 [9] can be supported by the proposed DM protocol, making devices more reliable. Service capabilities can be supported using the underlying EXALTED Lightweight DM protocol.

Structuring and Managing Devices in Capillary Networks
As principle and methods to manage LTE-M devices are provided in the previous section, management of Non-LTE-M devices behind gateway is detailed in section 8. Capillary network structuring algorithms are disclosed in section 7. Forming and structuring capillary network is a prerequisite prior DM can be initiated. Multicast, broadcast and mobility management solutions are provided in section 8. along with scenarios and security considerations.

Conclusion
The conclusion summarizes added values brought to EXALTED and to the state of the art. Preliminary information on performance evaluation is included.

Implementation option onto ETSI M2M Architecture
The annex shows how ELFOMA can be mapped onto OpenMTC platform which is an actual implementation of ETSI M2M Architecture.

2. General Considerations

This section presents different aspects that have been taken into consideration in order to design device management solutions which will be described in section 3.

2.1 Envisioned scenarios

The main envisioned scenarios in EXALTED are specified in Deliverable 2.1 [1], and they can be divided into three main use cases:

1) Intelligent Transport System (ITS)
    a. Remote Monitoring of Vehicle Data
    b. In-Vehicle M2M Diagnosis
    c. Railway Remote Monitoring and Failure Detection
    d. Parking Time Check
    e. Vehicle Collision Management
    f. Gateway vehicle for car-to-car communications
2) Smart Metering and Monitoring (SMM)
    a. Energy Smart Metering – Building Management
    b. Industrial Monitoring
    c. Environmental monitoring
    d. Security – Surveillance
3) E-Health Scenario

DM is required in all above scenarios. Key features such as device configuration, device monitoring, remote diagnostic, firmware update, etc… are needed. In addition, in most cases, M2M devices are managed by a M2M Gateway which is connected over the LTE-M network and the internet to the application server. Safety related scenario presents time-critical constraints, while some other applications dedicated to monitoring might be time-tolerant. It is therefore important that device management related payloads size shall be minimized to reduce transmission time. For some cases, it is worth noting that deployed M2M devices are ultra-small: very low power and very constrained in terms of memory and CPU. To this end, in addition to payload reduction, low complexity payload encoding should be considered.

In addition to device management functions, other service oriented supporting functions such as posting data to application server and downloading data from distant server are required in the M2M devices and M2M gateway. To fulfil these underlying functions, M2M devices have to embed additional particular protocol (e.g. FTP or other application specific protocol). Due to the aforementioned footprint constrains of M2M devices, it is likely to merge management protocol and service oriented protocol. This aspect will be considered in EXALTED.

2.2 Requirements

Table 2-1: lists key DM needs derived from envisaged scenarios as described in section §2.1 and sorted by categories according to their nature. Not all needs are listed; the list is limited to key needs.

Categories	Needs
Managing Capillary Devices	Non-LTE-M devices are deployed behind M2M Gateway. Non direct communication between M2M Server and Non-LTE-M devices
	M2M Server must manage M2M Gateway, which in turn, manages Non-LTE-M devices
	Non-LTE-M devices are constrained devices (limited resources: memory, processing, battery, etc.
	Non-LTE-M devices in capillary network must have self-organizing capabilities
	Non-LTE-M devices might be moving from one capillary network to another
	M2M Gateway needs to keep track of active Non-LTE-M devices being managed in order to fan out commands to them
	Capillary devices need to send sensor data to M2M server
	At power up, devices are automatically detected and attached to a Gateway which registers them onto a M2M Server
Addressing	M2M Server shall be able to generically address Non-LTE-M devices regardless of the type of capillary network
	Non-LTE-M devices may not have IP address
	M2M Server needs unicast and multicast method to address Non-LTE-M devices. Devices may be grouped together and can be addressed at once
Device Management functions	Provisioning is needed, it consists in bootstrapping all necessary information to Device or Gateway in order to contact M2M Server. For instance, Bootstrap data can be: APN info, credentials, URL of the M2M Server, etc.
	Massively deployed devices/gateways may not be easily accessible with reasonable costs. The ability to control the software update of devices/gateway remotely is required to efficiently manage the life cycle of the devices/gateways.
	In addition to firmware update, managing the life cycle of embedded software is also critical. Software component management functions are needed to install, remove or upgrade software in devices.
	M2M Server needs to send new settings to Devices/Gateways. For instance, periodicity of data posting, server URL for posting data, software update, diagnostic policies, etc.
	Diagnostic and monitoring are needed as they enable management authority to proactively detect and repair troubles even before services are impacted, or to determine the potential problems with a device. This contributes to ensure the reliability of devices.
Data Collection	LTE-M Devices need to post data directly to M2M Server whilst Gateway aggregates data collected from managed capillary devices before posting them at once to the M2M Server.
	Device data collected by M2M Server must be made available to M2M Application

Remote Control	M2M Server needs to trigger remote action on the selected device/gateway. For instance, to remotely switch on/off streetlights.
	High priority action should be triggered by the way of Device/Gateway Wake-Up mechanism
System Security	System Security is critical and is almost ubiquitous, all the above categories need to be secured
	Authentication and Authorization is needed for all above needs
	Data integrity check is needed for all message exchanges
	Data Confidentiality is needed for all message exchanges
High Availability High Scalability	The existing electricity/water/gas meters, cars, payment terminals, etc… will be transitioning to a "smart" version with full connectivity. Therefore Device Management System needs to support an extremely high numbers (billions) of connected LTE-M devices/gateways. This applies to all above categories
	As devices are unattended, automated process is needed to manage massive deployed devices
	Protocol Signalling reduction is needed
	Proposed solutions should be spectrum efficiency as eNodeB has to support increasing number of devices
Low-Energy Consumption and Low –Cost devices	Low cost devices are mostly constrained, with very limited CPU and memory capabilities and are mostly battery operated. Low complexity solutions are needed to reduce the costs of devices.
	Solutions minimizing the transmission time are needed as they help to save energy
	Solution minimizing the transmission frequency are needed as they help to save energy

Table 2-1: Needs to be addressed Device Management

Table 2-1: lists the high level needs to be addressed by device management related components. After some refinement, they are translated into technical requirements, as stated in EXALTED Deliverable 2.1 [1].

Table 2-2 provides a mapping between device management related technical requirements and approaches proposed in this document to address them.

ID	Title	Priority	Addressed by Functionalities
FU.1	Support of large number of devices	Mandatory	Lightweight Device Management protocol, 3.1.1 Multicast and Broadcast, 3.1.3.4
FU.3	Support of diverse M2M Services	Mandatory	Services Extension, 3.1.6
FU.4	Network initiated packet-data communication	Mandatory	Server initiated communication, 3.1.2
FU.5	Local and remote device management	High	Lightweight Device Management protocol, 3.1.1

FU.6	Unique identification of devices	High	Naming and Addressing, 5.
FU.7	Security and provisioning	Mandatory	Security solutions, 3.1.7
SV.3	Efficient provisioning of a set of M2M equipment	Mandatory	Security solutions, 3.1.7
SV.5	Delegation and distribution of functionalities	Mandatory	Architecture, Distributed DM, 3.2, 8.
SV.6	Security	Mandatory	Security solutions, 3.1.7
NT.1	Heterogeneous networks	Mandatory	Management of Non-LTE-M devices behind Gateway, 3.1.3
NT.6	End to end device to device communication	Mandatory	Device-to-device messaging, 3.1.5
NT.8	Mobility Management	Mandatory	Mobility management, 3.1.3.3
NT.9	Reliable delivery of a message	High	Lightweight Device Management protocols, 3.1.1
NT.12	Self-diagnostic and self-healing operation	Medium	Device Reliability, 3.1.4
NT.13	Multicast and broadcast communication	Mandatory	Multicast and Broadcast, 3.1.3.4
NF.1	Scalability	Mandatory	Lightweight Device Management protocol, 3.1.1 Multicast and Broadcast, 3.1.3.4
NF.2	Energy Efficiency	Mandatory	Lightweight Device Management protocol, 3.1.1 Services Extension, 3.1.6
NF.3	Extensibility and adaptability	Medium	Services Extension, 3.1.6
NF.6	Address space scalability	High	Addressing, 5.
DV.1	Self-organized M2M equipment	Mandatory	3.1.3.2
DV.2	Reliable M2M equipment	High	Device Reliability, 3.1.4
DV.6	Gateway detection and registration	Mandatory	Management of Non-LTE-M devices behind Gateway, 3.1.3, 8.4
DV.7	Protocol translation at the gateway	Mandatory	Management of Non-LTE-M devices behind Gateway, 3.1.3, 8.3
DV.8	Information routing at the Gateway	Mandatory	Management of Non-LTE-M devices behind Gateway, 3.1.3, 8.5
DV.9	M2M equipment wake-up	Mandatory	Server initiated communication, 3.1.2
DV.10	Remote configuration	Mandatory	Lightweight Device Management protocol, 3.1.1
DV.11	Software update over the air	Mandatory	Lightweight Device Management protocol, 3.1.1

Table 2-2: Technical requirements

2.3 EXALTED System Architecture

This section briefly introduces the system architecture as defined in Deliverable D2.3 [2]. Components of architecture are then described. For further information on the system architecture and LTE-M, please refer to [2][3].

Proposed Device Management solutions must take this system architecture into account. Particularly, how device management components will be mapped in the EXALTED system architecture. This mapping will be provided in the next chapter.

2.3.1 Overview

The EXALTED System Architecture as defined in Deliverable D2.3 [2] is depicted in Figure 2-1.

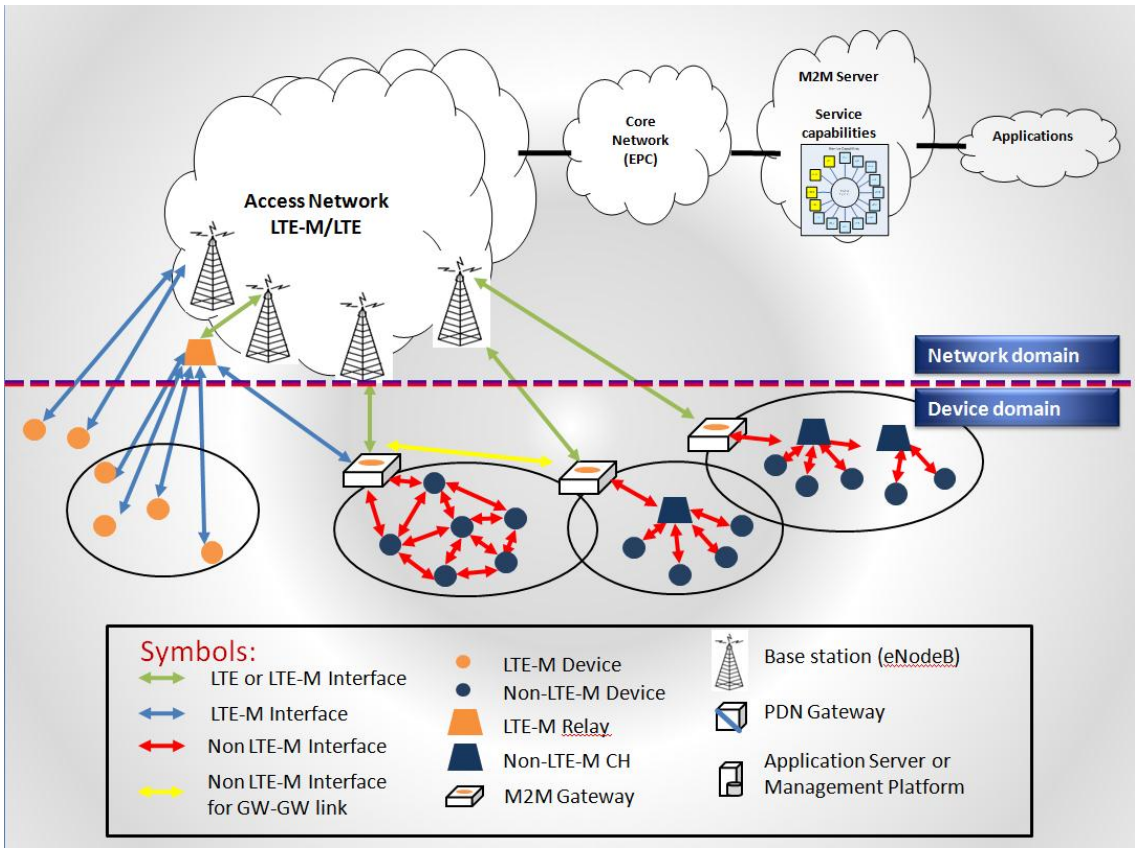


Figure 2-1: EXALTED System Architecture

The System Architecture comprises 2 main domains: the Network Domain (ND) and the M2M Device and Gateway Domain (DD).

Network Domain (ND): All components whose functionality is related with the control of applications, security and the management of devices belong to the ND. In EXALTED the wide area Access Network is restricted to the LTE-M/LTE system. Moreover, the EPC responsible for the management of cellular radio network and the eNB in the Evolved Universal Terrestrial Radio Access Network (E-UTRAN) are part of the ND. It is assumed that the application may run on a M2M server accessible from the Internet using the EPC. In the ND reside also the logical components, which are responsible for specific functions, such as the authorization and management of devices and network components.

M2M Device Domain (DD): The DD includes all kind of devices that support one or more applications. The link between DD and ND is the Uu interface defined in 3GPP. However, the used air interface is not LTE, but LTE-M, an autonomous radio access network coexisting with LTE in the same spectrum and specified in [3].

2.3.2 Entities in the ND

As depicted in Figure 2-1, the ND comprises 5 entities:

1. M2M Application. This logical entity runs the service and business logics of the application which are driven by the workflows of use cases. For instance, Smart metering application would collect meter indexes from households and generate billings process. This entity is out of EXALTED scope.
2. M2M Server. A logical entity that provides M2M functions that can be shared by different M2M Applications. Device management, control and service functionality are part of this entity. It uses the Evolved Packet Core (EPC) functionalities to communicate with Devices and Gateways.
3. Evolved Packet Core (EPC). The EPC consists of Packet Data Network Gateway (PDN-GW), Serving Gateway (SGW), Mobility Management Entity (MME), Home Subscriber Server (HSS), and Policy Control and Charging Rules Functions (PCRF). EXALTED does not intend to propose any changes in the EPC.
4. LTE-M eNB. This entity is connected to the EPC and provides LTE-M network connectivity to LTE-M relay, LTE-M enabled devices or LTE-M enabled gateways.
5. LTE-M relay. The main function of this entity consists in providing LTE-M coverage extension.

2.3.3 Entities in the DD

As depicted in Figure 2-1, the DD comprises 4 types of devices:

1. LTE-M device. This type of device has LTE-M interface and can access the Network domain, either by directly accessing the LTE-M network, or through an LTE-M Relay.
2. M2M Gateway. It provides the interconnection between the LTE-X (i.e. LTE/LTE-A/LTE-M) network and the capillary networks (consisting of one or more devices). It can provide various functionalities, such as protocol translation, routing, resource management, device management, data aggregation, etc.
3. Non-LTE-M enabled device. This type of device does not have an LTE-M interface. They form capillary network(s) using other network access technologies, such as Zigbee, and IEEE 802.11x. It can access the ND through a M2M Gateway, and run M2M applications locally.
4. Non-LTE-M Cluster Head. This type of device can be considered as capillary devices with some additional capabilities (e.g. traffic aggregation). Like regular Non-LTE-M Devices, they are also part of capillary networks and the communication from a regular Non-LTE-M Device may be directed through and managed by a CH.

2.3.4 Components of EXALTED Architecture

The mapping of functional components onto the aforementioned entities is depicted in Figure 2-2.

Note that the EPC Core Network is based on 3GPP MTC, the inner components of the EPC entity is also detailed in Figure 2-2. The relation with the 3GPP MTC approach and the backward compatibility with LTE are highlighted. EXALTED has adopted the indirect 3GPP

model, which assumes that the M2M Application does not connect directly to the operator network.

In the DD, all device entities have the following components:

- M2M Applications. Applications embedded in device, gateway or CH
- M2M Service Capabilities. Service oriented functions are encapsulated in this component, thus can be shared by the embedded M2M Applications.

In addition to above components, Gateways have an additional “Enhanced Gateway functionalities” component which exposes services that related to gateway.

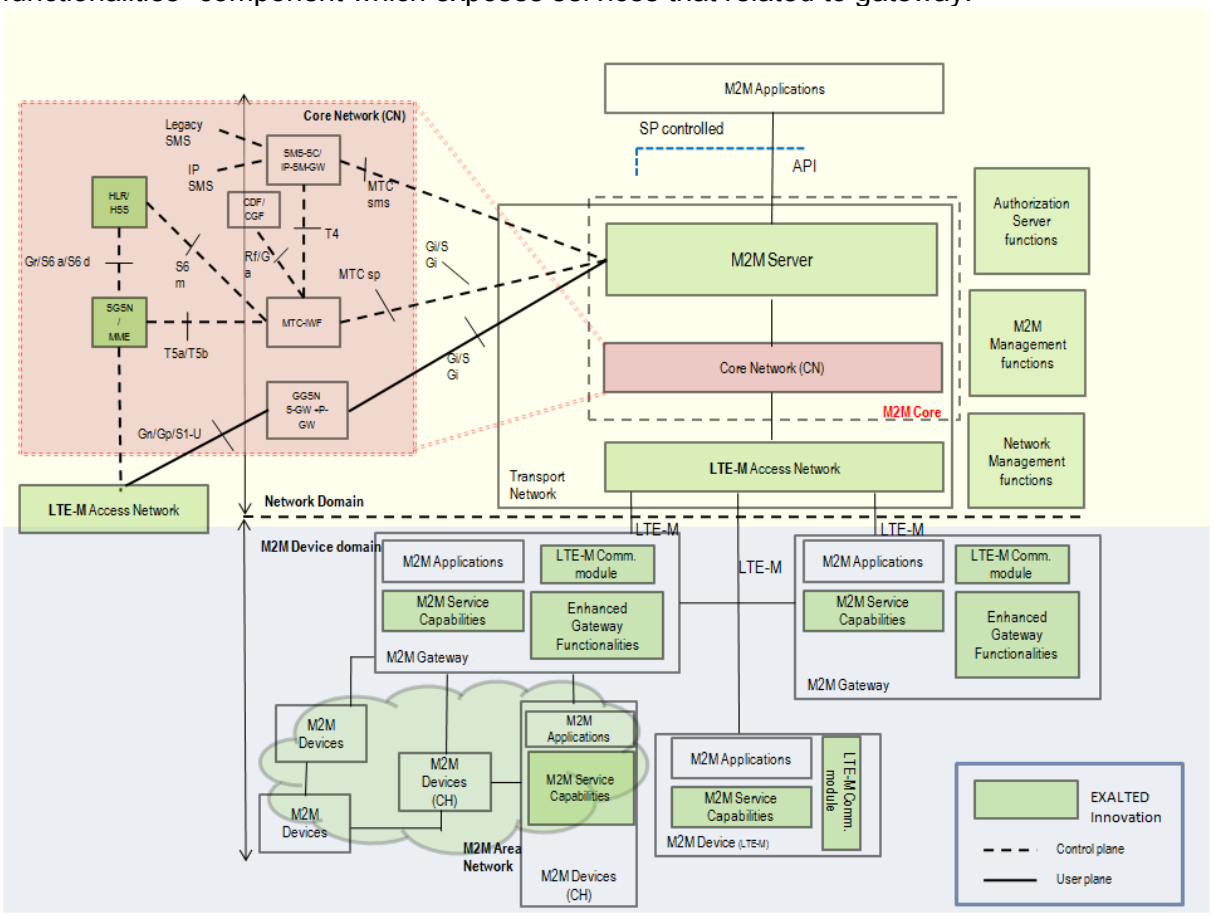


Figure 2-2: Components of the EXALTED architecture

3. Device Management Solutions

Based on considerations in section 2. , required device management related functionalities are listed in Table 2-2.

The architectural aspect is then considered in order to provide mapping of device management related components onto the EXALTED system architecture. Interfaces between components are also described.

Key algorithms pertaining to the identified components are detailed in separate and dedicated sections.

3.1 Solutions overview

Based on the main envisioned scenarios in EXALTED as specified in Deliverable 2.1 [1], Device Management related needs have been identified and categorized in Table 2-1. These needs were analysed in order to create a set of requirements. This latter was then consolidated with requirements coming from other Work Packages. The consolidated results are EXALTED Technical Requirements, presented in Deliverables 2.1 [1].
Each technical requirement could therefore be complementarily or jointly addressed by different Work Packages. The proposed solutions should also be aligned to EXALTED high level objectives: scalability, energy efficiency, low cost solutions, reliability and security. It is important to propose solutions that leverage or extend existing standards, in order to achieve interoperability and sustainability objectives.

The following device management related components have been developed to address technical requirements as listed in Table 2-2.

3.1.1 Lightweight Device Management protocols

Two DM protocols have been developed within EXALTED. These protocols aim to reduce the size of payloads which contributes to enhance the spectrum efficiency hence the scalability. As the transmission time is reduced due to payload reduction, the energy efficiency is enhanced. The low complexity of the solutions also contributes to reduce the cost of device.

3.1.1.1 OMA-DM v1.x complaint Lightweight Device Management (ELFOMA)

ELFOMA supports usual device management functions such as Provisioning, Device configuration, Firmware/Software update, Diagnostic and Monitoring, etc. This lightweight solution aims to reduce the payload footprint by 85% (average) and to lower the encoding complexity while maintaining backward compatibility with the widely deployed OMA-DM v1.x specifications on over 1.4 billion of mobile devices. This approach enables operators to reuse existing OMA-DM v1.x servers to manage constrained M2M devices. Existing OMA-DM Management Objects can also be reused. This solution contributes to enhance the Scalability, Energy efficiency and to further reduce the cost of devices. This solution is detailed in section 4.2.1.

3.1.1.2 Lightweight Device Management over CoAP

In the event existing OMA-DM v1.x server is not leveraged, a different Device Management approach is proposed, it is based on CoAP. This solution is detailed in section 4.2.2. Simple device management operation like change of configuration of a device, remote control of device (e.g. turn light ON/OFF), reporting a monitoring event or value (e.g. battery level) can usually be accomplished by sending a pair of key-value, where the key can be represented by a resource, and the value reflects the actual

command (e.g. 1 for ON and 0 for OFF). In addition, any other device management procedure that includes more complex scenarios (i.e. update of device software/hardware) can be also employed by using CoAP protocol, to manage a device as well as a group of devices. The CoAP protocol can be utilized for these operations due to available multicast and unicast requests, header option fields and the simple congestion control mechanism are applicable for the remote monitoring and the diagnostic of the devices. In order to employ CoAP for device management, protocol features are mapped with required device management procedures and DM module based on REST is defined.

3.1.2 Server initiated communication

Management authority set DM actions on M2M Server, these pending actions can either be polled by the end devices or pushed to end devices.
Polling approach: This is the most frequently used approach, as is it more scalable from the M2M Server perspective. Device Management operations are initiated by client devices, that is, the end devices makes the initiative to connect to M2M Server to start a DM "session". This action is usually predefined by a connection frequency set in the end devices. Basically, in this polling process, the end device is asking M2M Server whether there is any DM action to perform. Polling approach is not energy efficient as the end device has the polling scheduler which is affecting the duty cycle.
Push approach: This is also known as server initiated communication. This approach enables the device to optimize its idle cycle by reducing or eliminating the polling. For applications that do not require high reactivity, the Time To Live (TTL) of the SMS may be adjusted to device's sleeping duration. Whenever M2M Server needs to push new DM action to end device, the server can send a SMS notification message to have the device to start the DM "session". This approach is particularly suitable for time-critical applications where high reactivity is expected, but there is a trade-off with respect to energy efficiency.
As OMA-DM enablers can be reused in the proposed OMA-DM v1.x compliant Lightweight DM solution, OMA Device Management Notification Initiated Session enabler [62] can be leveraged to support this push approach. This is performed over the interface DM-1 (Figure 3-1 & section 3.3.2.5)
The alternative CoAP Lightweight DM protocol can operate over SMS, refer to section 4.2.2.2. Server initiated communication is thus enabled.

3.1.3 Management of Non-LTE-M devices behind Gateway

3.1.3.1 DM functions

OMA Gateway Management Object enabler [39] is leveraged in the ELFOMA in order to manage Non-LTE-M devices deployed behind M2M Gateways. DM functions such as device provisioning, software component management, configuration, etc.. can be supported over ELFOMA. M2M Gateway and Non-LTE-M CHs may have multiple air interfaces in order to communicate with mixing end devices using different short range communication technologies (Zigbee, Bluetooth, etc.)

In addition to supporting standard DM functionalities, the following aspects have been addressed:

- Self-organized capillary devices and network clustering
- Mobility management of capillary devices
- Multicasting and broadcasting message to a group of capillary devices

3.1.3.2 Self-organization of capillary network

Clusters in capillary networks are formed dynamically. This self-organizing capability aims to optimize energy consumption. As the battery level of a CH is gets low more quickly than the battery of a non-LTE-M device, the cluster can be reformed by occasionally changing the device that has the CH role. This algorithm is detailed in Section 7.

3.1.3.3 Mobility management

Mobility management of capillary devices is handled at the application level over the EXALTED Lightweight DM protocol. Upon attachment of a device to the M2M Gateway, this latter update the local device inventory which serves to manage groups and provides routing information. If required, the Gateway can automatically register the new device member to M2M server, so that network initiated command can be issued towards to this device through the gateway. Server initiated command helps the device to further save energy as the polling frequency can be lowered thus lengthen device's idle cycle. The inventory and registration process are also applied whenever a device is detached. This approach is described in section 8.7.

3.1.3.4 Multicast and Broadcast

Multicast and broadcast offload the traffic on the LTE-M link; the spectrum efficient and the system scalability are therefore enhanced. As such, M2M Server can target a group of devices by sending one message to the M2M Gateway, over the proposed EXALTED Lightweight DM protocol. The gateway fans the command out to devices belonging to the targeted group. This approach is described in section 8.6.

3.1.4 Device Reliability

Device reliability is an important objective in EXALTED. OMA Diagnostic and Monitoring Management Object enabler [60] can be leveraged in ELFOMA to fulfil this objective.
A novel Self-Diagnostic and Self-healing solution has been developed in EXALTED, Deliverable 6.3 [9]. This distributed self-diagnostic and self-healing approach relies on DM protocol to trigger actions, to distribute diagnostic rules, to collect status between nodes. Management Object has been defined in Deliverable 6.3 and can be supported by the proposed EXALTED Lightweight DM solution.

3.1.5 Device-to-device messaging

Device-to-Device messaging is an important feature to distribute the application logic to the device and gateway levels. For instance, sensors data can be posted to a gateway over the EXALTED Lightweight DM protocol, upon data analysis by the distributed application logic, the gateway may trigger application specific actions to actuators. Given the anticipated massive number of devices not being attended by human, the automation of application logic with a distributed approach contributes to enhance the scalability of EXALTED.
This new enabler can be supported by the proposed EXALTED Lightweight DM solution, and is described in section 6.3.2.

3.1.6 Services Extension

Low cost devices are mostly small and constrained. They have very limited resources such as CPU, memory, battery. Very low cost devices are not able to support the following required services, as various protocols must be embedded in devices e.g. device

management protocol, FTP, MQTT for messaging, self-diagnostic communication protocol, etc

- Management of capillary devices
- Data collection and uploading sensor data to M2M server
- Device control, triggering actuators
- Device Self-diagnostic and self-healing
- Device to device messaging

The envisioned service extension approach only relies on the EXALTED Lightweight DM solution to support the above services on low cost devices. This resource saving solution also reduces the complexity, thus the cost of the device; aspects such as communication concurrency, multiple protocol-related security credentials are no longer needed.

3.1.7 Security solutions

Main result is the promotion of the E2E security at application level. Integrity and confidentiality of the messages are preserved with well-known techniques. There is a split between secure application payloads that can be routed to the destination by whatever means that may change over time and the message transport part that enables to switch from a network to another one by sophisticated mechanisms proposed in the project.

Security of a system is a trade-off between resources optimization and protection. Some work done in the project enable to reduce the traffic by aggregating application payloads for example. There is no general answer to the contradiction between security and optimization objectives: either the service operator accepts that there is no E2E security and many optimization algorithms can be considered or optimizations cannot be applied.

The favourite security solution is to promote a hardware Secure Element soldered in the end-device because only a hardware component is secure enough to protect the communication and also because only an integrated chip able to be soldered can be produced at an acceptable price for the M2M industry.

Because of the high risk to be attacked there is no online pairing mechanism proposed as a generic solution. Pairing is a mechanism that sets a secret between two peers that do not share any pre-shared key or other secret in advance. Because Secure Elements are prepared in factory they do embed a “Fabrication secret key” that is used to securely bootstrap the Group Key setting which means that the pairing is done offline. There is a constraint with this design because the M2M server database must be provisioned with the related keys and device details before enabling a successful Group Key setting.

Together with the pairing the association between the device and the Secure Element is done offline at the manufacturer factory. The device and the Secured Element are “paired” to prevent the Secure Element to be used for another device

Secure Element managed by MNO and consequently the study of the protocol to switch from the current MNO to a new one is left to the specific security group of the project. The assumption is that the Service Provider manages the Secure Element.

3.2 DM Architecture

This diagram provides a high level architecture view of for device management functions and related services functions, as described in section 3. The functional components are described in next section 3.3

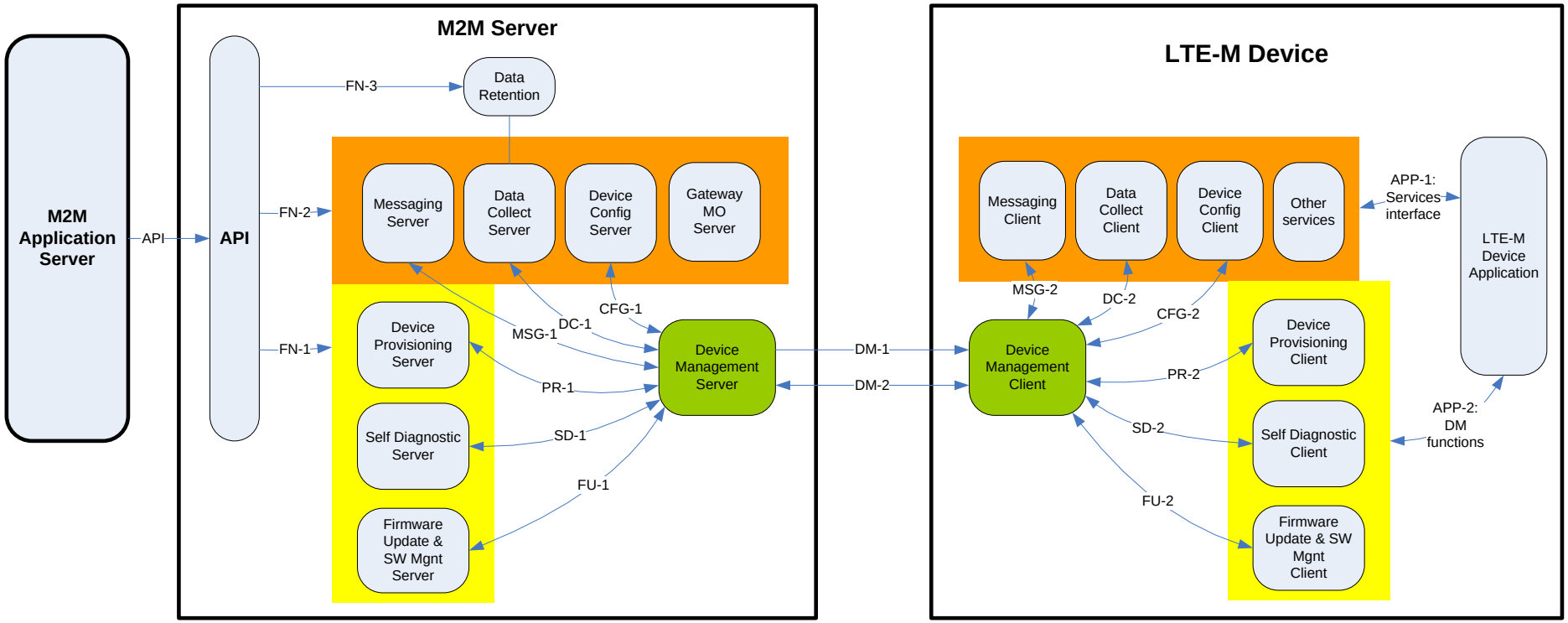


Figure 3-1: Direct Management of LTE-M Device

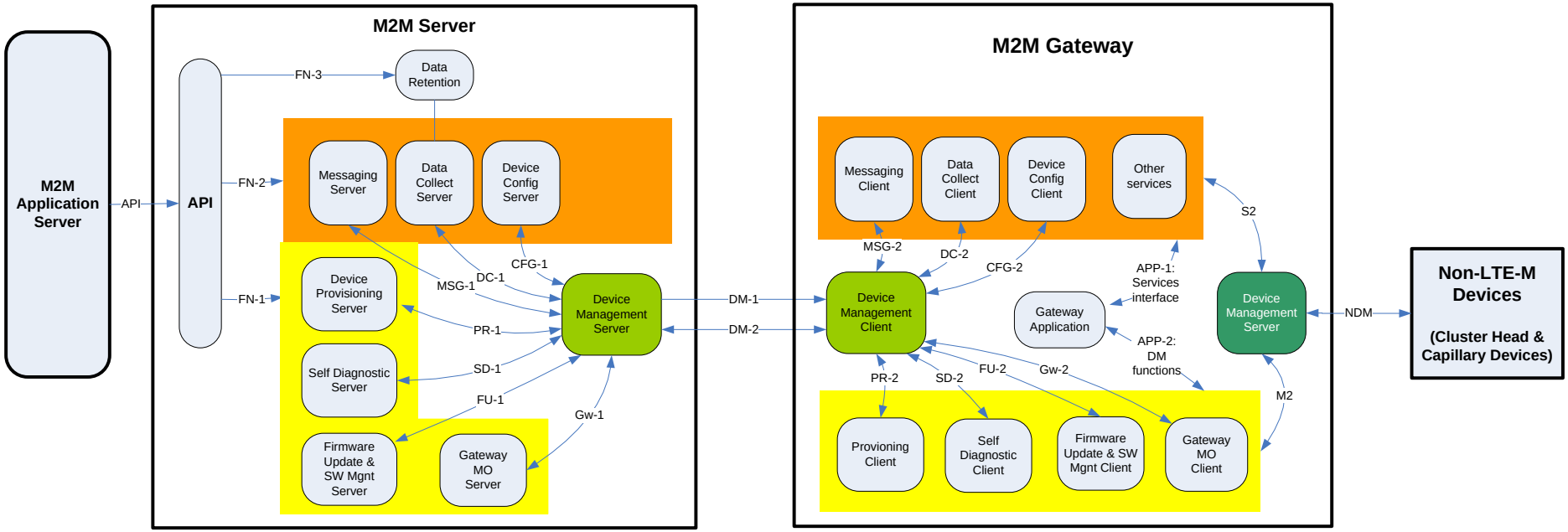


Figure 3-2: Indirect Management of non-LTE-M Devices through Gateway

Different DM related communication types can be derived from the EXALTED System architecture as depicted in Figure 2-1:

1. M2M Server manages LTE-M devices directly
2. M2M Server manages directly M2M Gateway which, in turns, manages Non-LTE-M devices (Capillary Devices and CHs)

Figure 3-1 and Figure 3-2 covers all above configurations

Communication within capillary network is disclosed in section 8.

3.3 Functional Components and interfaces

This section describes functionalities and interfaces of device management related components as depicted in Figure 3-1 and Figure 3-2.

3.3.1 Protocol Endpoints

3.3.1.1 M2M Application Servers

M2M Application Server is a logical entity that can host one or more M2M Applications. There may be multiple M2M Application Servers, one per application basis. Workflows that reflect Intelligent Transportation System (ITS), eHealth, Smart Metering and Monitoring scenarios are implemented by M2M Application. Service and business logics pertaining to the M2M Application is handled by this entity. For instance, a service and business logic of Smart metering application would collect meter indexes from households and generate billings process. This server is usually hosted by the owner of the application, e.g. Utilities Company. This endpoint is out of EXALTED scope.

3.3.1.2 M2M Server

M2M Server is a middleware logical entity operated by telecom operators or service providers. This entity comprises 2 high level logical components:

- Management functions (yellow block in Figure 3-1 and Figure 3-2), including DM usual functions such as
  - Device bootstrapping and provisioning
  - Firmware Update and Software Management Client
  - Self-Diagnostic
  - Management of devices behind Gateway
  - Other functions, such as OMA-DM enablers can be reused with ELFOMA DM solution (section 4.2.1), not all DM functions are described in this report.
- Service Capabilities (orange block in Figure 3-1 and Figure 3-2):
  - Data collection
  - Device remote control
  - Device-to-device messaging
  - Other services, not depicted in Figures, can be hosted in this block, for instance, automation logic, refer to section 3.5.6

M2M Server exposes helper functions such as device management functions (yellow block in Figure 3-1 and Figure 3-2) and other high level services capabilities functions (orange block in Figure 3-1 and Figure 3-2) to M2M Applications. Device management functions grouped in the yellow block can be mapped to EXALTED system architecture's 'M2M Device Management functions" block, and likewise Service Capabilities functions block can be mapped to "M2M Service Capabilities" block.

In addition to the provided valued-added services, this middleware architecture provides the following advantages to M2M Applications:

- Scalability. M2M Server, could be based on cloud-based model, is highly available and highly scalable. It is able to handle a large number of device connections. The front-end scalability of the system is handled by M2M server.
- Highly decoupled architecture. M2M Application providers only need to concentrate on business logics of their applications. Most technical aspects such as communication protocols, security, device provisioning mechanism, etc… are delegated to M2M Server. Hence, managing a heterogeneous fleet of devices is transparent to M2M Applications. E.g. devices with different communication protocols can be managed transparently.

- Interoperability is key factor to the growth of M2M market. This can be achieved by exposing standardized interfaces.
- Cost effective. M2M Server can be shared by different M2M Application Servers, operational cost can thus be minimized.

M2M Server comprises the following endpoint components:

3.3.1.2.1 M2M Server - API

This is the logical point of contact exposed by M2M Server to M2M Application Servers. Incoming function or service calls are then routed to the following underlying endpoints. This component is out of EXALTED scope.
In addition to the EXALTED system architecture aspect, EXALTED functional components can be reused as Enablers to build M2M solutions. For instance, device management components can be reused in ETSI M2M architecture oriented solution. In this case, the M2M Server API is already defined by ETSI TC M2M by the way of the mIa reference point which uses a Restful style interface.

3.3.1.2.2 Firmware Update & Software Management Server

This component is dedicated to issue Firmware Update and Software component management operations with LTE-M enabled devices and M2M Gateways. This Server interacts with the Firmware Update and Software component management Client hosted in LTE-M enabled devices including gateways. OMA Firmware Update Management Object (FUMO) and Software Component Management Object (SCOMO) enablers are leveraged [59] [64]. This component enables M2M Applications to deploy new firmware and software update by the way of the M2M Server. This is an essential feature to extend the life of the devices. New applications and services can be deployed remotely on massive devices, high reactivity to new business needs and cost saving objectives can thus be achieved.
Software management for Non-LTE-M devices in capillary network is addressed in section 8.

3.3.1.2.3 Self-Diagnostic Server

This component is dedicated to perform self-diagnostic and monitoring operations with LTE-M devices and M2M Gateways. This server interacts with the Self-Diagnostic Client hosted in end devices. OMA Diagnostics and Monitoring (DiagMO) enabler can be leveraged [60]. Distributed self-diagnostic and assisted-healing solution developed within EXALTED can be also be used in this server. This latter novel distributed self-diagnostic and assisted-healing approach increases the device reliability while offloading the monitoring and diagnostic traffic on the LTE-M link. To achieve this, the root failure detection and assisted-healing managers are deployed onto a tree hierarchy of nodes (M2M devices). Each node fan out the diagnostic and healing rules, received from upper level node, to child nodes or leaf nodes in capillary networks. On the M2M Server perspective, the rule is only sent once to the top parent node, which propagates it to lower level nodes. The traffic over the main link, namely between the M2M Server and the top parent node, is therefore significantly reduced. The diagnostic and healing management is deployed and delegated to the subsequent parent nodes. Each of these latter nodes collects and aggregates diagnostic results and alarms from lower level nodes, and forwards them to the adjacent parent node. The aggregated data is ultimately collected at the top level parent node and sent to the application over the main link. The two ways data traffic and signalling reduction can thus be achieved. Complete specification of this solution can be found in Deliverable D6.3 [9].

3.3.1.2.4 Device Bootstrap and Provisioning Server

This component is dedicated to perform bootstrapping and provisioning operations with LTE-M devices and M2M gateways. This server interacts with the Device Bootstrap and Provisioning Client hosted in the devices. Bootstrap is a process of provisioning the DM client to a state where it is able to initiate a management session to a new DM server.

Bootstrap can move a device from an un-provisioned, empty state, to a state where it is able to initiate a management session to a particular DM server. DM clients that have already been bootstrapped can be further bootstrapped to enable the device to initiate a management session to new DM servers. OMA Device Management Bootstrap enabler [61] is used in this solution.
The M2M Server can have multiple DM Servers, one per DM protocol basis. Devices embedding OMA-DM v1.x can be provisioned to connect to the OMA-DM v1.x Server, those embedded CoAP based DM protocol can be bootstrapped to the CoAP version of DM Server.

3.3.1.2.5 Messaging Server

This server provides device-to-device and server-to-device messaging capability. It interacts with the Messaging Client hosted in the devices. Messages posted by client devices are retained in this server until they are fetched by the recipient devices. Multicast messaging can also be enabled. The device-to-device messaging enabler will be detailed in section 6.3.2.

3.3.1.2.6 Data Collection Server

This component provides data collection and data retention capabilities. This server interacts with the Data Collection Client hosted in the devices. Data sent by client devices are saved in the data retention server within M2M Server. The data is then made available to M2M Application by the data retention component. The data collection enabler will be detailed in section 6.3.1.

3.3.1.2.7 Device Configuration Server

This component provides configuration and remote action capabilities. This server interacts with the Device Configuration Client hosted in the devices. This approach is described in section 6.2.2

3.3.1.2.8 Gateway Management Server

This server is dedicated to manage non-LTE-M devices behind M2M Gateway. It interacts with Gateway Management Client hosted in M2M Gateway, but not available in LTE-M devices. This server keeps track of capillary devices or group of capillary devices that are managed by gateways, directly or indirectly. Management functions such as bootstrapping, software component management and configuration of capillary devices can be achieved by this server. Management of capillary devices behind Gateway is described in section 8.

3.3.1.2.9 Device Management Server

EXALTED Lightweight DM solutions developed in EXALTED are leveraged by this component. This server interacts directly with the Device Management Client hosted in LTE-M devices and M2M Gateways. This is the key component that supports all above functions. Lightweight device management solutions are detailed in section 4.

3.3.1.3 LTE-M Device and M2M Gateway

LTE-M devices and M2M Gateway comprise 3 high level logical components:

- Application. M2M embedded client applications reside in this component
- Management functions (yellow block in Figure 3-1 and Figure 3-2), including Device Management usual functions such as
    - Device bootstrapping and provisioning
    - Firmware Update and Software Management Client
    - Self-Diagnostic
    - Management of devices behind Gateway

    - Other functions, such as OMA-DM enablers can be reused with ELFOMA DM solution (4.2.1), not all DM functions are described in this report.
- Service Capabilities (orange block in Figure 3-1 and Figure 3-2):
    - Data collection
    - Device remote control
    - Device-to-device messaging
    - Other services, not depicted in Figures, can be hosted in this block, for instance Data aggregation, algorithms developed within EXALTED can be leveraged in this logical component, refer to Deliverable D4.4, Traffic aggregation [6]

3.3.1.3.1 Application

Applications embedded in the device (LTE-M Device or M2M Gateway) are specific to envisaged use cases. For instance, collected sensor data may be packed and encrypted using a predefined security credential and file format defined by the M2M Application. The embedded device/gateway application uses the following endpoints components (sections 3.3.1.3.2 through 3.3.1.3.10) to interact with the remote M2M Application.

3.3.1.3.2 Firmware Update and Software Management Client

This component resides in the device and interacts with the Firmware Update and Software Management Server, as described in section 3.3.1.2.2, to download firmware and software components over the air. The firmware update and software management (install, upgrade, removal) processes are then performed by embedded agents. Operation status is reported back to the counterpart server. OMA Firmware Update Management Object (FUMO) [59] and Software Component Management Object [64] enablers are leveraged in this component. Software update process for capillary devices is disclosed in section 8.

3.3.1.3.3 Self Diagnostic Client

This client performs self-diagnostic, self-healing and monitoring actions as driven by the Self Diagnostic server, refer to section 3.3.1.2.3. Two solutions can be supported:

1. OMA Diagnostics and Monitoring (DiagMO) enabler can be leveraged [60].
2. Distributed self-diagnostic and assisted-healing solution [9] developed within EXALTED can be also be used. With this novel solution, this client has rule-driven root failure detection capability. Upon failure detection, the provided self-healing rule is executed to attempt to fix the issue. In the event the failure cannot be resolved, the Self-Diagnostic server is notified. This latter has the ability to update the self-healing rule in order to enhance the healing algorithm. This component also includes a self-diagnostic server. This latter sub component serves to manage child level nodes in order to apply this logic recursively within the tree hierarchy of nodes.

3.3.1.3.4 Device Bootstrap and Provisioning Client

This component interacts with the Device Provisioning server to perform provisioning operations, refer to section 3.3.1.2.4. Upon bootstrap notification process, device provisioning information is retrieved. This information enables the device to initiate DM session with DM Server, for instance: security credentials, M2M Server URL or IP address, LTE-M network configuration, etc. OMA Device Management Bootstrap enabler [61] is used in this solution. This is an important process to be completed before any further device management functions or services related functions can be initiated.

3.3.1.3.5 Messaging Client

The device/gateway application uses this messaging client to post and receive message to/from other device(s). This component interacts with the messaging server, as described in section 3.3.1.2.5, to complete this task. The device-to-device messaging enabler will be detailed in section 6.3.2.

3.3.1.3.6 Data Collection Client

The device/gateway application drives the data collection and data aggregation policy. Aggregated data is posted to the Data Collection Server, section 3.3.1.2.6, by this client. The data is then collected asynchronously by the intended M2M Application. The data collection enabler is detailed in section 6.3.1.

3.3.1.3.7 Device Configuration Client

This component interacts with the Device Configuration Server and is dedicated to retrieve and apply configuration set by M2M Server, section 3.3.1.2.7. Device Control (e.g. turn light on/off) feature is enabled by this component. This approach is described in section 6.2.

3.3.1.3.8 Gateway Management Client

This component is only required in M2M Gateway. It enables the M2M Server to manage Non-LTE-M devices and CHs indirectly, as direct communication between the M2M Server and capillary devices is not possible. Therefore, the M2M Gateway is managed by M2M Server, and in turn, the gateway manages capillary devices under it. This client keeps track capillary devices being attached. Fan out mechanism enables multicast and broadcast functionalities. This client interfaces with the Gateway Management Server, section 3.3.1.2.8, over the underlying device management protocol. OMA Gateway Management Object enabler [39] is used. This enabler defines 3 management modes: Transparent mode, Proxy mode and Adaptation mode.

- In Transparent mode, the M2M Gateway does not participate in DM session. The gateway just forwards message received from server to end device, and from device to server. This mode implies devices to support the envisioned EXALTED Lightweight DM client and protocol.
- In Proxy mode, the M2M Gateway manages capillary devices (Non-LTE-M devices and CHs) behind the gateway on behalf of the M2M server over the DM protocol. Two related DM sessions are thus established. One is between the M2M Server and the M2M Gateway (working as DM Client); the other is between the Gateway (working as DM Server) and the capillary device(s) behaving as DM client. This mode also implies devices to support the envisioned EXALTED Lightweight DM client and protocol.
- In Adaptation mode, M2M Gateway manages capillary device(s) behind the Gateway on behalf of the M2M Server without using the envisioned EXALTED Lightweight DM protocol.

Adaptation mode is most likely used by Gateways to manage capillary devices (Non-LTE-M devices and CHs), for the following reasons:

- Capillary devices are mostly off-the-shelf devices, EXALTED Lightweight DM protocol may not be supported natively. In this case, Transparent and Proxy modes are not applicable.
- Capillary devices are mostly constrained and likely to have duty cycle. The adaptation mode enables the M2M server to manage devices asynchronously, that is, DM commands issued by M2M server can be sent to gateway while the device is in sleep mode. The received DM commands can then be relayed to device upon wake-up. The proxy mode does not support this asynchronous communication need.

3.3.1.3.9 DM Client

This component implements the proposed EXALTED Lightweight DM protocol. This client interacts with the Device Management Server, 3.3.1.2.9, hosted in the M2M Server. All aforementioned functions are supported by the DM protocol. EXALTED Lightweight DM protocols are detailed in section 4.

3.3.1.3.10 Gateway DM Server

This component does not implement the EXALTED Lightweight DM protocol. It makes use of an interworking proxy component to interact directly with Non-LTE-M devices over the legacy protocol (e.g. using multiple air interfaces such as ZigBee, Bluetooth, etc...) in order to control the end device and to invoke action.

3.3.2 Interfaces

3.3.2.1 API interface

This interface allows M2M Application to access DM functions and services functions in M2M Server. The API definition is out of EXALTED scope.

3.3.2.2 FN-1 interface

M2M Applications access to DM functions (yellow block in Figure 3-1 and Figure 3-2) indirectly by the way of this interface. The definition of this interface is out of EXALTED scope.

3.3.2.3 FN-2 interface

M2M Applications access to Services functions (orange block in Figure 3-1 and Figure 3-2) indirectly by the way of this interface. The definition of this interface is out of EXALTED scope.

3.3.2.4 FN-3 interface

M2M Applications can retrieve device data by the way of this interface. Device data sent by devices are retained in the Data retention component, and made available to M2M Applications. Access to data is controlled per M2M Application basis. The definition of this interface is out of EXALTED scope.

3.3.2.5 DM-1 interface

DM Server can issue DM Notification [62] to M2M devices over this interface. This notification is used to bootstrap and provision devices. It can also serve as a wake-up message in order to initiate a DM session. This Server-initiated communication approach enables the devices to have a longer sleeping cycle or to increase the idle cycle by reducing the polling frequency. Energy efficiency of devices is therefore enhanced. This notification is issued by Device Bootstrap and Provisioning server and sent over SMS to targeted M2M Devices. Upon reception by DM Client, the notification is to be authenticated by the Device Bootstrap and provisioning client. The device can then be provisioned or new DM session can be started upon successful authentication.

3.3.2.6 DM-2 interface

EXALTED Lightweight DM protocols (as proposed in section 4.) are supported on this interface, over which DM client initiates the EXALTED Lightweight DM protocol, then DM Server may send DM commands to DM Client and this latter may return status and alerts to DM Server.

3.3.2.7 APP-1 interface

Embedded application in LTE-M devices and M2M Gateways uses an internal APP-1 interface to access Service Capabilities (orange block in Figure 3-1 & Figure 3-2) such as device-to-device messaging, data collection, data aggregation.

3.3.2.8 APP-2 interface

This interface allows embedded application to access Management functions (yellow block in Figure 3-1 & Figure 3-2) such as firmware update, software management, diagnostic, etc…

3.3.2.9 M2 interface

This interface is only required for M2M Gateway to enable DM functions to fan out DM commands to Gateway DM Server or to receive sensor status and DM request.

3.3.2.10 S2 interface

This interface is only required for M2M Gateway, and is used to enable Service Capabilities to fan out service commands to Gateway DM Server or to receive sensor data and service request.

3.3.2.11 NDM interface

It enables the M2M Gateway to send device commands to Non-LTE-M devices and, to receive sensor status and data from the latter end devices.
This interface does not implement the EXALTED Lightweight DM protocol. Instead, after DM commands being translated to device commands by the Gateway DM Server, they are sent over this interface using the legacy protocol (e.g. Zigbee, Bluetooth, etc…)

3.3.2.12 FU-1 & FU-2 interfaces

The FU-1 interface allows the Firmware Update and Software Management Server to send firmware update or software management commands to DM Client (residing in LTE-M device or M2M Gateway) using the underlying DM-2 interface. DM Client then uses FU-2 interface to forward commands to Firmware Update and Software Management Client, in order to invoke the requested firmware update or software management operation.
FU-2 interface also allows the Firmware Update and Software Management Client to report execution status back to the remote DM Server using the underlying DM-2 interface. The received status is then forwarded to M2M Server's Firmware Update and Software Management Server using the FU-1 interface.
Software life cycle management of Non-LTE-M devices behind the gateway is handled by Gateway Management components, described in section 8.

3.3.2.13 SD-1 & SD-2 interfaces

The SD-1 interface allows the Self-Diagnostic Server to send self-diagnostic and self-healing commands and rules to device DM Client using the underlying DM-2 interface. DM Client then uses SD-2 interface to forward the received commands and rules to Self-Diagnostic Client, in order to invoke the self-diagnostic operation or to update rules. For M2M Gateway, Self-Diagnostic client may fan out the received commands and rules to CHs and to Non-LTE-M devices by the way of the local Gateway DM Server over the M2 interface then NDM interface. Likewise, M2M devices behind the M2M Gateway can report diagnostic results and failure detections report back to the self-diagnostic client using interfaces NDM and M2.
SD-2 interface also allows the Self-Diagnostic Client to report aggregated diagnostic results and failure detections report to the remote DM Server using the underlying DM-2 interface. The received data is then forwarded to Self-Diagnostic Server using the SD-1 interface.

3.3.2.14 PR-1 & PR-2 interfaces

The PR-1 interface allows the Device Bootstrap and provisioning Server to provision LTE-M device and M2M Gateways using the underlying DM-2 interface. DM Client then uses PR-2 interface to forward the received notification or provisioning information to Device Bootstrap and provisioning Client.
Provisioning of Non-LTE-M devices behind the gateway is handled by Gateway Management components, disclosed in section 8.

3.3.2.15 MSG-1 & MSG-2 interfaces

MSG-2 interface allows the Messaging Client to send device-to-device messages to the DM Server using the underlying DM-2 interface. DM Server then uses MSG-1 interface to

forward them to Messaging Server. The received messages are retained in the Messaging Server until they are delivered asynchronously to intended recipients. That is, the pending message is delivered upon the next connection of the recipient device. When a new DM session is initiated, Messaging Server uses MSG-1 interface to redirect pending messages to the remote DM Client, over the EXALTED Lightweight DM protocol that is supported on DM-2 interface. Messaging Client consumes device-to-device messages delivered by DM Client over the MSG-2 interface.
For M2M Gateway, the embedded Messaging component also comprises a Messaging Server (not depicted in Figures) which redistributes the incoming messages to Non-LTE-M recipient devices behind the Gateway. This Gateway Messaging Server also aggregates device-to-device messages sent by Non-LTE-M devices. Note that CHs can relay Non-LTE-M devices' messages to Gateway. These two-ways data flows are enabled by the internal Gateway DM Server using the internal S2 interface and NDM interface.

3.3.2.16 DC-1 & DC-2 interfaces

DC-2 interface allows the Data Collection Client to send sensor data to the Data Collection Server, in M2M Sever, using the underlying DM-2 interface. DM Server then uses DC-1 interface to forward them to Data Collection Server. The received data is then made available to M2M Application by the way of the Data Retention component.
For M2M Gateway, the embedded Data Collection component also comprises a Data Collection Server (not depicted in Figures Figure 3-2) which aggregates sensor data sent by Non-LTE-M devices. Note that CHs can relay Non-LTE-M devices' data to Gateway. Sensor data is sent to Gateway's DM Server over the NDM interface, the device originated data is then forwarded to Gateway's Data Collection server over the S2 interface.

3.3.2.17 CFG-1 & CFG-2 interfaces

CFG-1 interface is used by Device Configuration Server to send configuration data to devices' DM Client using the underlying DM-2 interface. DM Client then uses CFG-2 interface to forward the received configuration data to Device Configuration Client in order to actually configure the device.
Device configuration of Non-LTE-M devices behind the gateway is handled by Gateway Management components, refer to section 8.

3.3.2.18 GW-1 & GW-2 interfaces

Management of Non-LTE-M devices behind M2M Gateways is ensured by Gateway Management Client and Server. The management scope covers device inventory, bootstrapping, provisioning, configuration, software installation/removal, software upgrade. In addition to the unicast model, DM commands can also be fan out by the Gateway to a group of multiple Non-LTE-M devices.
GW-1 interface is used by Gateway Management Server to send management commands to devices' DM Client using the underlying DM-2 interface. DM Client then uses GW-2 interface to forward the received commands to Gateway Management Client. This latter then fans out commands to Gateway's DM Server over the M2 interface. DM commands are translated by Gateway's DM Server to device commands as supported by Non-LTE-M devices, prior being sent to recipient end devices over the NDM interface, by selecting an appropriate air interface (Zigbee, Bluetooth, etc...).

3.3.3 Leveraging other work packages

Algorithms and solutions developed within other EXALTED packages can be used as underlying enablers for Device Management solutions. For instance:

- DM-1 and DM-2 interfaces implementing the EXALTED Lightweight DM protocol. As this high level protocol can be binded to any IP based transport level, hence it can be

realized over LTE-M [3] to optimize the bandwidth usage, LTE-M is designed to spectrum efficient

- Likewise, DM-1 and DM-2 interfaces can also be realized over an underlying IPv6 based network as disclosed in Deliverable 4.2 [5]
- NDM interface between Non-LTE-M devices within capillary network can use MAC protocols proposed in Deliverable D4.1 [4] as underlying efficient communication medium to yield further energy saving.
- Data aggregation solutions proposed in Deliverable D4.4 [6] can be leveraged in interface NDM and in data collection components
- Self-Diagnostic components and their SD-1/SD-2 interface are compatible with the monitoring sensor network solution proposed in Deliverable D4.5 [7]. Self-Diagnostic Master and Module as described in Deliverable D6.3 [9] can be mapped to Self-Diagnostic components.
- In addition to the need for an E2E security some security requirements and recommendations have been published in Deliverable D5.1 [8] that have been used to design the current proposal

3.3.4 Extension of DM functions and Service Capabilities

It should be noted that EXALTED DM solutions are not limited to the functional components described in the previous sections. New DM functions and new service capabilities can be added in the future. For instance, other OMA MO enablers can be added into the Management Function logical entity (yellow block in Figure 3-1 and Figure 3-2) such as LAWMO [71], DCMO [72], etc. Likewise, new service capabilities can be added into the Service Capabilities logical entity (orange block in Figure 3-1 and Figure 3-2), such as Device addressing (refer to section 6.3.3), automation and scripting, etc.

3.4 Security considerations

3.4.1 Application Programming Interface (API)

The M2M embedded application may interface with a component that supports the security features required by the application. Assumption in EXALTED is that this component is implemented through a dedicated Secure Element. Whatever is the implementation of this component the M2M application is willing to rely on an interface that is either standard or supported by a consortium to avoid a dependency on a specific provider. A good design is to avoid using a proprietary API in the application and to rely on an interface that is available and maintained by a community or a standard organization.

The favourite language for this security interface is C/C++ because it is the most widely supported by board providers. There is a tendency to promote the Java language for applications using a smartcard running JavaCard. The binary format of the protocol [73] used to communicate with a smartcard is not bound to any specific language. The fact that an embedded applet on the smartcard is implemented in JavaCard does not impose any implementation language for the application on the device.

The SIMalliance organization has put in place an Open Mobile API workgroup [65] to specify an API to enable mobile application to “have access to different Secure Elements in a Mobile such as SIMs or embedded SEs”. The API is described in an abstract mode (UML diagrams) that does not promote a specific implementation language. Even if it has been initially designed for Mobile phones it is generic enough to be recycled for the benefit of M2M applications. Using a C/C++ implementation of this specification to interface the Secure Element is the favourite choice. For the purpose of the project this API is extended with cryptographic functions.

Together with a standard API an implementation must be provided to support the device part of the communication with the Secure Element. There are two approaches for a Secure Element provider either the implementation is provided in source code form to be integrated by the application or it is provided in binary form as a library but it means that many processor type and operating system releases have to be supported that is quite an effort.

3.4.2 Application code location

In case the communication between the server and the device has to be secured and according to the complexity of the application code it is possible to locate the main part of the logic either on the application processor or on the Secure Element.

A first option is to classically process data on the application processor and to subcontract the security part to the Secure Element. An alternative option is to assign synchronization tasks only to the application on the device and to deport the full application logic to the Secure Element. Tasks like device management operation, payload building, M2M server commands interpretation could be implemented in the Secure Element. With this latter approach the part of the application that runs on the application processor can be designed as a Final-state automaton where all data received from the RF modem are routed to the Secure Element. There is no clock in the Secure Element. An internal scheduler must poll the Secure Element that on its turn sends commands to be processed by the application.

An intermediate option is to let the Secure Element to format and interpret the payloads sent/received to/from the M2M server and to manage the interface with the meter or any business oriented associated components.

3.5 Scalability considerations

Machina Research forecasts that the installed base of M2M connected Consumer Electronics devices will exceed 4.2 billion by 2020 [11]. Cisco created an infographic [12] that depicts the increasing number of ‘things’ connected to the internet. Similar to Ericson predictions [13], it states that by 2020, there will be 50 billion 'things' connected to the Internet.

Managing the anticipated massive number of M2M devices is a challenge. M2M devices are exchanging small amount of data with M2M applications over the radio access network. As signalling overhead may be large compared to small size of data being transmitted, the spectrum usage may not be optimized. The following scalability aspects are considered on the E2E point of view.

3.5.1 Infrastructure

M2M Server is the central point for device management. It centralizes DM logics issued by management authorities and drives the DM process until completion. M2M Server must be able to handle communication with massive number of M2M devices. When adopting connection-oriented DM Protocols (e.g. OMA-DM v1.x and ELFOMA 4.2.1), M2M Server consumes resource in order to keep track of on-going sessions which may cause scalability issues on traditional servers. Additional servers need to be added to the system in order to cope with the peak loads and maintain the system availability. This approach is not adaptive and not cost effective, as the server usage are not optimized when the load goes off.
Adopting a cloud based infrastructure helps to increase in a timely manner the availability and scalability of the M2M Server while optimizing the ROI (return on investment). Server instances can be automatically and dynamically created to handle additional M2M device requests, load balancers redirect request to a proper instance that contains the session context, thus enhancing the availability of M2M Server. When connection load is getting lower, server instances are automatically pulled off, the OPEX (operational expenditure) is therefore constantly adjusted to the level of server usage.

3.5.2 DM protocols

The scalability can be controlled using appropriate DM protocols. As M2M devices are managed by M2M Server over DM protocol. The complexity and verbosity of the DM protocol can affect the system scalability.

More M2M Server resources are required to process complex DM protocol. This complexity directly affects the delay for the server to response to incoming device requests. A low rate of requests processed per second negatively impacts the scalability of M2M Server. For instance, XML parser is required to parse XML payload, the use of dictionary is needed for other encoding scheme. Further memory resources are required to process verbose DM messages.

Connectionless oriented DM protocols (such as CoAP) are less resource demanding compared to connection oriented DM protocol (e.g. OMA-DM v1.x and ELFOMA).

EXALTED proposes different EXALTED Lightweight DM solutions for different business justifications. ELFOMA is recommended for operators or service providers wishing to reuse existing OMA-DM v1.x servers and OMA enablers. This solution aims to reduce the message payload while reducing the encoding complexity. Connectionless oriented DM protocols based on CoAP are also proposed.

3.5.3 Architecture

Architecture also plays an important role in the scalability landscape. 2 keys DM architectural concepts are contributing to the system scalability. The number of connection over the LTE-M main link can be significantly reduced using distributed DM and multicast/broadcast approaches as described in section 8. Network traffic and DM sessions pertaining to management of devices behind M2M Gateway are delegated locally at the capillary network to M2M Gateway or CHs. DM status are collected by CHs and ultimately aggregated by M2M Gateway prior sending to M2M Server. The number of simultaneous DM sessions on M2M Server is hence reduced.

In addition, the multicast and broadcast mechanism enables M2M Server to send DM commands at once over the LTE-M main link to target multiple devices belonging to a group or to an entire branch of end devices within a capillary network.

3.5.4 Algorithms

As mentioned in section 3.3.3, aggregation algorithms disclosed in deliverable D4.4 [6] can be leveraged in DM architecture. They make capillary networks more scalable, as data is collected and aggregated prior sending to upper node level and communication congestion is reduced.

3.5.5 Access Network

On the radio access network point of view, LTE-M has a better scalability compared to LTE. As LTE-M is tailored for M2M communication, signalling overhead and various simplifications enables an eNodeB to support a higher number of M2M devices than LTE can support. Spectrum usage has been optimized for transmitting small amount of data. For more information related to LTE-M refer to Deliverable D3.3 [3].

3.5.6 Automation

3.5.6.1 Monitoring process

Automation of service logics is another leverage to scale up the system. DM procedures issued by management authorities can be automated through a progressive monitoring process. For instance, a firmware update operation to be applied on 10 million devices will potentially cause scalability issue on M2M Server, as devices would be connecting to server almost all at once. A Campaign Manager component can be introduced into the M2M Server to split the firmware update operation onto subsequent operations applied to smaller fleets of devices. Furthermore, upon receiving the trigger from M2M Server, M2M devices does not

establish the communication immediately with M2M Server, instead a randomized delay is applied. Huge peak of simultaneous device connections can thus be avoided. This Campaign Manager component also track devices that failed to proceed firmware update, retry mechanism can be applied automatically based on a predefined retry policy. A high level E2E workflow is disclosed in section 8.6.3.

To handle various automation needs, a rules based scripting engine can be introduced in the M2M Server. M2M Applications can upload scripts on M2M Server to create or to update automation processes. Device diagnostic and healing can be automatically applied without human attendance, further OPEX saving can thus be achieved. For instance, a script configure monitoring process on device (define which indicator to monitor). In the event of an alert which indicates a potential upcoming service failure on device, the script drives the DM Server to send further diagnostic commands or healing rules to device. Based on the outcome of diagnostic results, the script may decide to reconfigure the device or to upgrade the device to a newer software or firmware release.

3.5.6.2 Generalization

Using a Campaign Manager together with a Scripting Engine as described in the previous section fits well with the E2E security. Regarding security the only scenario studied for the automation is to use M2M Gateway and CHs as network components only. DM commands are prepared on the server side under the form of scripts that are encrypted and signed using DM keys. These scripts are conveyed by M2M Gateways and CHs without interpreting the semantic of the scripts. The only feature required on the M2M Gateway/CH is to cache scripts to minimize the traffic.

As described above the Scripting Engine runs some rules. One design option is that DM actions are triggered on application message answers sent back by the M2M server. Because of the E2E security requirement there is no M2M Gateway Agent to relay the M2M server then the preferred way to notify the non LTE-M device to run a DM script is on application message answers. This lazy management way is only there to optimize the M2M server activity. Note that on critical application end-device could be directly notified.

On the M2M server a script has been previously prepared as described in 3.5.6.1 and it is ready to be downloaded by a set of end-devices. When notified by the Campaign Manager, the end-device DM Agent downloads the DM script, checks its signature, decrypts the script and runs it. With this pull oriented model the M2M server is less impacted and the scalability of the server is preserved. An optimization would be to define all DM command script scenarios in advance and to implement them in the device DM Agent. Only a script identifier and arguments would be retrieved from the server.

An alternative design would require having DM Agent on M2M Gateways but the consequence is also to have the related Secure Element on the M2M Gateways but this is not compatible with E2E security.

3.6 Other considerations

High level concepts are provided for the following business need.

Multiple DM protocols may be supported in M2M Server, this may be needed to manage heterogeneous fleet of M2M devices running different DM protocols or different DM versions. Furthermore, for operational reason, all devices may not be upgraded to the latest software version. The proposed DM architecture can support this need. There will be multiple DM Server logical entities, one per DM protocol basis, they all interface with the aforementioned enabler servers. M2M Devices will be bootstrapped to a particular DM protocol, by provisioning the proper DM server information (i.e. DM Server URL) to devices.

4. Device Management Protocols

This section details two EXALTED Lightweight DM protocols. The first proposed solution (section 4.2.1) is an adaptation of OMA-DM and it is compliant with OMA-DM v1.x servers. The second solution (section 4.2.2) is based on CoAP. Security aspects are considered in 4.3.
Existing solutions are first described in the following section.

4.1 Existing and Future solutions

The following existing DM solutions have identified and described hereafter: TR-069, OMA-DM's current and future solutions, CoAP.

4.1.1 TR-069

This protocol entitled CPE WAN Management Protocol (CWMP) is specified in the Technical Report TR-069 by the Broadband Forum (former DSL Forum) [34]. This application layer protocol is widely used for remote management of internet access end-devices, such as gateway, modems, routers, set-top boxes and VoIP-phones. This latter Customer-Premises Equipment (CPE) oriented devices are auto-configured by an Auto Configuration Server (ACS). In addition to auto configuration, ACS also provides other CPE management functions such as dynamic service provisioning, Firmware management, software management, diagnostics, status and performance monitoring.
The interconnection between CPE and ACS has the following characteristics:

- Bidirectional SOAP/HTTP-based protocol supporting standardized Remote Procedure Calls (RPC) Methods
- Communication security can be ensured by the use of SSL/TLS at the transport level (HTTPS), or the use of shared secret for authentication over HTTP
- CPE can only be managed by one ACS. CPE Management by multiple ACS is thus not possible
- Management session is always started by CPE
- ACS can issue Connection Request to CPE over HTTP. Upon receiving and authenticating the connection request, CPE will start a management session within 30 seconds
- In order to issue a connection request, the ACS must have the IP of the targeted CPE
- ACS can use STUN (Session Traversal Utilities for NAT) mechanism to reach CPE located behind a NAT

TR-069 is well designed to manage wired end-devices, such as internet access gateways, set-top boxes, modems etc. Standardized data model defined by TR-106 ensures interoperability of discovery and management functions. The support of STUN mechanism enables ACS to manage end-devices located behind NAT. In addition, CWMP is endorsed by Home Gateway Initiative, Digital Video Broadcasting and WiMAX Forums as the protocol for remote management of home network devices and terminals.
Despite of these strengths, TR-069 is not well suited for remote management of M2M constrained devices due to the following issues:

- Connection request is only defined for HTTP. Connection Request over SMS is not defined
- Spectrum efficiency
  - Large payload due to SOAP (XML-based)
  - One command per HTTP message yielding large traffic
- Energy efficiency

    - Correlates with the verbosity of the protocol
    - STUN mechanism also drains device battery
- Capillary devices support
    - Lack of support for gateway and capillary devices
- Scalability
    - Stateful management session, connection oriented protocol
    - No broadcast & multicast to capillary devices

4.1.2 Open Mobile Alliance Device Management (OMA DM)

In Open Mobile Alliance (OMA), the DM Working Group is responsible for the standardization of Device Management activities. The latest "Candidate" specification is version 1.3 while the latest "Approved" specification is version 1.2.1 [35].

4.1.2.1 OMA DM v1 .x

OMA DM versions 1.x are designed for remote management of mobile devices such as mobile phones and PDAs. OMA DM uses XML for message exchange between the managed device and the management server. XML messages are formatted based on Document Type Definition (DTD) defined by SyncML. The DTD, common XML elements, message container elements, data description elements and protocol command elements are specified in OMA DM Representation Protocol [36]. The device management workflow is defined in OMA DM Protocol [35]. A DM session is started by the managed device and is composed of a setup phase and a management phase. Data model is defined by standardized Management Objects with tree-based structure. Device resources can be mapped to Management Objects and addressed by the DM server using protocol command XML elements. In addition to provisioning, device configuration, other DM functions can be supported through additional enablers such as FUMO for Firmware update, SCOMO for software management, DiagMO for Diagnostic, etc.
The interconnection between managed device and DM server has the following characteristics:

- Communication protocol is request-response protocol, and can be implemented over various transport protocol: WAP, HTTP, OBEX or similar transports
- Communication security can be ensured by the use of SSL/TLS at the transport level (HTTPS), or the use of built-in authentication and challenge of authentication
- DM session is always started by the managed device
- DM server can send a notification to device in order to trigger a DM session. This notification message can be delivered over SMS, OBEX or other transport protocols

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. According to the announcement made by OMA in Mobile World Congress in Barcelona on February 2012, the total number of mobile devices deploying OMADM specifications reaches 1.4 billion.
Furthermore, M2M Devices residing in a capillary network and without an IP address can be managed through a M2M Gateway using OMA Gateway Management Object enabler (GwMO) [39]. However, OMA DM 1.x specifications are not suitable for M2M constrained devices due to the following reasons:

- Spectrum efficiency
    - Large payload due XML-based message
    - However, WBXML has been defined to reduce the size of message size
- Energy efficiency
    - Correlates with the verbosity of the protocol

- Scalability
  - Stateful management session, connection oriented protocol
  - However, sessionless message approach has been considered

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.

4.1.2.2 OMA DM NG (Next Generation)

OMA has recently approved a work item called DM NG (Next Generation) v1.1 [37]. It is targeted to greatly improve the efficiency of the existing DM v1.x 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 v1.x protocol 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 recommended DM WG to analyse the option about transitioning to a RESTful architecture for DM NG. RESTful style architecture provides several advantages e.g. improving server scalability. The contribution also 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 introduced CoAP [33] with the intention to jumpstart the discussion on adapting DM NG for M2M Applications. As CoAP 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.

Improvements suggested by OMA DM NG certainly lead to simplification of the OMA-DM v1.x protocol along with the message payload reduction enabling network bandwidth optimization. However, the proposed protocol is no longer backward compatible with DM v1.x due to proposed protocol simplification and encoding scheme.

4.1.2.3 OMA DM 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” [38] 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. The solution must be spectrum efficient
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.

OMA DM Lightweight M2M work item has started recently 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 the widely deployed OMA-DM v1.x. Moreover, Lightweight M2M does not manage sensor or capillary devices residing behind a gateway.

4.1.3 Constrained Application Protocol (CoAP)

This protocol has already been mentioned in the D4.1 document [4]. The purpose of this section is to describe the protocol to check later how it can be used as a mean of the management of the secure element. Goals of CoAP is to be easily translated to HTTP for an easy integration with existing services and to provide a low overhead messages suitable for low resource M2M devices. Features as described in the IETF draft [33] are listed hereafter for future reference:

- Constrained web protocol fulfilling M2M requirements.
- UDP binding with optional reliability supporting unicast and multicast requests.
- Asynchronous message exchanges.
- Low header overhead and parsing complexity.
- URI and Content-type support.
- Simple proxy and caching capabilities.
- A stateless HTTP mapping, allowing proxies to be built providing access to CoAP resources via HTTP in a uniform way or for HTTP simple interfaces to be realized alternatively over CoAP.
- Security binding to Datagram Transport Layer Security (DTLS)

Compared to the client /server model of HTTP/TCP both endpoints may in turn take on either the role of the client or the server. This is why the term endpoint is preferably used.

Main feature of the protocol is its asynchronous communication mode and the associated optional reliability. Because the protocol is based on UDP it can broadcast multicast IP requests to many endpoints without bothering with messages induced by the reliability, data integrity and ordering of HTTP/TCP. “UDP's stateless nature is also useful for servers answering small queries from huge numbers of clients.”

4.1.3.1 Reliability

The inherent poor reliability of UDP is counterbalanced in CoAP with an error checking and correction mechanism. Because there is no session and then packets might get lost then a type is assigned to a message. One possible message type is CONfirmable (CON) which means that it must be answered by the recipient with the ACKnowledgement (ACK) mark. If a message with the ACK message type value is not answered before a default timeout then it is sent again. These CON and ACK values are combined with a message identifier to enable the tracking and to avoid duplicate processing. Should the message be confirmable or not it can still be answered with a ReSeT (RST) message type value, which means that the recipient is not able to process the message. A message that doesn't need to take advantage of this confirmation delivery mechanism is set with the NON-confirmable (NON) message type value.

```
CoAP message type ::= {CON, ACK, RST, NON}
```

Note that the recipient can drop a non-confirmable message without sending a reset message to clearly notify the endpoint that the message is rejected.

When an answer cannot be provided immediately to a confirmable request then an empty acknowledgement is answer to prevent the request to be sent again. When available the response is sent in another confirmable message that has to be answered in turn. Note that mechanism can also be used if the initial message is not confirmable.

4.1.3.2 Uniform Resource Identifier (URI)

The IETF document specifies a CoAP URI scheme that enables to specify the resource location.

```
coap-URI = "coap:" "//" host [ ":" port ] path-abempty [ "?" query ]
```

The definitions of "host", "port", "path-abempty" and "query" used in the above specification are adopted from the "Uniform Resource Identifier (URI): Generic Syntax" RFC [RFC3986].

If the port is not specified then the default 5683 is assumed.

4.1.3.3 Methods

As promoted by the REST specification each HTTP method have a corresponding mapped adequate behaviour related to the M2M business field. It means that GET, POST, PUT and DELETE methods have a redefined semantic in order to be inline with the initial goal. Keep in mind that the origin of this protocol is the Constrained RESTful Environments (CoRE) working group that "aims at realizing the REST architecture in a suitable form for the most constrained nodes". "One of the main goals of CoAP is to design a generic web protocol for the special requirements of this constrained environment, especially considering energy, building automation and other M2M applications. The goal of CoAP is not to blindly compress HTTP [RFC2616], but rather to realize a subset of REST common with HTTP but optimized for M2M applications. Although CoAP could be used for compressing simple HTTP interfaces, it more importantly also offers features for M2M such as built-in discovery, multicast support and asynchronous message exchanges." Even if this might look a bit artificial there is a real advantage of this HTTP/CoAP methods mapping that is to recycle many HTTP technologies for the benefit of the M2M applications. Using CoAP in this context enables to take advantage of a low overhead and a simple protocol to implement the business logic.

One feature inherited from HTTP is the property of idempotentency, which means that a function invoked multiple times will have the same effect. This makes sense for an HTTP server but is very unlikely when sent to an M2M device which main justification is to host a sensor or actuator. For this reason the POST method that is not idempotent has to be preferred to GET that is idempotent.

4.1.3.4 Security

CoAP relies on Datagram Transport Layer Security (DTLS) that is an adapted TLS for UDP. The scheme used is no more the “coap:” but “coaps:” and security can be achieved either with symmetric pre-shared keys, asymmetric keys without certificate or asymmetric keys with a certificate. With TLS there is a handshake to create a shared secret to enable to generate a session key. Drawback is that this is not applicable to multicast communication. An alternative to DTLS is to use the Encapsulating Security Payload (ESC) of IPsec to protect the communication. This is an option but still, this is not applicable to multicast communication and could be demanding particularly on the server side because of the key replacement mechanism managed by the Internet Key Exchange Protocol (IKE) within IPsec.

4.1.3.5 Message format

The detailed format is available in the IETF document. A message has a header of 4 bytes followed by options and a payload. It is summarized below in a BNF way

```
CoAP Message ::= Header [Options] Payload

Header ::= Version Message-type Option-Count Message-code
Message-ID

Version ::= 2-bit unsigned integer value

Message-type ::= CON || ACK|| RST || NON

Option-Count ::= 4-bit unsigned integer value

Message-Code ::= 8-bit unsigned integer value

Message-ID ::= 16-bit unsigned integer value
```

Role of the various options are summarized in the following Table 4-1:

Options	Role	Comment
Token	Management of the request/answer matching	
Uri-Host, Uri-Port, Uri-Path, Uri-Query	Specification of the target resource	
Proxy-Uri	Absolute URI that enables to make a request to a proxy. It is the role of the proxy to forward the request	
Content-type	Specifies the format of the payload	

Accept	Indicates the acceptable media type like the classical text/plain. This is the way to create or parse a payload.	CoAP recommends to request new Internet media Type from IANA that are relevant for M2M
Max-Age	Indicates the maximum time a data can be cached	Related to caching mechanism
ETag	Provides the current value of the "entity-tag"	Related to caching mechanism
Location-Path, Location Query	Indicates the location of a resource as an absolute path. Similar to Uri-Path and Uri-Query	Used for new location created with the POST method
If-Match	Enables to make a request conditional of the current existence or value of an Etag	Related to resource update, protects against accidental overwrite
If-None-Match	Enables to make a request conditional on the non-existence of the target resource	

Table 4-1: Options description

There is an interesting feature specified by CoAP that differentiates request to an origin server and request made through a proxy. The document describes a proxy as "a CoAP end-point that can be tasked by CoAP clients to perform requests on their behalf". This feature could be recycled the other way around to factorize requests to be sent to all devices that are part of a capillary network. This feature is activated by specifying the request URI with a specific "Proxy-Uri" option.

4.2 Device Management Protocols

Based on the outcome of section 4.1, existing standardized DM solutions, TR-069 and OMA-DM v1.x, do not meet EXALTED high level objectives, thus are not appropriate for remote management of M2M constrained devices. OMA has set up new work items to standardize upcoming device management solutions, OMA DM NG and OMA DM Lightweight M2M. These new solutions are not available yet, furthermore they are not backward compatible with the widely deployed OMA DM v1.x specifications.

EXALTED proposes two lightweight device management solutions:

1. Exalted Lightweight For OMA-DM v1.x (ELFOMA)
   This solution is recommended for operators who wish to reuse their existing OMA-DM v1.x servers to support new M2M constrained devices
2. CoAP based DM protocol
   This solution is recommended for operators who wish to manage M2M devices without reusing existing OMA-DM servers.

4.2.1 Exalted Lightweight For OMA DM v1.x (ELFOMA)

According to statement made by OMA at Mobile World Congress in Barcelona, February 2012, the total number of mobile devices deploying OMA DM specifications reaches 1.4 billion. This novel solution is therefore dedicated to network operators and service providers who wish to save cost by reusing their existing OMA DM v1.x servers in order to continue managing existing mobile devices and to incrementally manage new M2M devices.

Key points of this solution are:

- Compatible with OMA-DM v1.x servers

- Compatible with existing OMA DM Management Objects and enablers such as FUMO, SCOMO, DiagMO, GwMO, LaWMO, etc
- Low message encoding complexity
- Message payload reduction by 85% (average), compared to SyncML-based messages.

This solution addresses the following EXALTED high level objectives:

- Energy efficiency. Message encoding complexity and message transmission time are both reduced, hence power consumption is minimized.
- Device Cost. The new message encoding scheme requires less memory and processing capability, the cost of the device is consequently reduced.
- Spectrum efficiency. As size of messages is reduced by 85% the bandwidth usage is optimized.

Regarding the scalability aspect, this solution shares the same drawback as OMA-DM v1.x, as DM sessions are stateful, DM servers have to keep track of sessions. However, the scalability is better managed by hosting DM servers on cloud platforms. In addition, GwMO [39] enabler can be used to broadcast/multicast DM commands to M2M devices located behind gateways. Broadcasting and multicasting processes make use of Fanout MO as defined in GwMO [39]. DM command sent to a M2M gateway is then fanned out to devices belonging to the targeted device group. Results or status posted by devices are collected by the M2M gateway. The aggregated data is then sent at once to the DM server. This application layer broadcasting and multicasting mechanism reduces the traffic between the M2M gateway and the DM server, it therefore contributes to increase the scalability of the solution.

4.2.1.1 ELFOMA System description

The system depicted in Figure 4-1 is reusing existing OMA-DM v1.x server (a) as is, to support both the former OMA-DM v1.x enabled mobile devices (b) and the new constrained M2M devices (c). An OMA-DM adapter proxy (d) is introduced between constrained M2M devices (c) and the server (a). Server originated OMA-DM protocol v1.x messages are converted by the proxy (d) onto ELFOMA, Lightweight messages for constrained devices. On the other way, ELFOMA are converted back to OMA-DM protocol v1.x before being redirected to the server.

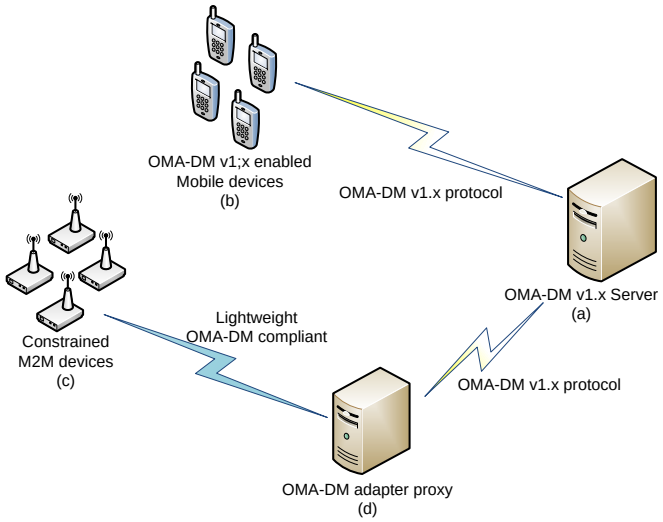


Figure 4-1: ELFOMA system concept

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

- All OMA-DM Management Objects are reused as is

- OMA-DM Setup phase and Management phase remain unchanged
- No protocol simplification
- Fully compliant with SyncML security framework

The key traction of this solution is to reuse existing OMA-DM v1.x servers “as is” to incrementally support new constrained M2M devices. The management of existing fleet of mobile devices is not affected. The investment is limited to the OMA-DM adapter.
The definition of Lightweight OMA-DM compliant messages yields further cost savings. The size reduction of message payload enables communication cost savings as the transmission time is minimized. It also reduces the risk of message retransmissions due to potential loss of network connectivity. In addition, ELFOMA messages have positive impacts on the device cost and design, as the required resources (e.g. amount of memory, processing power and power consumption) are mostly constrained.
ELFOMA are fully OMA-DM compliant messages, it contain the same information as the formal OMA-DM message. This solution has a message encoding scheme which maps formal OMA-DM messages onto lightweight messages. This transformation is applied by the OMA-DM adapter proxy server.

4.2.1.2 Security consideration

The proposed concept is fully compliant with SyncML security framework:

- server and client mutual authentication
- authentication challenge using nonce
- messages integrity check using server nonce, client nonce and hashed secret
- protection against replay attacks using MessageID
- encrypted data can be included in OMA-DM package

In addition, Transport Layer Security (TLS) can be used to exchange DM messages securely, e.g. over HTTPS.

4.2.1.3 Existing Payload Reduction Methods

OMA-DM v1.x messages are SyncML (Synchronization Markup Language) documents, which are based upon standardized XML communication and thus can be implemented on any platform. However the message payloads are verbose. Additionally, potential long URI and tree-based data model contribute to further inflate the size of payloads. To achieve the payload reduction, OMA-DM adapter proxy can compress SyncML based messages to binary based messages.
A common compression method consists in converting XML documents onto a binary representation format. The conversion is reversible, namely, the binary representation can be converted back to XML document without loss of information. Fast Infoset [40], EXI [41] and WBXML [42] specifications aim to optimize both the document size and processing performance:
Fast Infoset (or FI) [40] 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) [41] 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.
Wap Binary XML (WBXML) [42] 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) in a compact manner over mobile networks. WBXML has also been proposed as an addition to the W3C WAP.
Other XML compression techniques and studies have been analyzed by various papers [43][44][45][46].

All above methods can be leveraged in OMA-DM adapter proxy to handle the OMA-DM packages conversion.

4.2.1.4 ELFOMA Payload Reduction Method

While the aforementioned XML compression methods in section 4.2.1.3 can be applied to SyncML-based messages, ELFOMA encoding aims to further optimize both the OMA-DM v1.x message payload size and 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 Payload Reduction method is defined as follow:

1. ELFOMA message content is plain text format. Further compression can therefore be applied on top if needed, e.g. HTTP compression.
2. The SyncML Representation protocol [47] specifies Content Model for each SyncML element type. OMA-DM messages are then serialized to SyncML based on DTD which reflects the specified 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 4-2 depicts the concept.

   For instance, Figure 4-3 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 OMA-DM messages. Figure 4-4 is an excerpt of OMA-DM payload highlighting the SyncHdr element type. Note that 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 one child elements are represented in 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 4-5 shows the ELFOMA syntax for SyncHdr element type, note that all child elements defined by the content model in Figure 4-3 are taken into account. This replication is necessary to achieve ELFOMA full compatibility with OMA-DM.

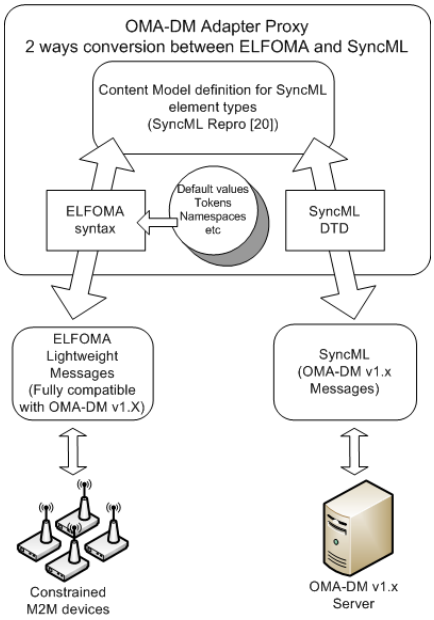


Figure 4-2: OMA-DM adapter proxy concept

```
SyncML Content Model for SyncHdr:
(VerDTD, VerProto, SessionID, MsgID, Target, Source, RespURI?, NoResp?, Cred?, Meta?)
```

Figure 4-3: Content Model for SyncHdr

```
<SyncHdr>
   <VerDTD>1.2</VerDTD>
   <VerProto>DM/1.2</VerProto>
   <SessionID>1</SessionID>
   <MsgID>2</MsgID>
   <Target>
      <LocURI>
         http://www.syncml.org/mgmt-server
      </LocURI>
   </Target>
   <Source>
      <LocURI>IMEI:493005100592800</LocURI>
   </Source>
   <Cred>
      <Meta>
         <Type xmlns='syncml:metinf'>
            syncml:auth-basic
         </Type>
         <Format xmlns='syncml:metinf'>
            b64
         </Format>
      </Meta>
      <Data>QnJ1Y2UyOk9oQmVoYXZl</Data>
   </Cred>
   <Meta>
      <MaxMsgSize xmlns='syncml:metinf'>
         5000
      </MaxMsgSize>
   </Meta>
</SyncHdr>
```

Figure 4-4: OMA-DM payload – SyncHdr sample – 519 bytes

```
ELFOMA syntax for SyncHdr element:
SyncHdr=VerDTD;VerProto;SessionID;MsgID;Target;Source;RespURI?;NoResp?;Cred?;Meta?

ELFOMA syntax for Source element:
Source=LocURI;LocName?

ELFOMA syntax for Meta element:
Meta=Type?;Format?;#PCDATA

ELFOMA syntax for Cred element:
Cred=Meta?;Data
Cred={Type?;Format?;#PCDATA};Data
```

Figure 4-5: Element mapping to DSL model

```
SyncHdr=1.2;DM/1.2;1;2;http://www/syncml.org/mgmt-server;IMEI:493005100592800;;;{{syncml:auth-
basic;b64};QnJ1Y2UyOk9oQmVoYXZl};{;;MaxMsgSize=5000}
```

Figure 4-6: DSL – SyncHdr sample – 146 bytes

3. Values of child elements (e.g. VerDTD, SessionID, MsgID…) are directly placed into CSV fields. Figure 4-6 shows how the OMA-DM payload (Figure 4-4) is translated to ELFOMA using the syntax defined in Figure 4-5. As the actual values are mapped to the dedicated fields, there is no need to specify the name of the child elements (VerDTD, VerProto, SessionID…). This CSV mapping approach reduces payload size. With respect to the SyncHdr example, the lightweight payload size is, for now, 3,55 times smaller (146 vs 519 bytes) than its counterpart OMA-DM payload. This lightweight solution is processing efficient, namely, low parsing complexity and unlike XML compression techniques

described in section 4.2.1.3, special encoding/compression is not required to generate lightweight messages.

4. A CSV field could be left empty if no value is assigned to the element type. In Figure 4-6, the RespURI and NoResp fields are left empty, as they are not specified. When converting ELFOMA back to SyncML message, optional child element types being left blank do not appear in the SyncML message. However, OMA-DM adapter proxy could assign 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 4-5, 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 Meta field from the next element, Data. Figure 4-6 shows the use of curly brackets to map the Cred element values. Note that the content model of 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 SyncHdr are considered part of the SyncBody.
   When converting ELFOMA back to OMA-DM v1.x, OMA-DM adapter proxy automatically inserts the “SyncML” element with the proper value (e.g. Syncml:Syncml1.2). Likewise, “SyncBody” element is automatically inserted by the proxy.
8. Namespaces are essentially used in Meta elements to define custom variables. Constrained devices are likely to only deal with variables and values (key/value), namespace may be dropped. For this reason, the mapping of namespaces is handled by OMA-DM adapter proxy, and variable name collision is avoided. Namespace definition (xmlns=’syncml:metinf’) has therefore been discarded in ELFOMA, as shown in Figure 4-6. When converting ELFOMA back to SyncML messages, namespaces will be inserted based on the namespace mapping tables kept by OMA-DM adapter.
9. Curly brackets “{” “}” are used to specify a list of child elements of the same type (e.g. Item+). Figure 4-7 depicts an excerpt of OMA-DM message with a Replace command applying to 5 variables. The counterpart Lightweight version is shown in Figure 4-8.
10. Square brackets “[“ “]:” can be used to factorize common elements. Fields to be factorized is positioned using CSV syntax. For instance, the following 2-fields expression:
    {./DevInfo/DevID;soft/v1}{./DevInfo/Man;soft/v2}
    Can be factorized as follow:
    [./DevInfo/;soft]:{DevID;v1}{Man;v2}

```
<Replace>
 <CmdID>3</CmdID>
 <Item>
  <Source>
   <LocURI>./DevInfo/DevId</LocURI>
  </Source>
  <Meta>
  <Format xmlns='syncml:metinf'>Chr</Format>
  <Type xmlns='syncml:metinf'>  text/plain</Type>
  </Meta>
  <Data>IMEI:493005100592800</Data>
 </Item>
 <Item>
  <Source>
   <LocURI>./DevInfo/Man</LocURI>
  </Source>
  <Meta>
  <Format xmlns='syncml:metinf'>chr</Format>
  <Type xmlns='syncml:metinf'>text/plain</Type>
  </Meta>
  <Data>Device Factory, Inc.</Data>
 </Item>
 <Item>
  <Source>
   <LocURI>./DevInfo/Mod</LocURI>
```

```
    </Source>
    <Meta>
    <Format xmlns='syncml:metinf'>chr</Format>
    <Type xmlns='syncml:metinf'>text/plain</Type>
    </Meta>
    <Data>SmartPhone2000</Data>
  </Item>
  <Item>
    <Source>
      <LocURI>./DevInfo/DmV</LocURI>
    </Source>
    <Meta>
    <Format xmlns='syncml:metinf'>chr</Format>
    <Type xmlns='syncml:metinf'>text/plain</Type>
    </Meta>
    <Data>1.0.0.1</Data>
  </Item>
  <Item>
    <Source>
      <LocURI>./DevInfo/Lang</LocURI>
    </Source>
    <Meta>
    <Format xmlns='syncml:metinf'>chr</Format>
    <Type xmlns='syncml:metinf'>text/plain</Type>
    </Meta>
    <Data>en-US</Data>
  </Item>
</Replace>
```

Figure 4-7: OMA-DM payload –Replace commands – 1170 bytes

```
Content Model for Replace:
Replace=CmdId;NoResp?;Cred?;Meta?;Item+
Replace=CmdId;NoResp?;Cred?;Meta?;{Item}{Item}…

Content Model for Item:
Item=Target?;Source?;SourceParent?;TargetParent?;Meta?;Data?

ELFOMA payload as translated from Figure 4-7:
Replace=3;;;;{[;./DevInfo/;;;{chr;text/plain};]:
{;DevId;;;;"IMEI:493005100592800"}
{;Man;;;;Device Factory, Inc.}
{;Mod;;;;SmartPhone2000}
{;DmV;;;;1.0.0.1}
{;Lang;;;;en-US}}
```

Figure 4-8: ELFOMA payload –Replace commands – 180 bytes

11. Nested child commands can be specified using curly brackets. High level commands such as Atomic, Sequence can include several commands. The Atomic command below contains 1 Replace and 2 Exec commands:

    Atomic=CmdID;NoResp?;Meta?;
    {Replace=CId2;nores?;cred?;meta?;{item1}{item2}}
    {Exec=CId2;nores?;cred?;meta?;correlator?;item}
    {Exec=Cid3;nores?;cred?;meta?;correlator?;item}

12. The extended Meta element type has "Type" and "Format" which can be respectively set by default to "chr" and "text/plain" by OMA adapter proxy. Hence the lightweight payload in Figure 4-8 is further reduced to 154 bytes :

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

Figure 4-9: ELFOMA payload- Replace command

13. If Data element type value contains special characters pertaining to ELFOMA syntax

domain, i.e. ‘;’ ‘{‘ ‘}’ ‘[‘ ‘]:’ ‘?’, then the value shall be quoted. E.g. ”IMEI:493005100592800”

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

The described lightweight encoding rules can be further optimized as follow:

15. 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 [47]. The “Replace” protocol command element in Figure 4-8 can be replaced by token 20. The “SyncHdr” message container element in Figure 4-5 can be replaced by token 2C.
    Likewise, token codes can be used to specify the SyncML command (Cmd) referenced by a Status element type.
16. 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. OMA adapter proxy can handle a mapping between tree-based data model and flat data model. The use of 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 flat data variables. Node codes are specified in the Source or Target element type 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 backward compatibility issue.
    URI referenced by RespURI element type or by Target or Source element type within SyncHdr can be long. OMA-DM adapter proxy can handle a mapping between the original long URI and a URI short code or alias.
    OMA-DM Adapter proxy must be provisioned with node codes and URI short codes mapping. The original tree based data and long URI will be used as is, if no mapping is provided in the proxy.
17. Values for Type and Format elements e.g. “chr;text/plain”, “auth-basic;b64” can be mapped to tokens. This contributes to reduce the payload pertaining to the Meta element type. In Figure 4-6, the Type “auth-basic” has been replaced by an 8-bits token (1), the Format “b64” has also been replaced by token. By default, the pair of Type/Format tokens (0;0) represents (chr; text/plain).
    In rule #8, namespaces are removed. They are taken into account to define tokens. Therefore an element ‘Type’ belonging to namespace Syncml will not share the same token as an element ‘Type’ of a different namespace.

Applying the aforementioned optimizations 15, 16 & 17, Lightweight payload of Figure 4-6 and Figure 4-8 are further reduced as shown in Figure 4-12 and Figure 4-13 below. The source element type value in Figure 4-6 “http://www/syncml.org/mgmt-server” has been replaced by an alias “mgmt-server” in Figure 4-12. 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 4-4.
In Figure 4-13, the Replace command element in Figure 4-8 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 4-8 is no longer needed, 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 rule #12). The payload size in Figure 4-8, 180 bytes, has been reduced to 122 bytes. The size of the OMA-DM counterpart

payload, Figure 4-7, was 1170 bytes, 9.6 times larger.

```
2C=1.2;DM/1.2;1;2;mgmt-server;"IMEI:493005100592800";
;;{{1;1};QnJ1Y2UyOk9oQmVoYXZl};{;;1D=5000}
```

Figure 4-10: Optimization applied to payload in Figure 4-6 – 95 bytes

```
20=3;;;;{;2A;;;;"IMEI:493005100592800"}
{;2B;;;;Device Factory, Inc.}
{;2C;;;;SmartPhone2000}
{;2D;;;;1.0.0.1}
{;2E;;;;en-US}}
```

Figure 4-11: Optimization applied to payload in Figure 4-8 – 122 bytes

18. 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 4-2 has to be extended to handle two different Content Models, as shown in Figure 4-14.

    Conversion between ELFOMA elements Content models and SyncML elements content models is handled by the OMA-DM adapter proxy. For instance,
    SyncML Content Model for Item element type:
    Item: (Target?, Source?, SourceParent?, TargetParent?, Meta?, Data?)
    is mapped to ELFOMA Content Model:
    Item: (Target?,Source?,Data?,Meta?, SourceParent?, TargetParent?)

    SyncML Content Model for Replace command element type:
    Replace: (CmdID, NoResp?, Cred?, Meta?, Item+)
    is mapped to ELFOMA Content Model:
    Replace: (CmdID, Item+, Meta?, Cred?, NoResp?)

    The exception to handle the Meta content model differently, as described in rule #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

Applying this reordering scheme, Payloads shown in Figure 4-10 and Figure 4-11 have been further reduced as shown in Figure 4-13 and Figure 4-14. The size of replace command is now 11.25 times smaller that it's OMA-DM counterpart payload, 104 bytes vs. 1170 bytes.

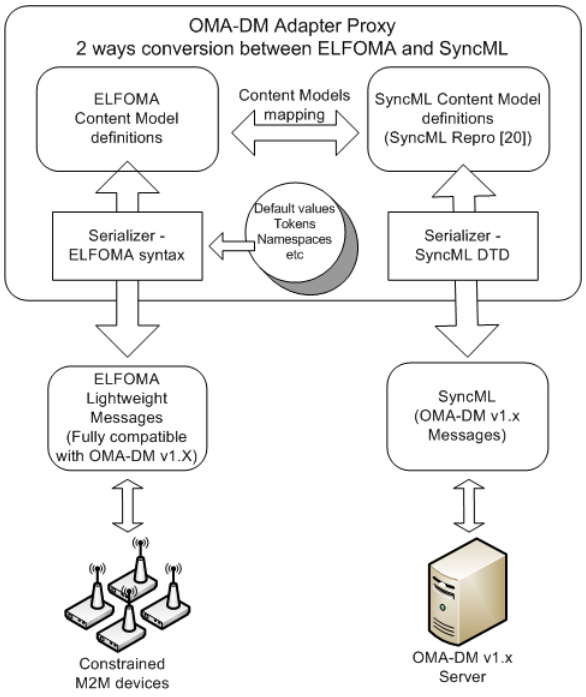


Figure 4-12: Optimized OMA-DM Adapter concept

```
2C=1.2;DM/1.2;1;2;mgmt-server;"IMEI:493005100592800";{{1;1};QnJ1Y2UyOk9oQmVoYXZl};{;;1D=5000}
```

Figure 4-13: Optimization applied to payload in Figure 4-10 – 93 bytes

```
20=3;{;2A;"IMEI:493005100592800"}
{;2B;Device Factory, Inc.}
{;2C;SmartPhone2000}
{;2D;1.0.0.1}
{;2E;en-US}}
```

Figure 4-14: Optimization applied to payload in Figure 4-11 – 104 bytes

4.2.1.5 Achievement

The assessment of payload verbosity is performed using a firmware update scenario leveraging FUMO enabler. This scenario is updating 3 firmware's (e.g. GSM module, GPS, NFC chipset) at once. Atomic command is used to execute these updates as a set. Although this scenario does not reflect characteristics of constrained device, its complexity involving 10 messages should provide a good foundation for the assessment. This scenario includes the follow aspects: client and server mutual authentication, alignment of firmware versions, firmware download and update process, firmware update status report. Refer to Figure 4-15.

OMA-DM v1.2 payloads are firstly generated for the 10 messages of the scenario. They are converted to ELFOMA messages using directives listed in 4.2.1.4.

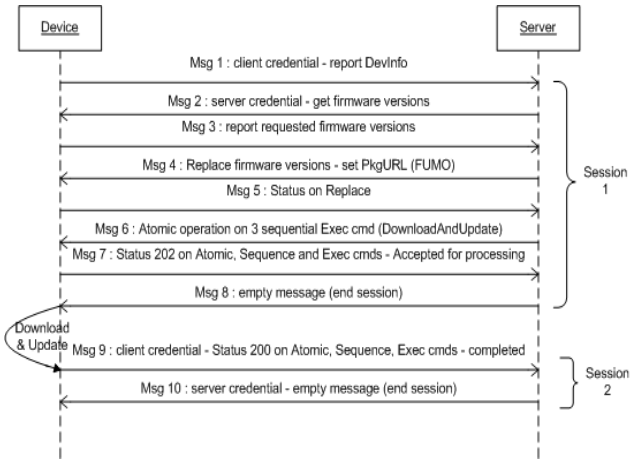


Figure 4-15: Scenario for Payload size assessment

In order to benchmark the proposed method against EXI and WBXML, OMA-DM payloads are then encoded with EXIfficient [48] an open source implementation of the W3C Efficient XML Interchange format specification, and with kXML2 [49]. JSON and GZIP are also included in the benchmark. Figure 4-16 depicts the payload size of each message. The compactness of different encoding is shown in Figure 4-17. The total ELFOMA payload size (message 1 through 10) represents only 15.2% (1954 bytes) of the cumulated SyncML payload weight (12858 bytes). ELFOMA over-performs EXI, WBXML and GZIP: the average compactness of EXI schema-less encoding is 38%, EXI schema-informed mode achieves 27.1% compactness. WBXML yields less compact payloads: 65.5% of SyncML size. GZIP reaches 34.5% average compactness. To enable EXI schema-informed mode, the SyncML DTD, as defined in SyncML Repro [47], 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, it occupies more space in memory than payloads.

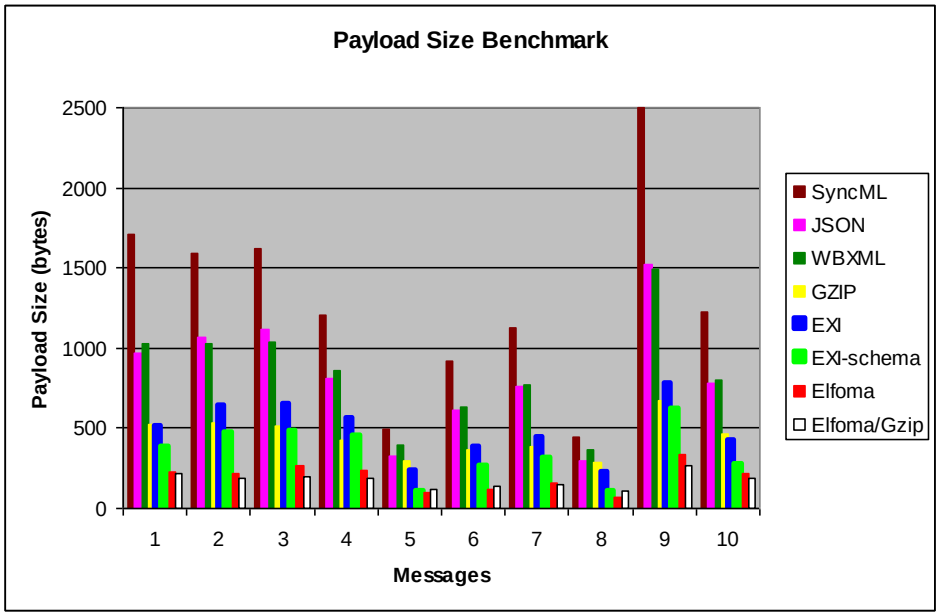


Figure 4-16: Payload size benchmark

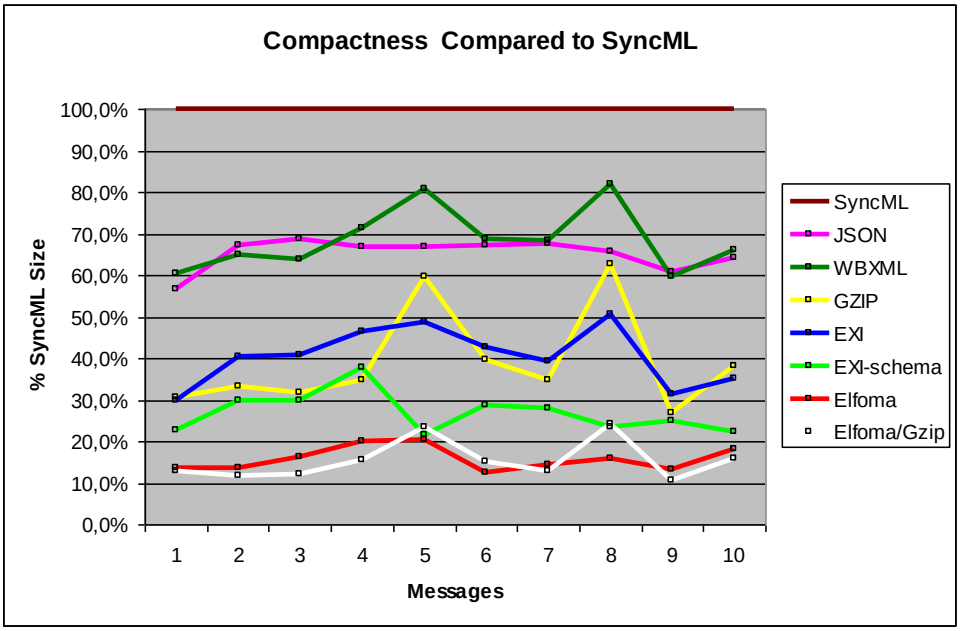


Figure 4-17: Payload Compactness compared to SyncML

ELFOMA compactness ranges from 12.6% to 20.4%. As it is plain text, GZIP can be applied on top to achieve 13.7% compactness. Note that GZIP is not performing well on small payload (less than 110 bytes), in this case the resulting GZIP size exceeds the uncompressed size.
The compactness of EXI schema-less ranges from 30% to 50.6% and EXI schema-informed ranges from 21.6% to 37.6%. The results are conformed to compactness results as stated in EXI evaluation [50].

4.2.2 CoAP Device Management

4.2.2.1 Over UDP

The general aim of using CoAP for DM is to avoid verbose messages and to execute DM operation more efficiently. To enable execution of DM procedures, OMA-DM protocol utilize client on the device that provides interfaces specific to a device management functionality defined in OMA DM specifications. The client is responsible for connecting to remote DM servers, authentication and parsing the protocol messages. Management Client executes itemized commands and processes the statuses returned from the execution of such commands on the connected DM server.

Actions such as firmware and software update can be accomplished by sending compressed executable code modules that enables updates on all system levels for simple sensor devices and reduces energy consumption and dissemination time. The CoAP protocol can be utilized for DM operations due to available multicast and unicast requests, header option fields and the simple congestion control mechanism that are applicable for the remote update, monitoring and the diagnostic of the devices. These simple DM operations can be achieved by sending unicast/multicast request to a device/group of devices that needs to be managed.

This subsection describes Device management module based on REST for simple session-less procedures over the CoAP protocol.

Running such services on constrained device consumes energy and resources and can influence device and network lifetime.

OMA DM firmware/software update job can be run using:

(1) Immediate download and update initiated out of band: Exec command is issued on the DownloadAndUpdate node in the tree. This action starts the download after which update agent is invoked. Generic message is sent to the DM server when update is finished.

(2) Immediate download and update initiated in band: during DM session package data is sent to the device and Exec command is issued on the Update node in the tree. Only the update is invoked with this action. Generic message is sent to the DM server when update is finished

(3) Out of band download, followed by an eventual update: This scenario requires two Exec commands to be run:

   i. Package data is downloaded after Exec command to download node in the tree. After completed download Generic message is sent to DM server

   ii. Device manager server issues an Exec command on the Update node in the tree to update the previously downloaded package. Alert message is also sent to DM server after completed update.

All above methods requires bootstrap, device info exchange, and constant sessions that drain device memory, CPU and battery. In the following sections device management over CoAP using simple device management and GZIP to transfer device update package is described.

4.2.2.1.1 CoAP Management Object (MO)

In OMA DM, all management objects are grouped into the management tree, where each node in the tree corresponds to the service or the application and can be addressed with an URI. Constrained devices services are written as a small executable codes called modules.

Instead of using client on the device to disseminate procedures on device management object tree, modules are replaced directly on the device and all module version information is kept in text file on the device. Management object for CoAP DM have different meaning and it is used in different context than in OMA-DM, i.e. it contains only “passive” information about device. This information includes device firmware/software version and other possibly required data depending on a deployment (e.g. if device have more than one application, each application can be described: module version, name, etc.).

The package version must be incremented on each update and synchronized between the version on the device and the device management server. Accept package version, package name, description as well as module details can be contained in this file. In the Table 4-2 below FUMO OMA-DM methods for accessing these elements and executing command on tree node are compared with equivalent CoAP device management actions. In this manner SCOMO can be also employed using CoAP.

Entity Name	OMA-DM Description	OMA-DM Access	CoAP Description	CoAP Access
PkgName	This optional node specifies the Name associated with the firmware update package.	Get	GET request is sent to obtain CoAP MO file that contains package name	GET
PkgVersion	This optional node specifies the Version information for the firmware update package. The version information is device manufacture specific and can contain any data.	Get	GET request is sent to obtain CoAP MO file that contains package name	GET
Download	This interior node is the target of an ‘Exec’ command in order to initiate a firmware download for the specified update package and is optional. This optional node specifies the Name associated with the firmware update package.	Get, Exec	CoAP replaces directly package with new version using PUT request	PUT
Download/PkgURL	This node specifies the URL where the	Get, Replace	Removed. This is not	/

	firmware update package or download descriptor is located.		mandatory since update is employed directly	
Update	This interior node is a target of an 'Exec' command in order to initiate a firmware update for the specified update package and is optional.	Exec, Ger	CoAP replaces directly package with new version using PUT request	PUT
Update/PkgData	This node is the target of a 'Replace' command when DM is used to directly provide the binary firmware update package.	Replace	Removed. This is not mandatory since update is employed directly	/
DownloadAndUpdate	This interior node is the target of an 'Exec' command invoked to initiate a firmware download and update for the specified update package and is optional. The update MUST take place as soon as practical after download.	Exec, Get	CoAP replaces directly package with new version using PUT request	PUT
DownloadAndUpdate/ PkgURL	This node specifies the URL where the firmware update package or download descriptor is located, that is to be downloaded and installed at the next practical opportunity.	Get, Replace	Removed. This is not mandatory since update is employed directly	/
State	This should not be used with CoAP device management as it requires implementing additional interface for indicating the current state of the device with respect to firmware update status.	/	The state of invoked command is returned as ACK message	State return using Confirmable messages (Update Successful / Failed)

Ext	Node for supporting vendor specific extensions.	Get	This is optional field and can be inserted inside CoAP MO file	/

Table 4-2: OMA-DM vs CoAP DM for FUMO object

Firmware and Software update management interface
CoAP device management jobs are executed using PUT request to replace device package version. The update package is compressed using GZIP and transferred over the air. The package version is incremented on each update and the version number is saved in the package definition file.

The update process presented in the diagram displays device management over the CoAP:

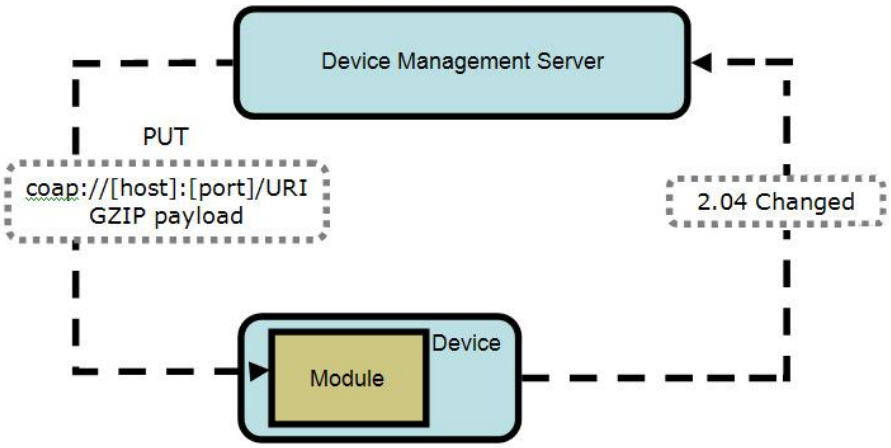


Figure 4-18: Device Management over CoAP

Device management server can send a GET request to obtain the package (module) version, or this step can be avoided if the server has this information. Then, compressed executable code is sent with PUT request and extracted in the destination folder. Extraction and deletion of compressed package requires implementing of simple module that will execute these actions when replace is received on the predefined URI.

Addressing devices

All CoAP requests must be sent to URI-port or to a port number that can be obtained via a discovery operation (by doing a "/.well-known/core" lookup on at least one group member node, or from equivalent service discovery lookup). Lookup enables discovery of a hosted resources.

It is necessary to enable executing services on a certain device group by sending a command from the application (i.e. device management server). Configuration of the network and its hierarchy can be used for grouping, e.g.:

```
URI
coap://[capillary-network]:[port]/all-devices/
```

where URI represents firmware version for all devices in the first capillary network. The “all-devices” node is a group end point that enables communication with all devices within a group by mapping group authorities onto IP multicast address [63].

The authority portion of the URI is used to identify a node (or group) and the resulting URI is bound to a unicast or multicast IP address.

Since the EXALTED is targeting more than one use case, common used words for device management procedures overall can be mapped with a single character token to shorten the URI and thus lower the devices’ overhead. For device management operations, these tokens should be strictly defined for a different function sets to enable mapping of sub-tree with a token:

```
Semantic
coap://[host]:[port]/device(s)/resource

Example1
coap://capillary3/device3/battery

Example 1 with token
coap://capillary3/device3/b
```

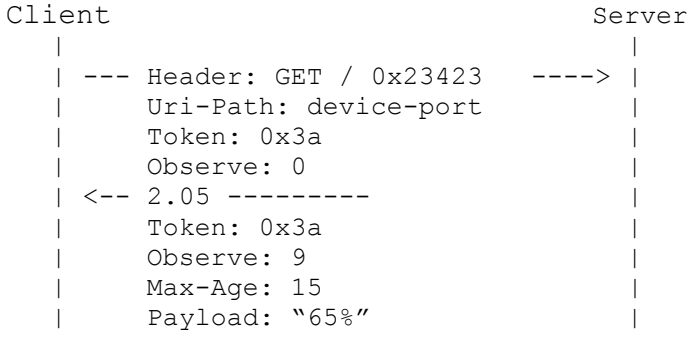
Letter “b” can be used as a token for battery resource in all deployments and all capillary networks.

Remote Monitoring

Remote monitoring procedure can utilize CoAP Observe Option to enable subscribing to device resource change. This option modifies GET method to retrieve a current state of the resource identified by the resource URI (e.g. coap://host:port/path), and adds the resource to the observers list by sending GET with Observe Option = 0.

Remote monitoring operation (e.g. battery level change) can be achieved by subscribing to an observers list on the server that sends GET request to a port or URI that represents a group of devices. For a resource (e.g. battery level), a list of devices which observe resource is maintained.

Subscribing to a resource is presented in a diagram below



By sending a GET request with Observer Option set to “0” the device is added to the list. If resource does not exist or if erroneous, the error response code is returned (4.xx/5.xx) and the device is deleted from the list of the observers.

4.2.2.2 Over SMS

In 3GPP, recent M2M standardization activities lead to investigating an SMS optimization for small data transmission. IETF draft [52] also considering SMS transport for the CoAP protocol, indicating a common interest for the low-cost M2M communication over the existing network infrastructure by employing SMS transport for infrequent small payloads.

In next section the SMS as a potential mean of transferring M2M information is evaluated. Another motivation for employing device management over SMS is the addressability of the devices in the network behind the Gateway. As most devices will not have an IP address; the Gateway can be a contact point with the network outside the premises. Therefore, the M2M device can be manipulated (GET, PUT, POST, DELETE) directly over an SMS, or over the SMS Terminal (Gateway) that can read and forward the request over supported transport protocol (i.e. CoAP in this case).
The main motivation for leveraging an SMS in the M2M context is significant number of M2M devices operated over SMS, a low-cost infrastructure and a small and infrequent traffic transmission per terminal. In order to employ SMS as a CoAP transport - payload size, security, multicast messaging and reliability are considered.

According to 3GPP, in order to leverage SMS over LTE there are following optimizations methods:

- SMS over Signalling Gateway (SG):
    1. removal of control protocol (CP) layer
    2. enhancements that give greater flexibility in the deployment of the MSC functionality
    3. Extension of 2) to support stateless SMS-IWF.
- use of the pre-established NAS security context to transfer the SMS PDUs as NAS signalling without establishment of all the radio bearers or RRC security context

As LTE supports packet based service only, circuit switch (CS) services can be used by employing fall-back (FB) to enable the provisioning of CS-domain services (i.e. SMS in this case). A CS fall-back enables terminal connected to E-UTRAN to use GERAN or UTRAN to connect to CS domain. This function is only available in case E UTRAN coverage is overlapped by either GERAN coverage or UTRAN coverage.
There are problems with CSFB in LTE that can impact this approach for employing CS in LTE (i.e. call set-up latency, possible impact on data application, etc.)

In the Release 11 [51], SMS over SG is replaced with work item SIMTC (System Improvements to Machine-Type Communications). As use of “SMS over IMS” can be regarded as rather heavyweight for low end M2M applications, the SMS in LTE will be most probably supported within SIMTC specification.

In addition the removal of the CP protocol layer does not necessarily imply removal from the protocol stack. Instead, the “removal” can be effectively achieved by enhancing the state machines in the UE and in the network so that the CP-ACK is rarely sent. Removal of the CP layer would save one or two messages on the radio interface.

In this section CoAP over SMS mapping is analysed and implemented and M2M communication over SMS is evaluated for small data transmissions in a live network conditions. The evaluation includes: regular, concatenated and multicast messaging between devices. Note that this evaluation represents the worst case scenario, when there is no agree over certain SLA with Service Provider. The final device management server should employ direct SMS-C capabilities of sending the large number of messages with high speed that will enable efficient device management over SMS.

4.2.2.2.1 SMS Payload size consideration

A conventional SMS message can be encoded by 7-bit, 8-bit and 16-bit encoding:

- 7-bit character encoding allows the maximum of 160 characters per SMS message (140 bytes)
- 8-bit character encoding allows the maximum of 140 characters per SMS message
- 16-bit character encoding allows the maximum of 70 characters per standard SMS message.

According to [52], the currently safest solution is to use 7 bit encoded SMS and common Base64 encoding for CoAP payload. For the 7-bit character encoding, the maximum message size is 140 bytes (1120 bits). If a message is larger than 140 bytes, the SMS is concatenated into many 134-bytes messages that together make one large SMS with up to 39015 characters (255 concatenated SMS messages). Every concatenated SMS payload size is 6 bytes smaller as the UDH header is required for each message. In addition, many Service Providers do not support concatenated SMS, and many mobile devices can only display up to 3 concatenation. Using the 16 bit encoding for embedding CoAP binary messages requires re-encoding of data.

The maximum SMS payload size depends on the selected deployment approach, because different implementations and additional mechanisms like encryption, produces extra overhead.
When the SMS is utilized for transport of the CoAP messages, two options are possible depending on the payload size: each CoAP message fits into one SMS message; or CoAP messages are larger than SMS payload. In the latter case, the CoAP over SMS can be deployed by:

(1) Sending the CoAP message as one SMS that will concatenate a certain number of SMS messages (maximum of 255)
(2) Segmenting a payload into a certain number of SMS messages
(3) Utilizing Blockwise transfer [53] in order to split CoAP message into 140 bytes segments and to map each segment into a SMS.

This research covers and evaluates transmission of the CoAP messages that fit into one SMS message as well as approaches (1) and (2).
In case (1), the first SMS will have approximately 124 bytes payload, 6 bytes User Data Header (UDH) and 10 bytes CoAP header. The following concatenated messages will have 134 bytes for the payload and 6 bytes for the UDH header before reaching the final 255th message. Sending a large number of concatenated SMS messages can take time, e.g. it may take up to two minutes to transfer thirty concatenated text messages [54].

In case (3), every CoAP message is segmented into one SMS with 10 bytes for the CoAP header, thus leaving 130 bytes for the payload. The existing SMS terminals without the adjustments cannot merge CoAP messages segmented with Blockwise transfer, but can read concatenated CoAP SMS messages. Maximum CoAP payload for concatenated messages is 31620 bytes (255 messages); but this strictly relies on the provider's policy and device capabilities. In [55], it is suggested that the Blockwise transfer should not be used for large payloads without further notice.

The presented values for the maximum SMS packet size and the maximum CoAP payload size are given without encryption.

Type of transport	SMS Header Size [bytes]	CoAP Header Size [bytes]	MAX SMS packet size [bytes]
Concatenation	6*n	10	124+134(n-1)
Single packet per SMS	0	10*n	130*n

Table 4-3: CoAP over SMS payload size consideration vs required number of SMS for the given payload

n – the number of SMS required for transporting the complete payload

4.2.2.2.2 Reliability

Although UDP is not consider as a reliable transport layer like TCP, the CoAP implements lightweight reliability mechanisms based on the confirmation of message reception with acknowledge response.
The RESPONSE_TIMEOUT is the time after a message is resent if the acknowledgment was not received. Also, the timeout should be increased exponentially as the number of retransmissions increases and should have the MAX_RETRANSMIT value to limit the number of re-tries. For a new confirmable message, the retransmission counter is set to 0. If the message is retransmitted, the retransmission counter is incremented, and the timeout is doubled. The retransmission stops when the counter reaches the MAX_RETRANSMIT value. This mechanism prevents messages from being resent indefinitely and allows device to control the number of retransmissions. A Basic congestion control is provided by the exponential increase of timeout between two transmissions. Addressing devices in the standby mode can also utilize this time to initiate procedures by sending notification messages and waiting for an answer; in order to resend the message after the RESPONSE_TIMEOUT.
An end device should send a Reset message if the confirmable message was received, but it couldn't process it because some context is missing.

4.2.2.2.3 Multicast messages

Exchanging a large number of CoAP messages over SMS is feasible, but with limitations imposed by SMS Service Centers - SMS-C. These limitations may not have high impacts on devices with long sleeping cycle. On the other hand, they have impacts on M2M applications that require high reactivity from devices (e.g. an application needs to turn light on/off within 5 seconds). If multicast is used, it is recommended to agree over SLA with Service provider and to implement a specific interface for SMS Gateway, in order to enable sending a large number of SMS in a short interval. To multicast a message to a group of devices, the SMS Gateway sends one specific request containing a list of MSISDNs along with the same SMS content for all devices in the group. SMS delivery report/status and device-originated ACK are collected by the SMS-C or SMS Gateway. The overall report/status or device-originated ACK are then aggregated and sent to SMS Gateway via a callback function.

4.2.2.2.4 Security

The SMS security is a challenge - cryptographic protection that offers confidentiality and integrity is not available for SMS messages. Different encryption methods have diverse

memory and power consumption requirements, and it is essential to use solution that will provide the best relation of security and resource consumption for the certain scenario. An authentication mechanism is mandatory in order to avoid communication with unauthorized client/server. Other available SMS security mechanisms: Security Parameter Index (SPI), the Ciphering Key Identifier (KIc) [56], etc… are rarely used in today deployments. It is necessary to employ certain mechanisms to provide minimum protection against spamming, DoS attacks and flooding. There is proposal to encrypt the payload with Symmetric Key encryption [57]; but this may introduce the security risk because it is not safe to send a new secret over the air that can be exposed to a third party.

4.2.2.2.5 Payload size approach with size reduction

Short Message Service as it is, does not provide any security. In this section mechanisms for enhancing payload size and security are proposed for regular and multicast transmissions over SMS.

The following approach for SMS security on device-application level that also reduces payload size is proposed:

(1) Each device can have an initial secret which is known by the devices itself (e.g. provisioned by device manufacturing process), and optionally a group device secret.
(2) The signature is calculated as follows: signature = H(DeviceID+secret+Base64(CoAP_payload)), where H is MD5 or SHA1 hashing algorithm. Therefore, the signature is not always the same (prevents duplicated/stolen signature and replay attack) and it provides a content integrity check.
(3) Sending device calculates the signature (2) and put it in the CoAP header and hash it again using signature+((base64(Huffmancode (CoAP_payload)))
(4) Upon receiving the full CoAP message, the device applies the same signature calculation as (2).

Message is authenticated and accepted if signatures (3) & (4) matches. The message signature can be calculated using MD5 or SHA1 hash. As we aim to reduce the payload size, we are proposing MD5 as it fixed-length hash value is always shorter than SHA1.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
        <name>Wasp Mote Libelium</name>
        <imei>3516560216681931</imei>
        <SensorData>
        <humidity>1225</ humidity >
        <temperature>30.5</temperature >
        <CO>312</CO>
</ SensorData >
```

Figure 4-19: The reference payload

The proposal is to generate message signature using MD5 hash value of device secret ID (random 13 numbers keyword) and payload (Figure 4-19). Payload is compressed using Huffman code, and then encoded with base64.

```
e65263108b67a325bb000c1ed07deabe
```

Figure 4-20: The example of the message hash calculated from the payload and device secret Id

Jmx0Oz94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iSVNPLTg4NTktMSI/Jmd0OyZsdDtuYW1lJmd0O1dhc3AgTW90ZUxpYmVsaXVtJmx0Oy9uYW1lJmd0OyZsdDtpbWVpJmd0OzM1MTY1NjAyMTY2ODE5MzEmbHQ7L2ltZWkmZ3Q7Jmx0O1NlbnNvckRhdGEmZ3Q7Jmx0O2h1bWlkaXR5Jmd0OzEyMjUmbHQ7LyBodW1pZGl0eSAmZ3Q7Jmx0O3RlbXBlcmF0dXJlJmd0OzMwLjUmbHQ7L3RlbXBlcmF0dXJlICZndDsmbHQ7Q08mZ3Q7MzEyJmx0Oy9DTyZndDsmbHQ7LyBTZW5zb3JEYXRhICZndDs=e65263108b67a325bb000c1ed07deabe

Figure 4-21: Base64 encoded payload with signature

The resulting payload (**Figure 4-21**) is 21.93 and 27.7 percent smaller than the native and signed payload respectively.

Ojx0aWQWb11mZ1tjYisRIA4QA/6VWjrEhOrTCAEtNx77goKBA33//YYJ6vSComKDeEk5ilPdo5xQ9+0b4GVgcIXy4OPbky2YtTSwBZbAsXN6L3P6Ofy8y9z+ffvdXNvYOOFlsCxamlg97LcwgzT5BuTFqaWAFF8oWs6RyJpba2uYPVtiTu7WdI5C0yqmEhu+02PFIkx2zP31rcweyG77TY8UiTHatMqothfMqZ/LbYPb5lTUlVTB6tDbPwZj38UiJaZPe65263108b67a325bb000c1ed07deabe

Figure 4-22: The payload after Huffman and Base64 encoding with signature

Sending small encrypted payload less than one SMS can result in actual increasing payload size as message signature itself can be longer than a payload. Very often M2M devices are simple sensors with limited battery, memory and processing capabilities, being grouped in the network behind the Gateway. Encrypting/decrypting payload can consume a lot of device resources. M2M Gateway can provide these functionalities and further forward CRUD (Create/Read/Update/Delete) requests over CoAP data packets acting as a proxy.

The payload size comparison before and after compression and encryption is given in the Table 4-4:

SMS	1 SMS	3 SMS
Reference payload size (base64)	16b	374b
Payload size with signature	48b	404b
Reference payload size after Huffman compression	12b	195b
Payload size compressed with Huffman and Base64 encoded with signature	48b	292b
Compression compare to native size [%]	-66.7%	21.93%
Compression compare to secure payload [%]	0%	27.7%

Table 4-4: The payload size comparison before and after compression and signing

The payload hybrid security approach shows efficiency only on larger number of SMS messages (3 or more). General problem of M2M security is its complexity, because often the many bytes are transferred only to secure a few bytes payload. Sending small encrypted payload can result in actual increasing payload size, because signature can be longer than a payload itself. Nevertheless, as each 7-bit encoded SMS is 160 characters long, the

compression ratio is considered from this length perspective. This means that for smaller payloads (one SMS) there is no compression gain and no loss as well. But for longer and more verbose payloads then one SMS, compression ratio of proposed mechanism is positive compared to a native payload size.

In the case of the M2M communication over SMS payload size can vary from few tens to few hundred of bytes depending on the scenario. If device sends SMS messages with payload by utilizing proposed mechanism, the other devices that receives payload can authenticate the sender. In addition, as stated earlier, for smaller payloads there is no compression gain, but as one 7-bit encoded short message is 160 characters long, there is no loss of compression as well.

Each phone number (MSISDN) in a network has limited number of SMS messages that can be sent in a certain time period. This limit can be increased by agreeing over SLA with Service Provider that will enable sender to achieve higher SMS traffic rate in a short intervals. To efficiently utilize this high traffic rates, it is suggested to employ the interface between the SMSC (Short Message Service Center) and the SMS gateway. This interface is not part of the GSM standard and connections are very loosely protected since the content is delivered not encrypted. The authentication of the gateway is done using header that contains login and password in plain text format. In addition, many SMSC protocols identifies sender using specific field of the short message.

Therefore, MSISDN can be easily spoofed, and since the sender is not authenticated message can appear to come from any phone number that attacker defines. Since there is no authentication, the attacker can also pretend to be the real SMSC by using the SMSC simulator. The proposed lightweight security mechanism can easily be used to identify if the messages is really coming from the MSISDN that is indicated in the message.

4.2.2.2.6 Device Management SMS Interface

This section describes the standard interface with device management module based on CoAP for simple session-less device management procedures over SMS transport. The CoAP protocol semantic can be also utilized for these device management operations.

Monitoring the end devices can be done by sending a GET CoAP message that can fit in one SMS. This enables access to predefined interfaces of the devices and obtaining updates from those devices. Remote firmware and software updates can be done by sending adequate POST messages. In addition, by sending a PUT message, software can be updated and changed.

Very often M2M devices are simple and cheap sensors being grouped in the capillary network, which cannot directly communicate with the network. In order to allow this communication, M2M Gateway is introduced to provide various functionalities, such as protocol translation, routing, resource management, device management, data aggregation, etc.

Remote Monitoring and Diagnostic

Remote monitoring operation (e.g. battery level, device state, etc.) can be achieved by sending a GET request to a number of devices that needs to be monitored. The simple GET procedure between the server and device is presented below:

1) Server sends an SMS message with embedded CoAP message to the M2M gateway
2) The Service Center returns a SMS-Submit-Report message to the DM server
3) The Service Center saves the received SMS message and forwards it to the M2M Gateway
4) The M2M Gateway returns a SMS-Deliver-Report to the SMS-C
5) M2M Gateway parses the SMS and forwards a CoAP GET request
6) Device responds with a content and acknowledgement of the received message
7) The Gateway forwards a CoAP message over SMS and receives a SMS-Submit-Report from the SMS-C
8) After delivering the SMS message to the Device Management Server, the SMS-C receives an SMS-Delivery-Report and confirms the message status with a SMS-Status-Report.

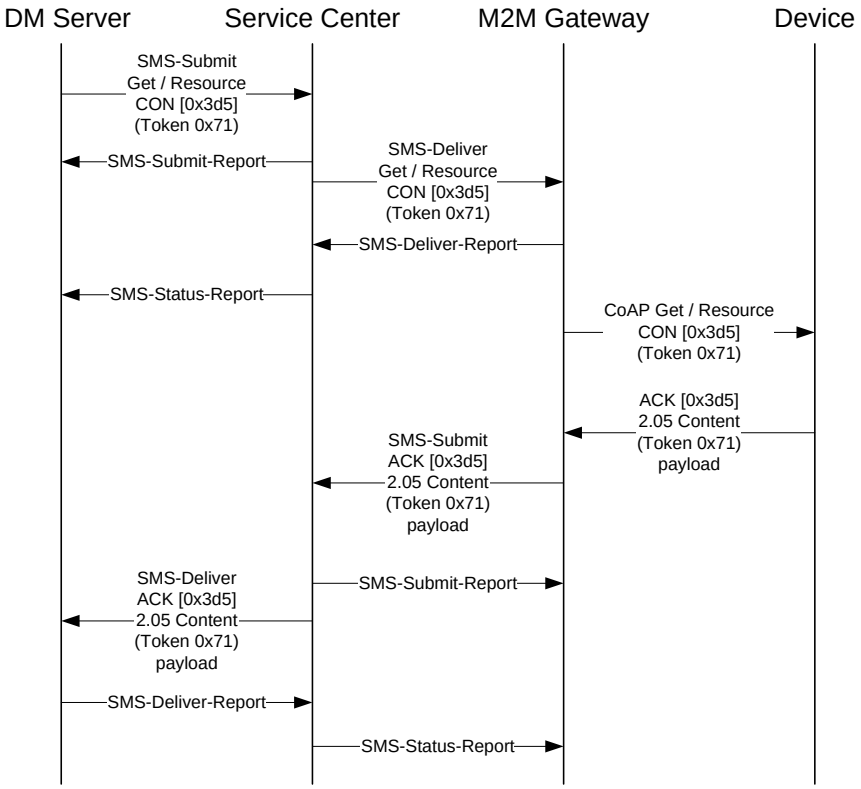


Figure 4-23: Remote monitoring of node over SMS with CoAP message

In a similar manner, every device management procedure can be executed. The removal of CP layer in this case means removing the SMS reports from UE and from the domain.

Remote Configuration and Firmware Update

Remote configuration and software/firmware update of devices can be achieved by sending PUT requests to a number of devices that needs to be configured/updated. This scenario is presented below Figure 4-24, reports are omitted due in a manner that excludes CP layer.

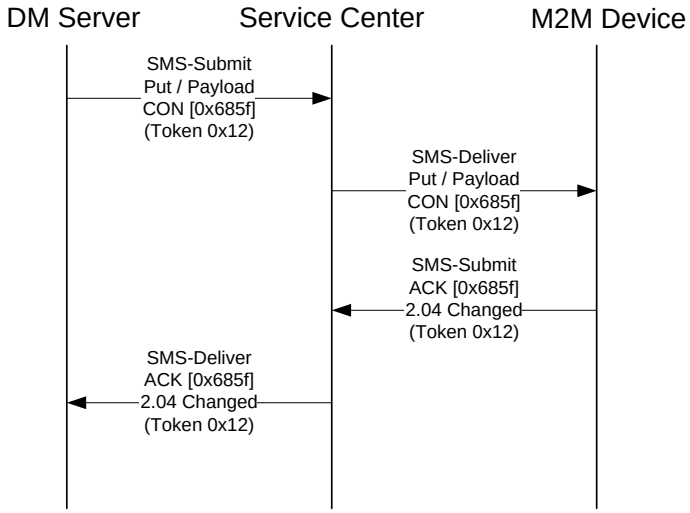


Figure 4-24: Remote configuration of node over SMS with CoAP message

4.2.2.2.7 CoAP over SMS evaluation

The SMS efficiency for CoAP was evaluated in implementation of CoAP over SMS in live GSM network. In this evaluation messages are sent between two devices and between five devices respectively. The implementation of this protocol is done on the Android powered mobile devices. First device requests resources from the other, which results with a CoAP message interchange over the SMS transport. Devices are capable to process the CoAP message and to respond to a request by parsing the message as defined in [58].

The main evaluation criteria were measurements of the message response time for SMS messages sent with CoAP GET and POST requests. The evaluation was conducted for regular, concatenated and multicast SMS messages:

1. Test 1: Regular M2M SMS communication
2. Test 2: Concatenated SMS test
3. Test 3: Multicast test

Regular SMS Test (1) is based on a sending of the GET request (Figure 4-25) to another device that responds with a resource value.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 | 0 | 1 |  1 | 12345 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 9   11 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| temperature|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4-26: The GET CoAP message

First number in the header is the CoAP protocol version. The sent messages are marked as CON (the second field of the header). The third number indicates that there are additional header options, and the rest numbers are method (i.e GET request), message ID; followed with a header option values in the second row; and the payload in the third row.

Code number from the header determines whether the message is GET or PUT and what method should be called on a device. The second device is parsing the message and returns a piggy-back ACK response (**Figure 4-27)** with a temperature value as a payload.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- +-+-+-+-+-+-+-+-+-+-+-+-+
| 1 | 2 |  0 | 69 | 12345 |
"22.3"
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-  +-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4-27: The response message with piggy-back ACK

The concatenated SMS Test (2) is based on a sending of the concatenated POST message (**Figure 4-28** to a device that responds with a single CON (confirmable) message.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| 1 | 0 | 1 |  2 | 12345 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| 1   2 |
| 41  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| <?xml version="1.0" encoding="ISO-8859-1"?>
        <name>Wasp Mote Libelium</name>
        <imei>351656021668193l</imei>
        <SensorData>
                <humidity>1225</ humidity >
                <temperature>30.5</temperature >
                <CO>312</CO>
   </ SensorData >|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 4-28: The POST CoAP with XML payload (3 concatenations)

The Multicast SMS Test (3) is based on sending GET request as in test (1), but instead one, message is sent to 4 devices in the same time.
The methodology used for all SMS communication tests is next: after each response which confirms message reception, the next request is sent. For cases (1) and (3) the other M2M device responds with a single message (**Figure 4-29**) with piggy-backed payload.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- +-+-+-+-+-+-+-+-+-+-+-+-+
|        1        |        2        |           0           |        65        |        12345        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-  +-+-+-+-+-+-+-+-+-+-+-+-+
                                  "22.5"
```

Figure 4-29: The ACK response message

Evaluation of CoAP over SMS was conducted for requesting and posting the resource from/to device. The second device receives the message and responds with a resource value with a piggy-backed CON message for GET request and with a CON message for a POST request.

4.2.2.2.8 Measurement result

The average response time for Regular and Concatenated SMS test was 6,87s and 26,98s respectively. Differences between the GET and POST response time for 1 and 3 SMS shows increased SMS sending rate impact on SMS-C delivery period.

	Test 1: Regular SMS (GET)	Test 2: Concatenated SMS (POST)

	Test 1: Regular SMS (GET)	Test 2: Concatenated SMS (POST)
Average response time [s]	6,87	26,98
Message size	1 SMS	3 SMS
Payload size [bytes]	16	374

Table 4-5: The response time for Regular and Concatenated SMS test

Evaluation shows that more time is needed to send payload spanned over 3 SMS, then for 3 messages sent one by one. This is due an SMS-C re-trying scheme that queues messages and sends it later if message limit is reached. Consequently, the higher the number of SMS sent in a short period is, the longer is delivery time and the lower is a chance that these messages will be delivered in a first try. Accordingly, the best practice would be to employ stop and wait mechanism and to segment payload on as many messages are required in order to decrease delivery time.

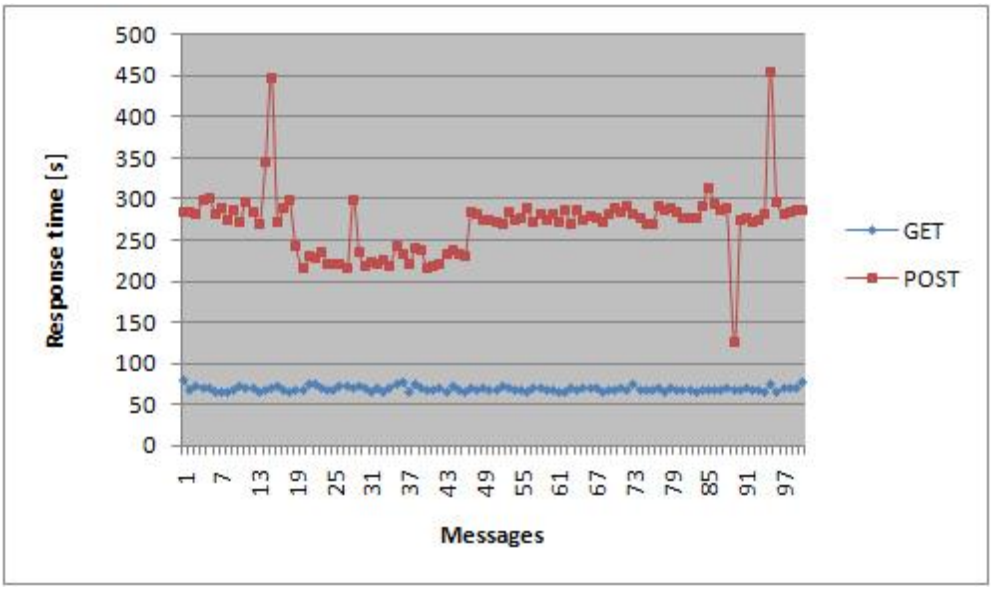


Figure 4-30: The message response time for GET and POST requests

Phone	Average	Standard deviation	Median
Phone 1	14,39s	6,11s	14,10s
Phone 2	22,86s	8,16s	23,60s
Phone 3	37,98s	21,09s	35,91s
Phone 4	50,65s	25,41s	45,12s

Table 4-3: The response time for Multicast test

Response times for phones 1-4 (in the table above) have similar trend except the phone 4. Device sends messages at once each time in the following order (Phone 1, 2, 3, and 4); thus the signal link is reserved in an order which results with longest response time from the device that last receives message. It is to expect that if we include more phones in the simulation, for every newly added phone, response time will be longer. SMS limitation can be overcome by obtaining SLA from Service Provider, which can guarantee a high SMS throughput. In addition, peaks are not so strong and frequent which may be seen from the similar median and average values for all phones.

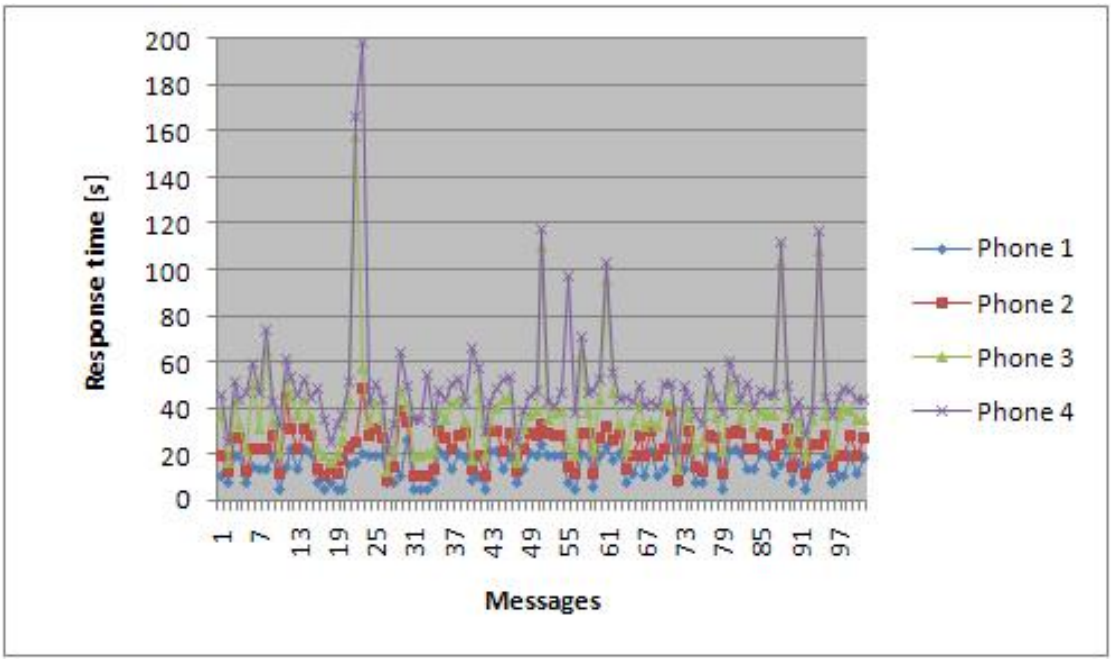


Figure 4-31: SMS Response time for Multicast test

Possible reasons for increased delays are: network activity, signal link capacity and Service Provider SMS policy (FIFO/LIFO). The reason for increased response times for larger number of SMS messages sent in short interval are FIFO (First in first out) Service Provider policy and over reservation of signal link. If the limit of the number of messages is reached, messages are entering retry scheme: the sending is delayed for a certain period of time that is growing with every next retry. If security proposal is used, concatenated message may be reduces by 10.6%.

Based on the experimental results, the following conclusion for SMS transport could be summarized:

(1) Limited Quota for regular MSISDN – each MSISDN has limited SMS throughput. Increasing this quota can be done by agreeing over specific SLA with SP and implementing the interface between SMS Gateway and SMS-C to send a high number of SMS messages.

(2) Long deliver time – SMS is not meant for data transport: messages can be received with significant delays, in reverse order and lost without notice. It is reasonable to expect for larger payloads that deliver time should increase significantly, so message Validity Time should be set to a longer value than the required sending period. This is not a concern for devices with long sleeping cycles, which is in general case in M2M networks. M2M Gateway can aggregate messages with larger payloads and to forward it to the end device after receiving the last segmented message and when

device is awake. Moreover, transmission time needs to be properly configured to account the round-trip delay (from the network edge to the SMS-SC and backwards).

(3) Hardware dependencies – the evaluation in this test is based on mobile devices that are not meant for high throughputs. Real SMS Gateway should have better capabilities in term of receiving and processing the messages, which will in general increase delivery time and communication in general.

4.2.3 ASN.1 data representation

Whatever is the transport layer the variety of devices to be administrated conducts to reconsider the choice made by the OMA-DM group with SyncML. This SyncML OMA open standard specifies a verbose protocol targeted for mobile handset. The ELFOMA (section 4.2.1) approach proposes to compress the OMA DM representation. The big advantage is to keep the compatibility with current deployed DM servers by deploying proxy servers only. To address SyncML size issue, current work at OMA aims at using a binary representation for the data management data, refer to OMA NG description is in section 4.1.2.2 and OMA-DM Lightweight in section 4.1.2.3.

Indeed, a binary data representation should be far more compact then character based representation like SyncML. In many respects the ASN.1 notation and encoding seems adequate for the purpose of the Data Management on small devices where computing resources can be limited and energy consumption very critical. The following sections first describe ASN.1 and study the most adequate encoding to optimize both the bandwidth to enable scalability at the server side and the processing time at device level. Only the encoding/decoding aspect is studied in this section and the main motivation is to use it for M2M adequate protocol. Standard committee working on this topic could be interested if there is opportunity to influence the encoding/decoding part of the protocol. Note that ELFOMA described earlier could also be encoded thanks to ASN.1. As mentioned later a new encoding/decoding method could also be specified based to meet specific requirements.

4.2.3.1 Abstract Syntax Notation One – ASN.1

Abstract Syntax Notation One (ASN.1) is a standard data description syntax that enables to specify data structure which gives independence from machine representation. This notation is then associated with encoding rules to provide the concrete data to be exchanged.

ASN.1 is widely used in many domains: UMTS, WIMAX and LTE use ASN.1 for its control messages, LDAP is specified using ASN.1, data transfers in Kerberos are specified with it as well as protocols used by International Civil Aviation Organization. It is used in many RFCs that specify the Internet protocols like RFC1157 that specifies SNMP.

Various encoding rules are standardized like Basic Encoding Rules (BER), Canonical Encoding Rules (CER), Distinguished Encoding Rules (DER) or Packed Encoding Rules (PER). The encoding rules define the bit patterns used to represent the abstract values. With the BER the physical representation itself is based on Tag – Length – Value (TLV).

From an ASN.1 description and an encoding rule a compiler is able to generate binary representations to be used together with a library to encode/decode the data. For a developer the way to proceed is first to describe its protocol using the ASN.1 grammar formalism and data types. Then these rules are compiled and the result of this compilation plus a run-time can be integrated in the code. Encoding and decoding functions/methods enable to manipulate the protocol data types in developer's code. An alternative is to code the rules by hand but a compiler relieves from this tedious implementation task. Note that while the ASN.1 rules are language independent the compiler has to target a specific language like C, C++ or Java. Compilers use to be very expensive but some freely available

compilers that enable to process small to medium protocols can be successfully used. Note that only products are able to generate for ARM boards.

ASN.1 has been standardized [66] by the International Telecommunication Union (ITU) in 2002. It is the result of a work started since 1984 by the CCITT (International Telegraph and Telephone Consultative Committee). The same text of the standard is endorsed by ISO. Both the notation and the encoding rules are part of the standard.

An example of the notation extracted from the ITU presentation of the ASN.1 [67] is reproduced in the Figure 4-32:

```
Order ::= SEQUENCE {
  header  Order-header,
  items   SEQUENCE OF Order-line}

Order-header ::= SEQUENCE {
  number   Order-number,
  date     Date,
  client   Client,
  payment  Payment-method }

Order-number ::= NumericString (SIZE (12))

Date ::= NumericString (SIZE (8)) -- MMDDYYYY

Client ::= SEQUENCE {
  name     PrintableString (SIZE (1..20)),
  street   PrintableString (SIZE (1..50)) OPTIONAL,
  postcode NumericString (SIZE (5)),
  town     PrintableString (SIZE (1..30)),
  country  PrintableString (SIZE (1..20))
               DEFAULT default-country }
default-country PrintableString ::= "France"

Payment-method ::= CHOICE {
  check        NumericString (SIZE (15)),
  credit-card  Credit-card,
  cash         NULL }

Credit-card ::= SEQUENCE {
  type         Card-type,
  number       NumericString (SIZE (20)),
  expiry-date  NumericString (SIZE (6)) -- MMYYYY -- }

Card-type ::= ENUMERATED { cb(0), visa(1), eurocard(2),
                 diners(3), american-express(4) }
```

Figure 4-32: Example of protocol specification using ASN.1

4.2.3.2 ASN.1 encoding rules

Because data representation may differ from one platform to another, typically Big-Endians/Little-Endians, there is a need to give a way to retrieve/send the data from/to a convenient data representation to exchange data.

Let's make the analogy with the inter-process communication technique. During a Remote Procedure Call (RPC) there is a mandatory step that adapts the arguments to unable correct interpretation by the caller/callee. This step is called the marshalling. It is a data transformation to get a suitable data format to be transmitted to the other process. The reverse operation is the unmarshalling. In ASN.1 there is an equivalent transformation that is called the data encoding/decoding.

Another analogy is the serialization operation used in object programming when an object is sent remotely. There is an equivalent step with ASN.1 that takes the data in clear format and encodes it using Encoding Rules.

There are several encoding rules specified that address different needs. The following table describes most well-known rules. BER, DER, CER, PER and XER are standardized.

Acronym	Name	Features	Comment
BER	Basic Encoding Rules	• Based on Tag, Length, Value descriptions (TLV) coded on bytes • Embedded structure value may itself be another TLV • Architecture independent: bit weight arbitrarily set	• This is the first encoding rules specified; this is where the "Basic" comes from. • No support of the subtype constraints because specified at the very beginning • High inflating rate from the actual data before encoding (about 40%) • Oldest encoding rules supported by all compilers
CER	Canonical Encoding Rules	• Kind of refinement of the BER • Indefinite length format • Two ways to represent bit string according to the length	• Adapted for protocol that have to transfer a large amount of data • Supported by few compilers
DER	Distinguished Encoding Rules	• Like CER, BER is a specialization of the BER • Define length format • Constraint on bit string representation	• Available on most compilers • Designed for secured data transfer (signature)
PER	Packed Encoding Rules	• Purpose is to compress the encoding data at the most: TLV are not systematically used • Uses the lower and upper bounds of numeric values to limit the size • Relies on subtype constraints • Based on a series of **bits** (length for example) for the encoding while bytes in BER • Variant of PER with basic, canonical, aligned,	• Design to reduce the BER inflating rate • More compact encoding than BER • Use many subtypes to remove the length field. • Because the tag is not systematically specified there is no extensibility of the protocol over time and any future evolution must be stated at the beginning. • Tags not encoded, length only encoded if not fixed or if it is large

		unaligned. Canonical adapted for signature while aligned introduce padding.	• Not only transfer is quicker than BER but also encoding/decoding operation is less CPU demanding • Less supported than BER, DER and CER but some open compilers exist • Trade-off between compaction (unaligned variant) and processing time optimization to be arbitrated
XER	XML Encoding Rules	Self-explicit encoding rules name	Very verbose
	Other encoding rules	• Any organization or company may design its own set of rules. • The Generic String Encoding Rules (GSER) is a RFC but it is not endorsed by ITU. It is a very verbose representation made for human readable textual	• Non-standard rules designed for specific use.

Table 4.6: ASN.1 Encoding Rules comparison.

4.2.3.3 Favourites encoding rules

Various encoding rules have been considered with respect to Device Management. As described in the previous paragraph ASN.1 grammar specification is used in association with different encoding rules. A simple approach to measure the performance of each encoding rule is to encode the same data with different encoding rules. The first example of data to be encoding will be the OMA-DM Replace command also used by ELFOMA in a previous section of this document (section 4.2.1). These data are described in the Figure 4-7: OMA-DM payload –Replace commands – 1170 bytes.

First step is to analyse the data and to specify the ASN.1 grammar of the document. The Replace command is specified in the Figure 4-33: ASN.1 grammar for OMA DM Replace. The grammar is simple and readable.

The second step is to compile the grammar with the ASN.1 compiler. A larger number of compiler exist today either commercial or open source.
For the testing phase the OSS Compiler [69] has been used. Some other compilers could be used than this commercial product.

The third step is to encode the data with the ASN.1 run-time provided together with the compiler. For the test the same data used in ELFOMA is taken. It is described in Figure 4-7: OMA-DM payload –Replace commands – 1170 bytes. Results are gathered in the following Table 4.7:

```
ModuleName DEFINITIONS ::=
BEGIN
  Replace ::= SEQUENCE {
```

```
    cmdId INTEGER,
    items SEQUENCE OF Item
  }
  Item ::=SEQUENCE{
    source SEQUENCE{
      locURI IA5String
    },

    meta SEQUENCE {
      format PrintableString,
      type IA5String
    },
    data IA5String
  }
END
```

Figure 4-33: ASN.1 grammar for OMA DM Replace

ASN.1 Encoding rules	Length (bytes)	Compression Rate
BER	280	4,18
DER	280	4,18
Aligned PER	222	5,27
Unaligned PER	198	5,91
CER	310	3,77
XER	1173	0,997

Table 4.7: Result of encoding with ASN.1

The reference XML data has a size of 1170 bytes. The most approaching ASN.1 encoding rules of XML is the XER that is bigger than the XML representation. Reason is that it has more or less the same encoding with an additional line: “<?xml version="1.0" encoding="UTF-8"?>”. All other encoding rules give a smaller encoded data.

BER and DER encoded data are only 280 bytes long and the smallest encoding rules are PER. Unaligned PER is smaller than aligned PER because of extra “0” bits padding added with aligned PER. As expected, the most compacted encoding rules are unaligned PER. The compression rate figures are aligned with the results of a paper published in “IT Professional” [68] that aimed to propose ASN.1 to replace XML representation for general purpose use.

Another criterion test is the time required to encode and decode data because this processing time is correlated with the energy consumed. In the same paper the time to encode data is 6.6 ms for the XER. Let’s approximate that XML and XER have the same processing time. Only 2.812 ms are required to process data for unaligned PER. Time to decode the information from XER is 22.0 ms and 5.008ms for unaligned PER, the reduction rate between processing time of XER and PER is 4.39. It means that in addition to the bandwidth consumption reduction the time to encode/decode data is also significantly reduced. It also means that energy consumption is reduced by using an ASN.1 PER encoding rule compared to an XML based representation.

For reference the binary encoded data produced with the unaligned PER encoding rules is printed below. An XML representation if definitively more user friendly but this is not the main requirement for communication between devices even if nice during development and test phases.

```
01 03 05 0F 5C BE 26 5E D2 77 66 DE BE 26 5E D2
72 01 C3 D1 C8 2B A6 5F 1D 17 F0 D9 87 4E E1 49
33 62 C9 74 D1 CB 36 0C 1A B1 60 C1 AB 96 4E 18
30 0D 5C BE 26 5E D2 77 66 DE BE 6E 1D C0 78 F4
72 0A E9 97 C7 45 FC 36 61 D3 B8 52 26 5E DA 71
E5 41 1B 0E 3E 9B F9 79 58 82 4E EC 6B 83 57 2F
89 97 B4 9D D9 B7 AF 9B BF 20 1E 3D 1C 82 BA 65
F1 D1 7F 0D 98 74 EE 0E A7 B7 0F 2E 94 34 6F DD
95 93 06 0C 03 57 2F 89 97 B4 9D D9 B7 AF 89 B6
B0 1E 3D 1C 82 BA 65 F1 D1 7F 0D 98 74 EE 07 62
B9 82 E6 0B 98 87 2E 5F 13 2F 69 3B B3 6F 5F 33
0E EC E0 78 F4 72 0A E9 97 C7 45 FC 36 61 D3 B8
17 2E E5 B5 69 80
```

Figure 4-34: Binary representation with PER Encoding Rules

4.2.3.4 Optimized representation

Various optimizations are proposed in ELFOMA that are also applicable to the ASN.1 representation.

- The use of default values can reduce the payload size, if the default data is transferred then it can be omitted and at decoding phase the defaults values will be added.
- Another proposal is to represent command as token codes of 2 hexadecimal characters (8 bit): 256 different commands can be specified with this token code.
- A mapping can be used between the tree-based model and the flat data model. Tree-based nodes referenced by a URI can be mapped to node codes which reflect flat data variables.
- A mapping can be used for different type of data for example:
    - for type specification, 1 can map basic authentication;
    - for format specification, 1 can map base64 encoding;

These improvements are tested through two examples: The XML of first one is presented in Figure 4-4: OMA-DM payload – SyncHdr sample – 519 bytes. The ASN.1 grammar is specified in the Figure 4-33.

```
TestModule DEFINITIONS ::=
BEGIN
  SyncHdr ::= SEQUENCE {
      verDTD PrintableString,
      verProto PrintableString,
      sessionId INTEGER,
      msgId INTEGER,
      target SEQUENCE {
       locURI IA5String
      },
      source SEQUENCE {
       locURI IA5String
      },
      cred SEQUENCE{
       meta SEQUENCE{
          type INTEGER,
          format INTEGER
       },
       data PrintableString
      },
      meta SEQUENCE{
       maxMsgSize INTEGER
```

```
        }
    }
END
```

Figure 4-35: ASN.1 grammar for OMA DM SyncHdr

In the table 7.3 the compression rates take the size of the initial SyncHdr payload that is equal to 519 bytes.

Encoding Rules	Length (bytes)	Compression Rate	Length (bytes) w/optimization	Compression Rate	ASN.1/ optimized ASN.1
BER	143	3,63	98	5,29	+46%
DER	143	3,63	98	5,29	+46%
Aligned PER	118	4,36	76	6,83	+55%
Unaligned PER	106	4,90	69	7,52	+54%
CER	158	3,28	110	4,72	+44%
XER	600	0,86	458	1,13	+31%

Table 4.8 : Results of SyncHdr payload encoding

The second example will be again the OMA-DM Replace command payload presented in Figure 4-8. In this case the improvement using the optimization is more significant because the document has a lot of data that are mapped to different tokens. The Unaligned PER is 13.6 times smaller than the original XML. We gain in performance of the encoding, but for more processing time for encoding and decoding data is required.

This optimization comes at a price: a file specifying the mapping between different elements and tokens is required and it must be located on each device that need to encode/decode the data.

Encoding Rules	Length (bytes)	Compression rate	Length (bytes) w/optimization	Compression rate	ASN.1/ optimized ASN.1
BER	280	4,18	150	7,8	+87%
DER	280	4,18	150	7,8	+87%
Aligned PER	222	5,27	94	12,45	+136%
Unaligned PER	198	5,91	86	13,6	+130%
CER	310	3,77	182	6,43	+70%
XER	1173	0,997	975	1,2	+20%

Table 4.9 : Results of Replace payload encoding

4.3 Security considerations

4.3.1 Requirements

From the security requirements [8] of the project all types of devices and Secure Elements should be managed in the same way. In practice it means that provisioning of a USIM card, hardware Secure Element or software Secure Element could be embedded in the device without impact on the way credentials are managed. This requirement is not addressed because it is much too strong. It is assumed for the implementation of the security requirement that the devices embed hardware Secure Element, software only Secure Element is not considered.

White-Box Cryptography (WBC) is the technology used to hide cryptographic key by software. It is implemented by putting together both the cryptographic functions and the keys in such way that it is very hard to find the keys. Theory [74] says that there is no way today to hide the keys for good: a patient attacker is able to retrieve the keys. There is also a M2M constraint that is to limit the code size and to avoid useless processing to limit the energy consumption. Except if new algorithms are found in the coming years WBC cannot be used in M2M without putting keys at risk. As a consequence the Secure Element implemented in software cannot have a pre-shared key configured before deployment and then the security bootstrap shall use a Diffie-Hellman mechanism to set the credentials when the devices are deployed with the major drawback that it is sensible to the Man-In-The-Middle (MITM) Attack.

It is even more critical for Low Cost Devices handled by many people. Some devices could be stolen and the attacker has all time he needs to introspect the binary code offline to extract the pre-shared keys. For these reasons (constraints on the consumption and lack of security) software Secure Element protection is not considered as a viable design today and it is dropped.

Not only Data Protection must be provided but the security system should not decrease the protection because of any misuse of subscription data by an attacker. The security bootstrap must be secure enough not to compromise the E2E security based on credentials set during this first phase or during key management phases.

4.3.2 End-to-End (E2E) security

The security in EXALTED is proposed at the application level. Application payloads can be exchanged in a secure way that protects the confidentiality and the integrity of the data; it means that application payloads are protected from eavesdropping and unsolicited updates. Application payloads are protected between ends, typically an end-device and the M2M server. This is a very strong assumption that prevents some algorithms studied in the project to benefit from the security also proposed in the project. There are no satisfactory answers to this contradiction in the project because optimizing performance at the CH or the gateway level for example supposes to have access to the content of the payload while there is no way for the security to make a distinction between an attacker and the optimization algorithm.

The only possible answer is to limit the requirements: either there is no more E2E but only security between the gateway and the M2M server for example and optimizations can be invoked to reduce the bandwidth or application payloads are protected by E2E but there is no data aggregation and payloads are untouched until the Server. Note that in the former case the local security applicable at the capillary network level can still be used of it exists. This might be an acceptable trade-off for applications that want to optimize the traffic with intermediate security level.

4.3.3 Federation of devices

To provide the E2E security feature then keys are required at both ends that may communicate. Since there might be a considerable amount of devices then the number of keys is dramatically high which induces large data base at server level and many messages to manage those keys. To address this issue it is proposed that some devices share their communication keys. A group key management scheme is the only viable solution for a significant M2M deployed solution with a significant number of devices that need secure communication.

4.3.3.1 Group Key

The main result of the design is the use of a group key that creates a federation of devices that share the same security key to communicate with the server. Intuitively these devices are part of the same capillary network but this is not mandatory. Devices sharing the same

group key could be part of various capillary networks or even dynamically move from one capillary network to another one. Because components between the end-device and the server are not security aware it enables to have a very versatile notion of group key. This group key has been highly suggested in [8] in the Provisioning section.

In the scenario considered for the testbed only the integrity of the payload is proposed to limit the implementation but within a group several keys can be shared. What is not possible with the current design is to have dedicated group by protection type: a device is part of the same group for confidentiality and integrity protection, not different groups.

Several federations of devices are shown in the Figure 4-36 below:

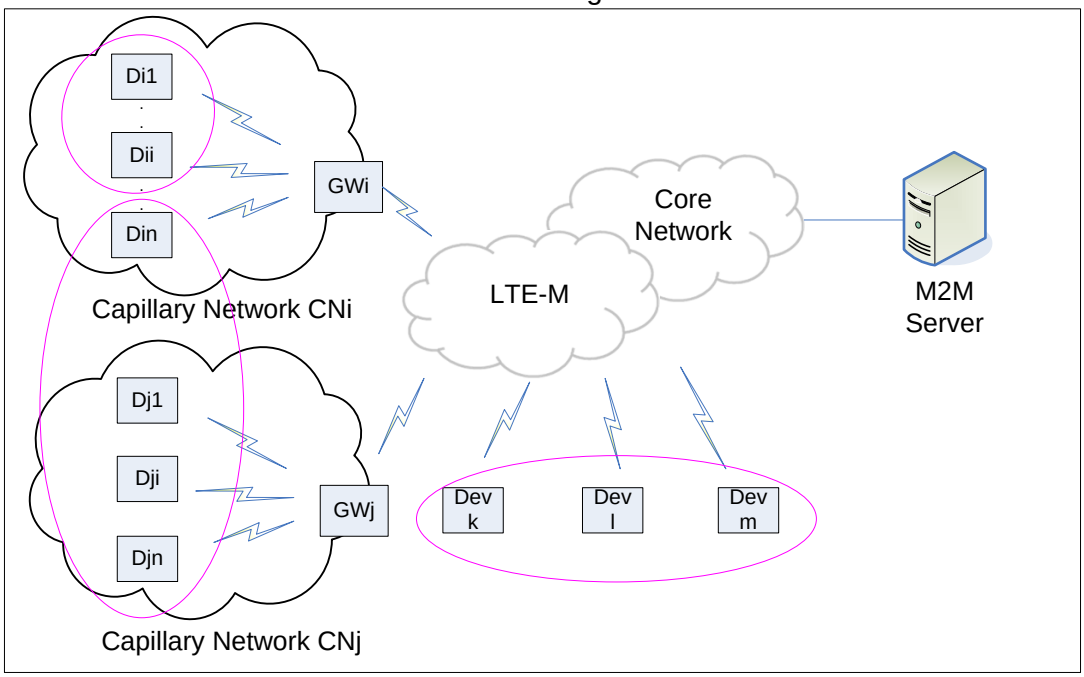


Figure 4-36: Group Key sharing among end-devices

There are 3 different federations with 3 different set of group keys: Devices Di1 until Dii share the same group key in the same capillary network while the devices Din, Dj1 to Djn are sharing the same group key over two capillary networks. In the figure there is a third federation of LTE-M devices sharing the same group key. If required all these devices could share the same group key even if there are LTE-M devices and non LTE-M devices and part of different capillary networks. The group key federation is separated from the physical organization of the devices and can be organized on the server at will.

The end-devices have to authenticate with the M2M server and a possible design is to use Public-Key Infrastructure (PKI) to implement the security or just to establish a shared group key for a device. This is an option that has been left aside because of the extra cost required by the certificate management and the burden to manage a Certificate Authority. The configuration phase is also much more demanding before deployment: a private key must be set in the Secure Element by whatever mean and certificate management must be synchronized by a server at device level. The option of having a pre-shared key for a set of end-devices has been preferred because much more convenient in daily life and more adapted to the M2M industry with large amount of low cost devices to be deployed on the field. Still, a requirement that must be addressed is to be able to uniquely identify a device for basic Device Management needs. This can be easily addressed by preparing any serial number within the secure element for this specific purpose. This unique device identifier is also required for the device addressing (section 5.).

4.3.3.2 Group Key setting

A handshake protocol to set the Group Key is described in 8.8.2. It has been already been published in another document [10] of the project because the intent is to use it for the testbeds. The main idea is first to use the Fabrication Key that is initially shared by both the device and the server to exchange two random values in a secure way. Then a Session Key is set using both random values and the last step is to enable the device to retrieve the Group Key sent by the server, still protected by the Session Key.

This is a generic protocol that does not presuppose the number of group keys that can be set in the device. In one scenario considered as generalization of the Group Key there is a specific Device Management Group Key and several group keys to be used by various service providers. With this design a device can be part of several federations of devices according to the need of the service provider.

This protocol complies with most recommendations from [8] in the Session Key Distribution section that are confidentiality and freshness.

4.3.4 Automation

The order of magnitude of the number of M2M devices leads to a different deployment model compared to what has been experimented with UICC in handsets. It is not desirable that an UICC is inserted by hand on each device to be secured because it would add extra device handling at deployment time and also because the pairing between the device and the Secure Element would have to be managed dynamically on site which leaves some room for the attacker.

With the Secured Element designed in the project it is soldered in the end-device and there is a need for a personalization step to set the keys in the secure element and to associate the device with its secure element in the device manufacturer factory. In practice it means that the Device unique identifier must be stored in the Secure Element. Depending on the rhythm of production and according the manufacturer process a set of devices will share the same fabrication key. Then the database of the server has to be filled with the data of the devices to be deployed. The fabrication key is part of these data. When devices are operational then the devices have to be activated. This activation replaces the pre-configured keys with the actual group keys. When the group key is set the end-device is ready to be used by the service and the exploitation starts.

5. Device Naming and Addressing

Based on high level point of view of the EXALTED system architecture, there are 3 main entities: M2M Server, Core/Access Network and Device. The device needs to be identified by the other 2 tiers.
Core/Access Network uses access network subscription identifier such as IMSI, to identify the device and to control its access to the network. This identifier is assigned by network operator.
M2M Server uses M2M Service provider subscription identifier to identify a device in order to tie the device data to a predefined service or business logic. For example, electrical metering data collected from device DevID1 will be used to establish the electrical power consumption billing for the household having the smart meter DevID1. The sensor data packet originated by the smart meter shall also include the device identifier DevID1. This type of identifier is used at the application layer and it is decoupled from operator assigned identifier.
Another type of device identifier is called the external identifier. Such identifier can be used by the access network to identify the device and by M2M Server to address the device. Currently, the mostly used external device identifier for mobile networks is the E162 MSISDN. However, forecasted massive deployment rate of M2M devices will lead to the shortage of MSISDN. Hence, it is not recommended to use MSISDN based identifier in M2M applications. The industry is currently working on creating a standard naming and addressing approach for M2M system.

5.1 LTE-M enabled devices

Within EXALTED, LTE-M enabled devices and M2M Gateways can be addressed by M2M Server over IPv6 and identified by the access network using IMSI (15 digits long). IPv6 addressing space is large enough to support massive deployment of the LTE-M enabled devices.
DM protocols as defined in the previous section can thus be supported on IPv6 connectivity between M2M Server and LTE-M enabled devices. The M2M Server (hosting DM Server) manages directly LTE-M enabled devices (embedding DM Client) over the EXALTED Lightweight DM protocols. Moreover, EXALTED has proposed an enhancement of the current DHCPv6 protocol, LTE-M enabled device can obtain IPv6 address faster while reducing the bandwidth usage, for more information, refer to Deliverable D4.2 [5].
M2M Server uses unique identifier to identify and to address devices at the DM protocol layer.
Device unique identifier is basically a string that uniquely identifies a device. Due to variable length, there is no limitation in term of addressability space. It could comprise for instance, serial number, device type, domain name, application name, etc. This concatenation may yield a long string which inflates the payload size. In this case, the long identifier could be hashed (e.g. MD5, SHA-1, SHA-2…) to shorten its length.

5.2 Non-LTE-M enabled devices

Non-LTE-M devices that are deployed behind M2M Gateway can however not be addressed directly by M2M Server. There are 2 DM approaches to manage devices in capillary network.

The first approach aims to enable M2M Server to address end devices directly, by implementing an address translation mechanism at the M2M Gateway. Using address translation approach the directly addressing is performed at the transport layer. DM Server in M2M Server can thus directly manage Non-LTE-M. DM Client of the capillary end device has to support the EXALTED Lightweight DM protocol as defined in the previous section. Several address translation mechanisms have been proposed in EXALTED, refer to Deliverable D4.2

[5]. One address translation algorithm is dedicated to Vehicular communication scenarios, IPv6 addresses are defined based on VIN (Vehicle Identification Number). Another address translation solution is tailored for 6LowPan capillary network. As such, end devices deployed within 6LowPan capillary network can also be managed directly by M2M Server. A third method enables M2M Server to address Non-LTE-M devices uniformly using IP address along a port number. Each end device in capillary network has therefore an address as follow: IPaddress:Port#. The M2M Server originated data packet is publicly routed to the M2M Gateway using the IPaddress. The M2M Gateway then use the Port# to lookup the corresponding private address used by the recipient device. This address translation scheme enables heterogeneous connectivity, as the port# can be mapped to the addressing scheme of any short range wireless technologies e.g. Zigbee, Bluetooth. The number of addressable devices is very large as each IP address (gateway) can, in theory, handle up to 65535 ports (end devices).

In the second approach, there is no direct addressing at the transport layer. M2M Server uses unique identifier to identify and to address devices at the application layer. In this indirect device addressing approach, M2M Server manages directly M2M Gateway, this latter then manages end devices. This approach is described in section 8.5. This naming and addressing scheme applies to Non-LTE-M devices (deployed behind M2M Gateway) is flexible enough to cope with the forecasted massive number of devices and with the future naming and addressing standard.
Device unique identifier is basically a string that uniquely identifies a device. Due to variable length, there is no limitation in term of addressability space. It could comprise for instance, serial number, device type, domain name, application name, etc. This concatenation may yield a long string which inflates the payload size. In this case, the long identifier could be hashed (e.g. MD5, SHA-1, SHA-2…) to shorten its length.

5.3 Addressing M2M devices

This section describes the approach to enable an entity in a public addressing domain to send messages to a M2M end device in a private addressing domain, behind a gateway.
A lifetime URI (Universal Resource Identifier) is assigned to each device. The URI will not change regardless where the end device is actually located. For more details, refer to section 6.3.3. This resource representing the device is obtainable via a protocol (e.g. HTTP, CoAP). For instance, the URI http://m2mserver.com/device/ID2345 identifies the device whose resource can obtained via http protocol. From this point, more resources can be defined to map new functions to be exposed to outside world.
Let's consider the following simple cases:
http://m2mserver.com/device/ID2345/firmware/pkg, represents the firmware package. HTTP PUT method should be used to send a new firmware package to the device.
http://m2mserver.com/device/ID2345/firmware/upgrade represents the firmware upgrade action. HTTP Post method could be used to trigger the firmware update.
http://m2mserver.com/device/ID2345/data, represents the last sensor data. HTTP Get method should be used to retrieve the sensor data.

6. DM Functions & Service Capabilities

This section describes how DM functions and Services can be realized over the EXALTED Lightweight DM protocol as described in section 4.

6.1 Principle

Each device has attributes, settings, status, sensor values, etc... to reflect its various properties. For instance, how often the device should post sensor data, which encryption algorithm should be used to encrypt sensor data, encryption key or certificate, credential to be used for authentication. This information is typically represented by a set of parameters, key-value pair; parameter name and associated value. For example, dataPostFreq=3600 (to post data every hour), dataEncryp=AES (encrypt data with AES), dataEncrypKey=0x67FAEC39 (symmetric key for AES). Configuring the data posting feature of the device implies therefore altering the value of nodes dataPostFreq, dataEncryp, dataEncrypKey. Basically, to realize a DM function, M2M Server interfaces with the device over a DM protocol in order to update parameters within the device.
Device parameters are grouped per function basis and organized as a hierarchical tree structure in the device. This tree is called device management tree. Each node (parameter) of the tree can be uniquely addressed with a URI. A Node is also called management object. For more information, refer to OMA DM Tree and Description [70].
OMA-DM Protocol [35], ELFOMA (defined in 4.2.1) and ELFOMA with ASN.1 encoding (refer to section 4.2.3) allow management commands to be executed on nodes. A node might reflect a set of configuration parameters for a device. Actions that can be taken against this node might include reading and setting parameter keys and values. For instance, create a new node on the device, read or update the value of the node, or delete a node. Another node might be the run-time environment for software applications on a device. Actions that can be taken against this type of node might include installing, upgrading, or uninstalling software elements. The following DM commands are supported by OMA-DM Protocol to cover the aforementioned actions: Add, Replace, Get, Delete, Copy, Exec, Alert, Atomic, Sequence, etc. For more information refer to OMA DM Representation Protocol [36].

The aforementioned components and functional interfaces are summarized in Figure 6-1: DM Enabler Architecture. An enabling function A is realized by interaction between EnablerA server and EnablerA client. These latter client and server communicate over the EnablerA interface, using the underlying DM protocol.
Based on the data posting example as described earlier, the data posting function in the device/sensor is enabled by the data posting enabler Client. The settings of this data posting (dataPostFreq, dataEncryp, dataEncrypKey) are persisted in the associated enabler management tree. The data posting enabler server (hosted in M2M Server) can manipulate the management tree of the enabler client over the enabler interface (e.g. create, read, update or delete the nodes in the management tree) in order to change the settings of the client. This node manipulation operation is actually performed over the DM Protocol; DM commands are used to address the nodes. Upon receiving and executing commands, the client enabler sends status report back to the server over the enabler interface using the underlying DM protocol.
It should be noted that the enablerA server may also have a copy of the enablerA management tree (not depicted in the below figure). M2M Application can access this management tree to retrieve the settings of the enablerA even though the sensor is offline. This tree can also hold new configuration values set by M2M Application. These pending settings will be applied to the sensor when it becomes online.

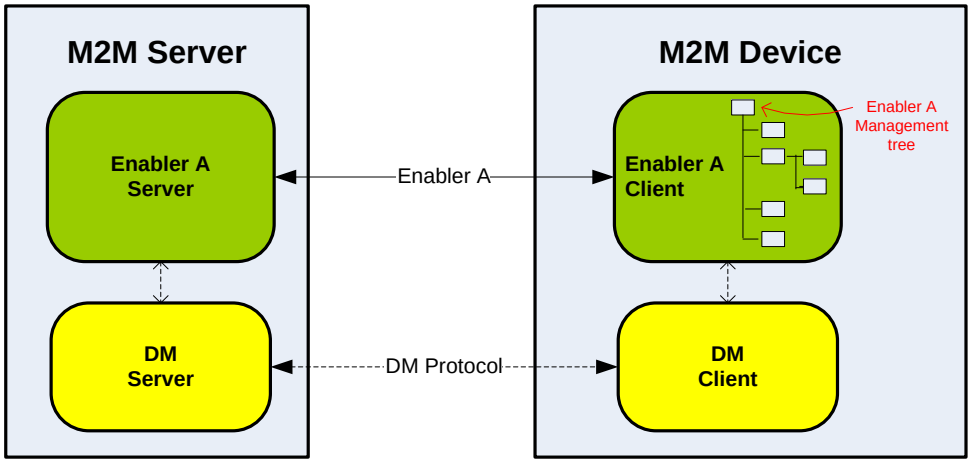


Figure 6-1: DM Enabler Architecture

This enabler architecture is also depicted in Figure 3-1 and Figure 3-2

6.2 Key DM Functions

OMA-DM protocol defines the message sequence and structure to support the execution of DM commands on Management Object. The EXALTED lightweight DM protocols as defined in section 4.2.1 and 4.2.3 are a lightweight adaptation of OMA-DM protocol, yielding compact DM messages.

Key DM functions are supported in EXALTED by leveraging existing enablers defined by OMA:

- FUMO [59], for Firmware Update
- SCOMO [64] , for Management of software components (install/upgrade/removal)
- DiagMO [60], for remote diagnostics and monitoring
- DCMO [72], for managing device capability
- LaWMO [71], for locking device and wiping device data remotely
- GwMO [39], for managing end devices deployed behind gateways

All necessary variables (nodes) to support these functions are defined and organized in a tree hierarchical structure.

Self-Diagnostic and Self-healing enabler developed in Deliverable D6.3 [9] can also be leveraged as an alternative solution to DiagMO.

As the procedures for the above DM functions are specified by OMA-DM, they are not detailed in this report. Description is only provided for firmware update and device capabilities configuration functions to show how the MOs enablers can be used in ELFOMA.

6.2.1 Firmware update support in ELFOMA

As defined in FUMO [59], the parameters associated with a single firmware update are assembled into a firmware update management object as shown in Figure 6-3. A firmware update management object may be either permanent or dynamic. There may be one or more such firmware update management objects in a device management tree. Only one update package, or reference to an update package, is associated with each such management object.

The following process steps are required in order to achieve an OTA firmware update:

1. Firmware Update Initiation, a trigger to have the device to start the process. This trigger may be initiated by M2M Server (e.g. SMS) or by the device itself (e.g. periodic polling scheduled)
2. Device Information Exchange. Device FUMO client connects to FUMO server (hosted in M2M Server) over the DM protocol
3. Based on the information provided by the FUMO client (i.e. device identity, current software/firmware version), the FUMO Server sends the proper firmware package data back to the device.
4. FUMO server initiates the client to start firmware upgrade. The device's FUMO client proceeds to the installation of the firmware package (e.g. by the way of a Firmware update agent, not depicted)
5. Notification of Firmware Update. Upon completion of the update operation. The device reports the status (success or failure code) to FUMO Server.

One firmware update scenario that comprises the above steps is depicted in Figure 6-4

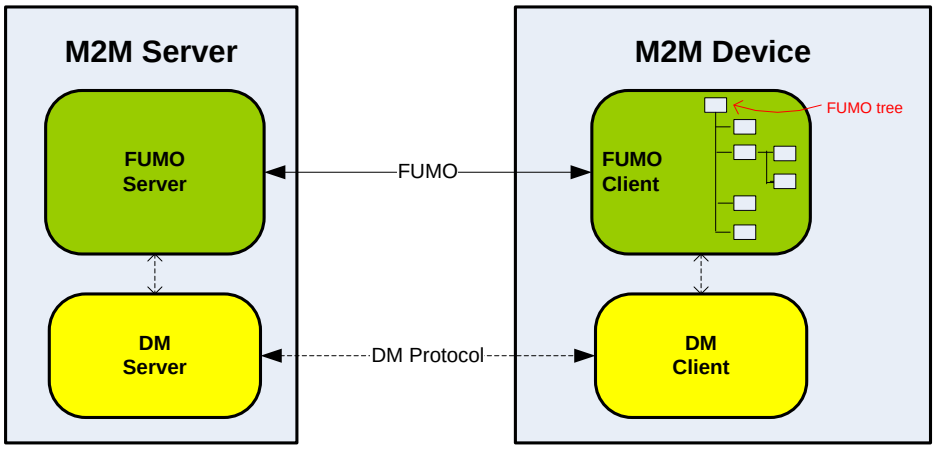


Figure 6-2: Firmware update architecture

This FUMO architecture is reflected in the global DM architecture (Figure 3-1 and Figure 3-2), where FU-1 is the internal interface between FUMO Server and DM Server. FU-2 is the internal interface between FUMO client and DM Client. FUMO Server and FUMO Client interface with each other over the underlying DM Protocol (DM-1 and DM-2).

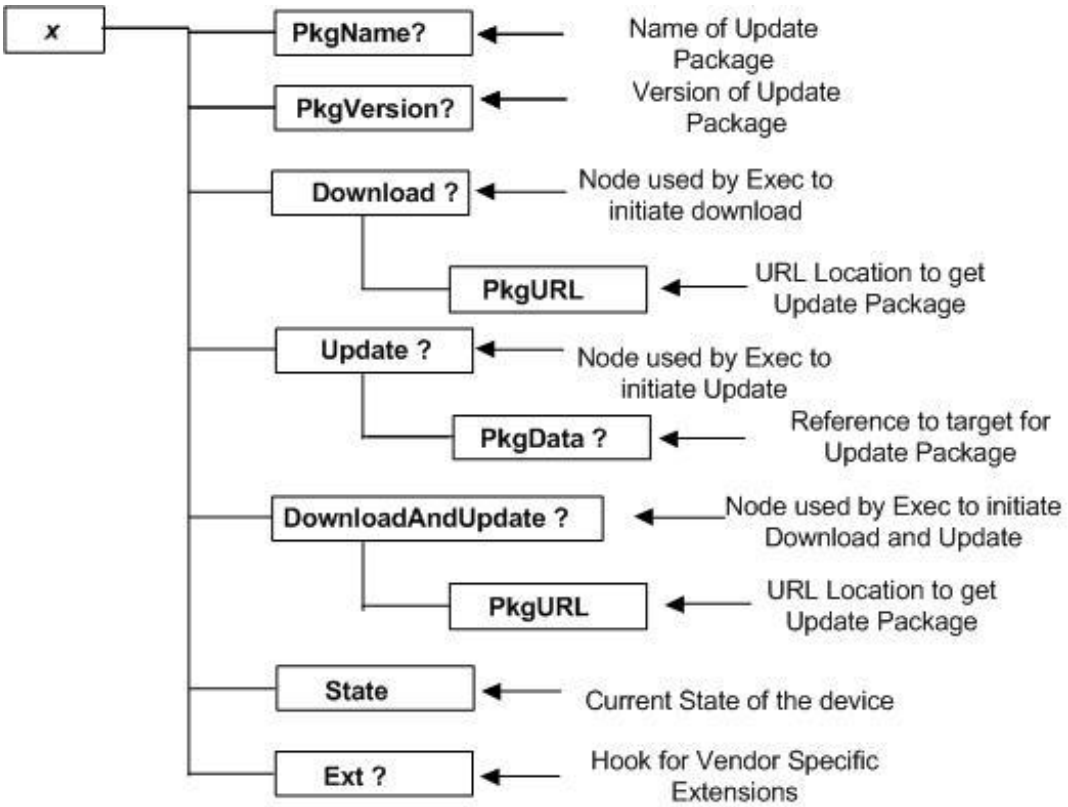


Figure 6-3: OMA Firmware Update Management Object (FUMO tree)

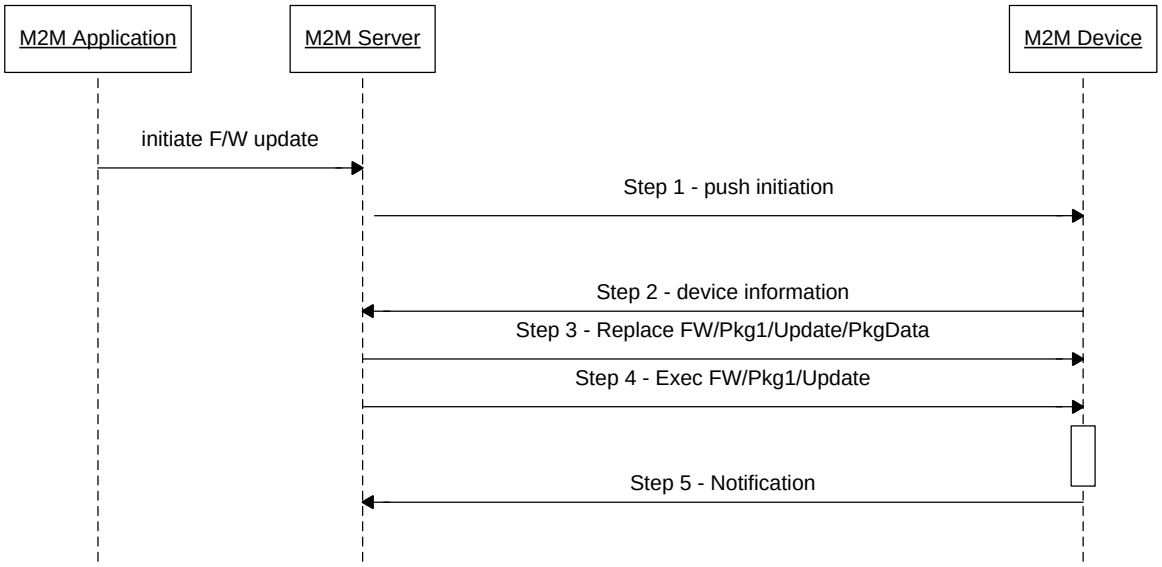


Figure 6-4: Firmware update process steps

An excerpt of messages in step 2 and step 3 are shown in Figure 6-5. Left column contains OMA-DM v1.x commands and the right column is the counterpart ELFOMA message content.
FUMO can be reused by ELFOMA to support firmware update function while reducing the message payload.

OMA-DM v1.x message

Step 3:

```
<Replace>
   <CmdID>11</CmdID>
   <Item>
      <Target>
         <LocURI>
            FW/Pkg1/Update/PkgData
         </LocURI>
      </Target>
      <Meta>
         <Format xmlns='syncml:metinf'>
            bin
         </Format>
         <Type xmlns='syncml:metinf'>
            b64
         </Type>
      </Meta>
      <Data>
      Zz6EivR3yeaaENcRN6lpAQ==
      </Data>
   </Item>
</Replace>
```

Step 4:

```
<Exec>
   <CmdID>3</CmdID>
   <Item>
         <Target>
            <LocURI>
               FW/Pkg1/Update
            </LocURI>
         </Target>
   </Item>
</Exec>
```

ELFOMA message

Step 3:

```
20=11;{FW/Pkg1/Update/PkgData;;
Zz6EivR3yeaaENcRN6lpAQ==;{b64;bin;}}
```

Step 4:

```
11=3;;FW/Pkg1/Update
```

Figure 6-5: Execution of DM commands on FUMO over OMA-DM & ELFOMA

It should be noted that in the above example, the firmware data is transmitted to the device over the DM protocol. Hence the constrained device does not need to implement additional protocol (e.g. FTP) to download the firmware, the complexity is thus reduced. The firmware data in the message as shown in Figure 6-5: Execution of DM commands on FUMO over OMA-DM & ELFOMA is intentionally made short for example clarity purposes.

This sample should provide a good insight on how to execute DM commands on Management Objects in order to perform a DM function on the device. No further message excerpts will be listed in the remaining report, instead the corresponding MO enablers will be described.

6.2.2 Device Capabilities and Configuration

There are several methods to reconfigure a device. The method to be used depends on the configuration scope. Bootstrapping and provisioning are used to configure device security credential, connection setting to network or to management server. Firmware and software update is another alternative to change device's feature settings. Discovering and managing device capability is another aspect of device configuration.

Due to diversity of M2M applications, various types of devices are deployed in the field. These devices must be configured by M2M Server generically. A device may have multiple capabilities. For instance, GPS chipset, accelerometer, temperature and humidity sensors could be embedded in a device. Device capabilities are not limited to sensing features, network connectivity related capabilities are needed for gateways and CHs. For instance, these latter entities may support various types of connectivity such as Zigbee, Bluetooth. M2M Server needs to discover device capabilities and to configure them.

OMA's Device Capabilities MO [72] enabler is be leveraged to configure devices. Each single capability of a device can be represented by the device capability management object as depicted in Figure 6-6: Device Capability Management Object. The architecture is depicted in Figure 6-7.

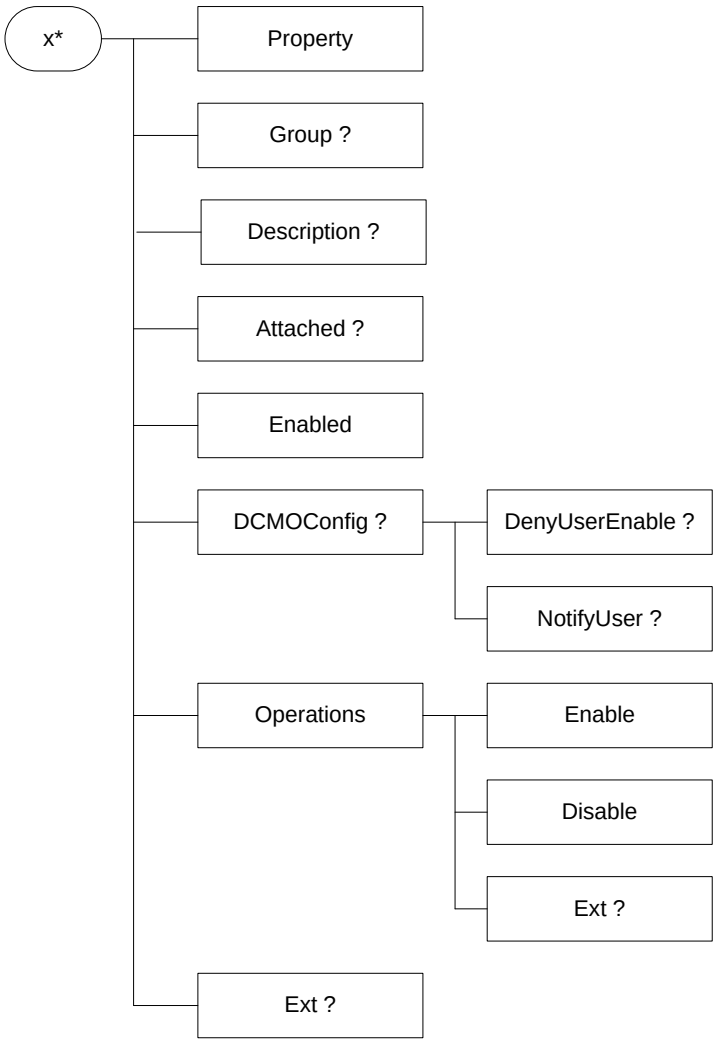


Figure 6-6: Device Capability Management Object – DCMO tree

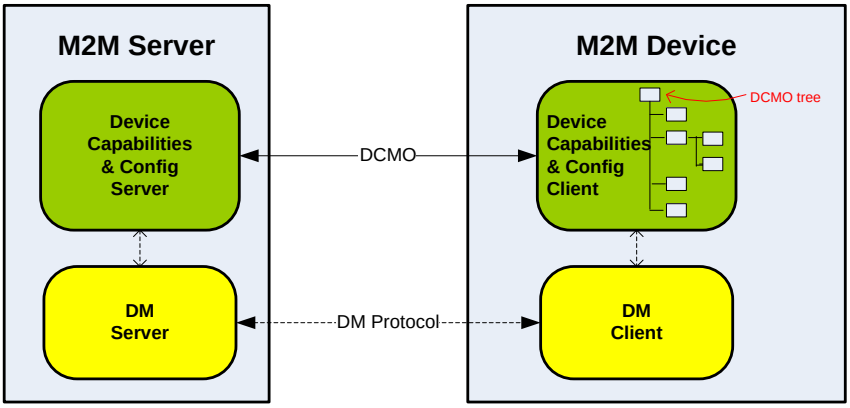


Figure 6-7: Device capabilities and configuration architecture

Optional nodes are marked with "?" sign. As M2M devices have no interaction with human, the node groups DCMOConfig are not used in EXALTED. The "Property" leaf node specifies the name of the device capability (e.g. GPS, GNSS, InputPeripheral, Bluetooth, Zigbee…). The optional "Group" leaf node contains the group name to which the device capability belongs to (e.g. Hardware, I/O, Connectivity or Software). A description can be optionally specified. The optional "Attached" node mostly not used for M2M Devices, however eHealth use case may need it. If this node exists in the management tree then the device capability is removable (extension that can be added or removed by user). The boolean "Enabled" leaf node indicates whether the device capability is currently enabled regardless whether it is attached or not. "Operations/Enable" and "Operations/Disable" nodes are used with DM Exec command to enable or to disable the device capability. This control enables service provider to control the level of service based on the subscription plan. Unsubscribed services or features can be remotely disabled. This business related remote control feature can help the device to save energy by turn off unnecessary device capabilities.

In addition to enabling and disabling a device capability, other configurations can be addressed by adding nodes in the "Ext" sub trees. For instance, an interior node may be added to specify properties related to sensor reading, as shown in Figure 6-8.
An interior node "Input" is added to the "Ext" sub tree, it is used to specify settings for any input peripherals, e.g. accelerometer, motion detection sensor, image sensor, temperature sensor, humidity sensor, any environment or eHealth related sensors. The "ReadingType" leaf node specify the sensor reading behaviour, the type could be:

0. Never, no automatic reading
1. On change, a new reading should be made available whenever the reading value has changed
2. Periodic, a new reading should be made available based on the value (second) specified in "Interval" node

The optional "interval" leaf node only exists if the ReadingType is set to 2 (periodic). This interval is expressed in number of seconds. Upon a new sensor value is read (ReadingType set to be 1 or 2), the "PostingType" specifies whether the sensor data should be posted, the type could enumerated as follow:

0. Not posted, the sensor data is retained in the device. The "Address" leaf node contains the URI of the local sensor data.
1. Parent, the sensor data is posted to the parent device, namely M2M Server or M2M Gateway or CH. For the 2 last cases, the "Address" leaf node contains the deviceID of the parent node and empty for the first case.
2. Topic, the sensor data is posted to a given message topic. The "Address" leaf node contains message topic name to which the sensor data will be posted to.

The latter message topic option enables the device to automatically post sensor data to a message topic hosted by a message queue server. Applications or devices (particularly

actuators) can subscribe to this message topic in order to receive sensor data. This mechanism helps to scale up the system, as some automation tasks can be distributed to the device domain without human interactions.

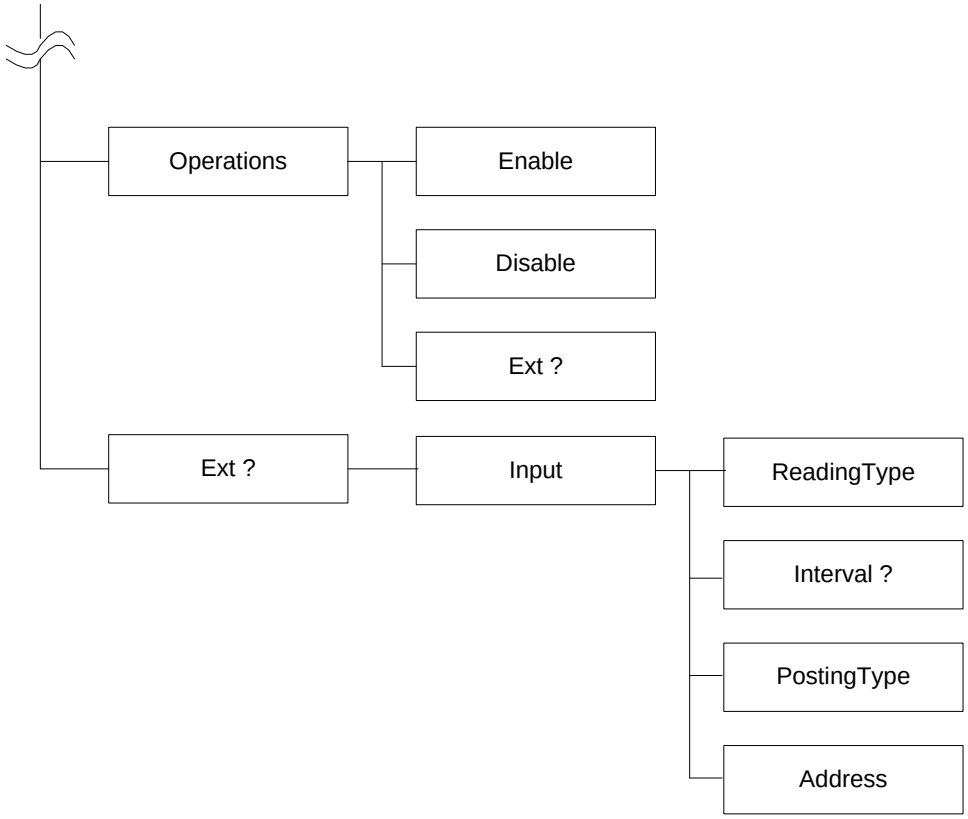


Figure 6-8: Extended DCMO

Other device settings can be addressed using this DC MO extension approach.

6.3 Service Capabilities

Beside DM functions described in the previous section, EXALTED lightweight DM protocols can be used to support other enabling services, such as, data transfer, device-to-device messaging, etc. New services can be added using the DM enabler architecture as shown in Figure 6-1. DCMO could be extended to support new services. This approach enables M2M Server to dynamically discover service capabilities supported in devices.

6.3.1 Data Collection

For energy efficient purposes, M2M Gateway has data aggregation capabilities. Based on the scenario, different data aggregation algorithms [6] developed within EXALTED can be leveraged to collect and aggregate sensor data. The data collection client enabler is responsible to aggregate data, to retain data and to post aggregated data to M2M Server. Data collection server enabler configures the counterpart client enabler over the data collection interface using the underlying EXALTED Lightweight DM protocol as shown in Figure 6-9. The enabler client sends the aggregated data to enabler server over the EXALTED Lightweight DM protocol. Collected data is then made available to M2M Applications through the data retention server. M2M Server may allow different M2M Applications to access to retention data server. With this approach, sensor data can be exploited by different M2M Applications without having the device to send the same data to each recipient application. The LTE-M bandwidth usage is thus optimized.

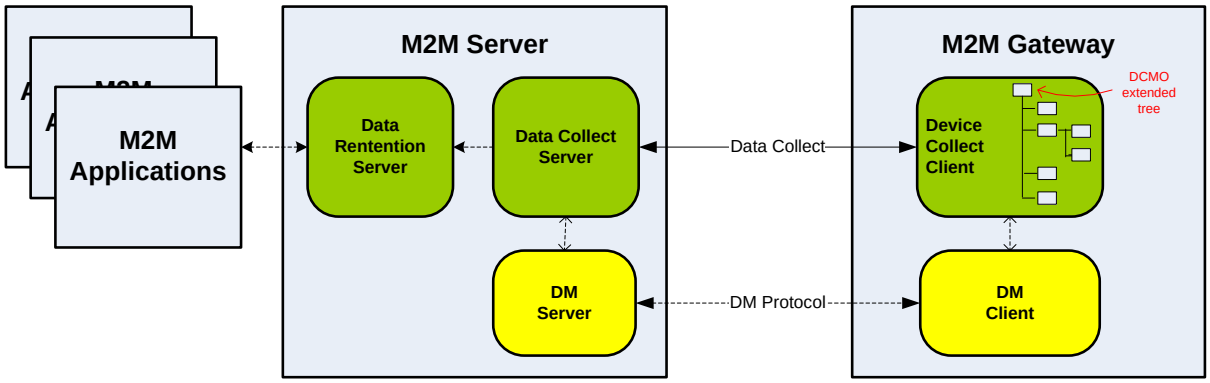


Figure 6-9: Data Collection Service capabilities

This Data Collection architecture is reflected in the global DM architecture (Figure 3-1 and Figure 3-2), where DC-1 is the internal interface between Data Collection Server and DM Server. FU-2 is the internal interface between Data Collection client and DM Client. Data Collection Client and Server interface with each other over the underlying DM Protocol (DM-1 and DM-2).

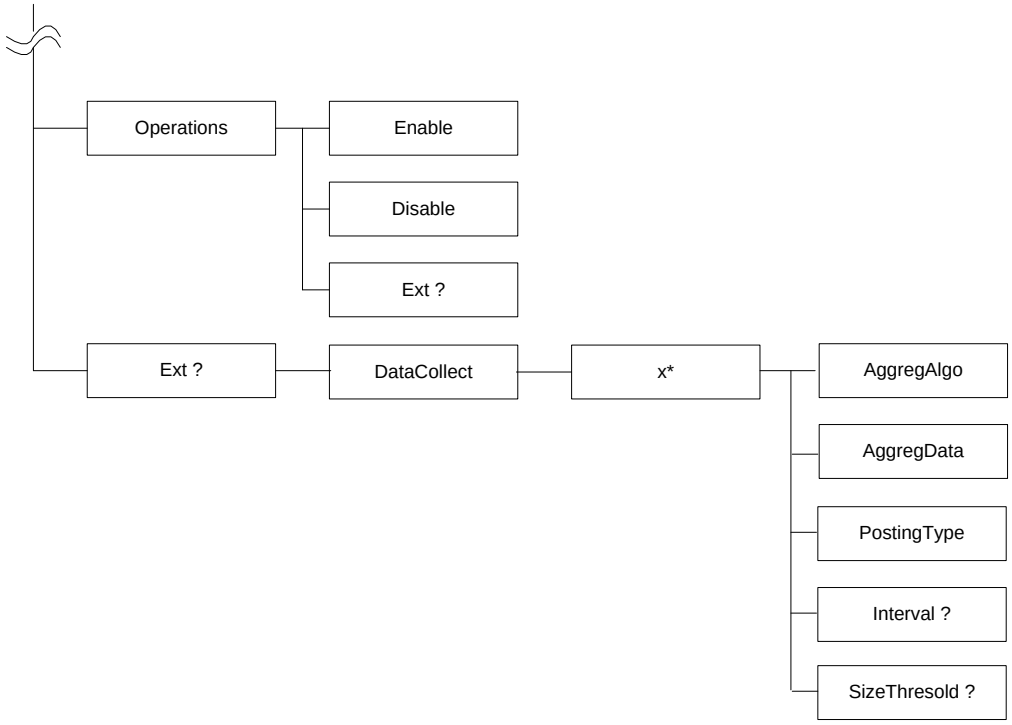


Figure 6-10 – Data Collection MO, extended from DCMO

OMA DCMO can be extended to support data collection as a service capability. Figure 6-10 depicts data collection related management object, extended from DCMO. An interior node “DataCollect” is added to the “Ext” sub tree, it contains interior nodes pertaining to each type of sensor data, e.g. /DCMO/DataCollect/Ext/DataCollect/Temperature is the placeholder to manage aggregated temperature sensor data, /DCMO/DataCollect/Ext/DataCollect/Humidity for humidity data collected from sensors. The “AggregAlgo” leaf node is used to configure the data aggregation algorithm [6] to be used. “AggregData” contains the aggregated sensor

data. “PostingType” specifies the data posting strategy, namely whether “Intervals” or “SizeThreshold” is used to determine when aggregated data is to be sent to M2M Server. “Intervals” and “SizeThreshold” are mutually exclusive. Aggregated data can be automatically sent to M2M Server based on 2 strategies: Periodic sending, the sending frequency is determined by the “intervals” node value expressed in number of seconds. Threshold sending, the aggregated data is automatically posted whenever the size of aggregated data exceeds the number specified in the “SizeThreshold” left node. This latter strategy is appropriate for scenarios in which sensor data is not time critical, the bandwidth usage can be optimized by specifying a proper threshold value.

6.3.2 Device-to-Device messaging

A simple device-to-device messaging mechanism can be enabled by defining a messaging management object. The messaging enabler is composed of Messaging Management Object (MsgMO) and Messaging client/server, as shown in Figure 6-11 and Figure 6-12.

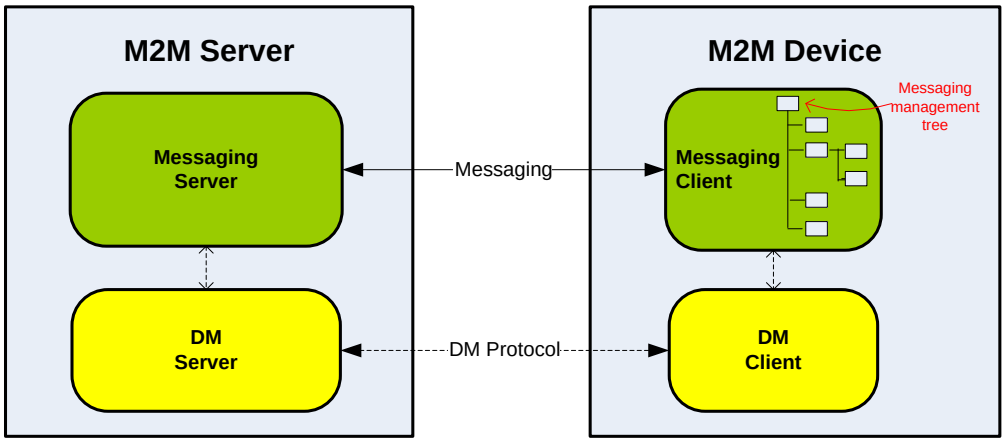


Figure 6-11: Messaging enabler architecture

This Device-to-device Messaging architecture is reflected in the global DM architecture (Figure 3-1 and Figure 3-2), where MSG-1 is the internal interface between Messaging Server and DM Server. MSG-2 is the internal interface between Messaging client and DM Client. Messaging Client and Server interface with each other over the underlying DM Protocol (DM-1 and DM-2).

Messaging MO is depicted in Figure 6-12, the ancestor element of “Msg” node defines the position in the DM tree of this MO.
“MsgCount” leaf node contains the number of message in the MO. Each message has an associated <x> interior node instance, accessible by the Message server. Each message has the following properties which are defined by the corresponding leaf nodes:
“SenderID” contains the device unique ID of the sender.
A description of the sender may be provided in the optional “SenderLabel” leaf node. This node may be used to search or filter messages, for example, in a semantic context.
“RecipientID” contains a unique identifier of the recipient device or group (groupID).
The message body is stored in the “MsgBody” leaf node. The content can be encrypted and binary data format can be Base64 encoded.

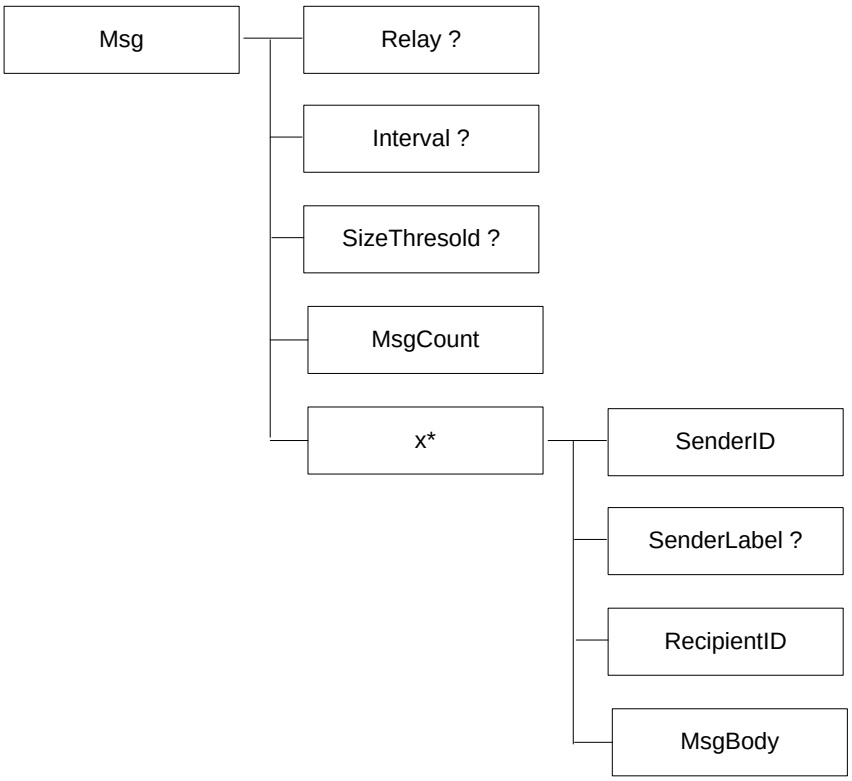


Figure 6-12: Messaging MO

Upon receiving an incoming message, the messaging client saves it onto a new instance <x> within the MO. The messaging client then determines whether the new message is intended to this device; by checking the “RecipientID” against the device ID of the device. It also checks whether the device is a member of a groupID specified in “RecipientID”. Messages intended to the device are consumed by the device application.
Messages that are intended to other devices will be relayed. The “Relay” leaf node specifies the message relaying strategy:

0. Relay immediately, for time critical application
1. Periodic relay, retained messages are aggregated and will be relayed at once to upstream or downstream device. For energy saving purposes, the interval between 2 relays is specified by the “Intervals” leaf node, expressed in number of seconds. This approach aims to optimize the duty cycle of the device
2. Retained messages can also be relayed based on a quota which is specified by “SizeThreshold” node value. Messages are queued, they will be aggregated and relayed at once whenever the size of queued message exceeds a threshold. This approach aims to optimize the bandwidth usage.

Further attributes may be defined to support advanced control and functions, such as TTL, topic name to support publish and subscribe messaging approach, etc. The messaging MO is limited to the above attributes set in this report to describe the concept.

Per OMA-DM enabler approach, sending a message requires the DM client to set one Replace command for each attribute through a URI in the request. With the current messaging MO, four Replace commands are necessary in an OMA-DM message. In ELFOMA (refer to section 4.2.1), these four attributes can be concatenated into one using Comma-Separated Values (CSV) formatting. Thus, only one Replace command is needed in the message.

A package having one message is represented as follow:

MsgMO = {Sender;Label?;Recipient;Message}

Furthermore, package containing more than one message is represented as:

MsgMO = {Sender;Label?;Recipient;Message}{Sender;Label?;Recipient;Message}{……..}{….} …{Sender;Label?;Recipient;Message}

Several messages can be aggregated into one single package. The transmission of bulk of messages only requires one Replace command in the message. This approach leads to a significant payload reduction.

Device-to-device messaging flow is depicted in Figure 6-13. Device dev1 sends a message to device dev2 via the gateway. The latter forwards the message directly to device dev2, as it is within the same capillary network. This message does not go through the DM server.

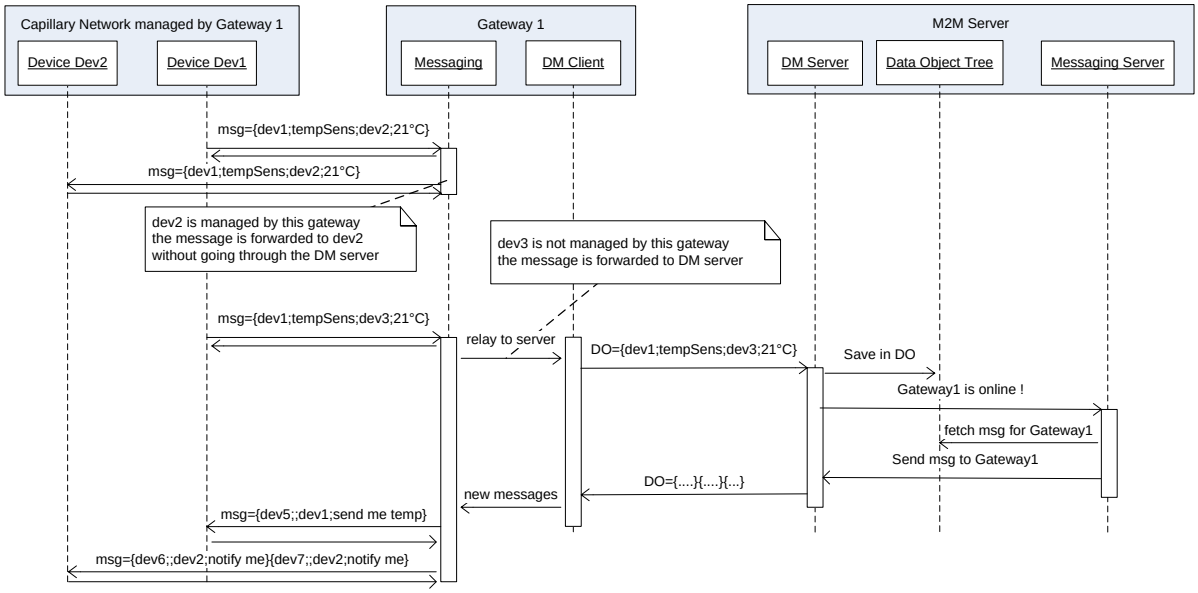


Figure 6-13: Device-to-Device messaging management flows

The workflow for Device dev1 to send a message to device dev3 via the gateway is as follows:

- Gateway1 is not aware of dev3, the message is forwarded to the DM server, over the DM protocol
- Upon receiving the message data object sent by Gateway1, DM Server saves it into the data object tree (database)
- DM server then informs the messaging server that Gateway1 is online. The messaging server search for pending messages, in the data object tree, that are intended to devices managed by Gateway1. All pending messages are sent back to gateway as a response over the DM protocol
- Gateway1 dispatches the messages to proper recipients.

Note that the message sent by dev1 and intended to dev3, is stored in the data object tree. It will be fetched by the messaging server and delivered to the gateway that manages dev3.

6.3.3 Addressing device behind gateway

Non-LTE-M devices deployed behind M2M Gateway may not have publicly routable IP addresses. In addition, in eHealth and ITS scenarios, these devices may move from one

capillary network to another one, thus detaching from a M2M Gateway and reattached to another M2M Gateway. This service capability aims to provide a global addressability to M2M devices, by using a unique and life time URI. This concept makes it possible for an entity to reach a M2M device regardless the type of capillary network to which it is attached to. This concept, as depicted in Figure 6-14 enables any type of M2M devices to be connected to the internet.

M2M Server uses GwMO enabler (described in section 8.) to track end devices. The Device Inventory MO contains information about an end device, such as to which gateway or CH it is attached to the type of the capillary network, the IP address of the gateway, etc (refer to section 8. for more details). Device Addressing server assigns a lifetime URI to each end device, hence to each device unique ID. Upon external request, the device addressing server map the provide URI onto a device unique ID. The targeted device can thus be located by searching for its device unique ID in the device inventory MO. The message can then be sent to the proper M2M Gateway which in turn will relay the message to the recipient in subsequent capillary network. For further information, refer to section 8.5.
It should be noted that the device-to-device messaging service, as described in the previous section, can be leveraged to map the incoming request onto device-to-device message which will be transmitted to the recipient over the EXALTED Lightweight DM protocol.

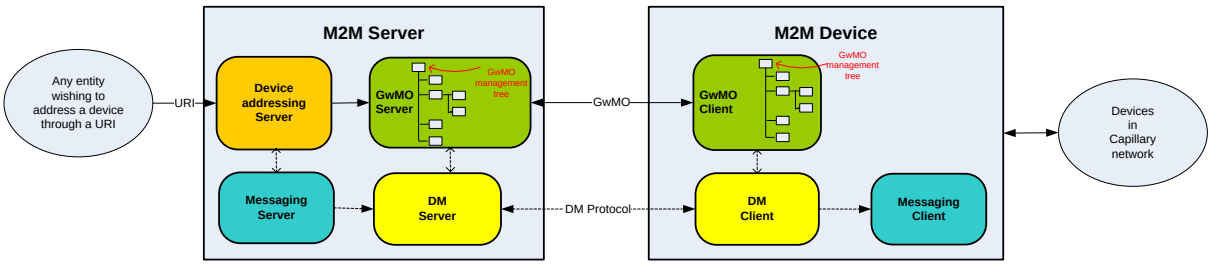


Figure 6-14: Addressing device using a public URI

Figure 6-15 depicts the workflow of device addressing capability in a mobility scenario.
Gateway MO server keeps track of device reachability information in the Gateway MO tree. This information is sent by the gateway upon device attaching or detaching to/from the gateway. The information contains the unique identifier of the device, type of end device, type of capillary communication, private address in the capillary network, etc. Refer to section 8.4 for further details.
In this example, the application is interfacing with the device over a CoAP protocol using the lifetime URI assigned to the DeviceC. The device capability set the message onto the GwMO leaf node pertaining to DeviceC. The device capability can also search for the DeviceC in the GwMO to determine whether it is reachable or not. The Gateway MO Server is informed about pending message to be delivered to gateway A, to which the recipient DeviceC is attached to. M2M Server may trigger the gateway to start DM session by the way of SMS. The message is pushed to gateway during the DM session, and delivered to DeviceC.
In the event DeviceC is not reachable (not attached to a gateway), the message is retained in the M2M Server. The message will be delivered to the gateway B to which the DeviceC will be attaching to.
The application always uses the same URI regardless where the DeviceC is located.

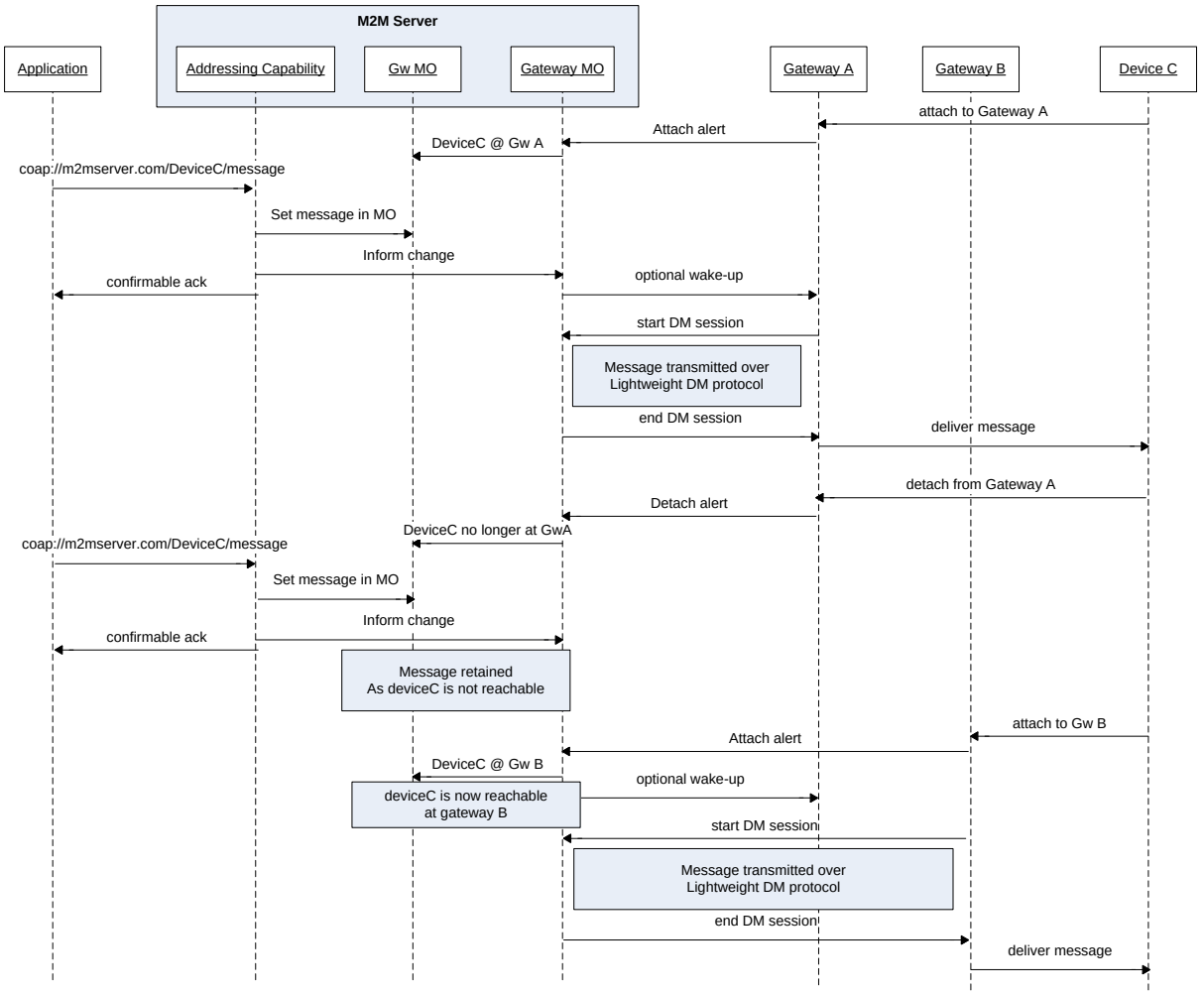


Figure 6-15: Device Addressing workflow in a mobility scenario

7. Capillary Network Structuring

7.1 Preliminaries

In EXALTED a capillary network of M2M devices requires Internet connectivity, which is provided by a gateway device via a link to the core LTE-M network. Apart from its protocol translation duties, the gateway is hence responsible for the delivery of all device traffic to the LTE-M, and distribution and correct targeting of all control traffic to the capillary network, besides any downlink data streams. Considering the large number of devices in the capillary network, this could easily overload the gateway, even if it is provisioned and designed to provide uninterrupted service to all devices in instances with high traffic demands. Therefore, it is desirable to reduce this traffic load coming from M2M devices as much as possible, so that the gateway can operate efficiently and support the network without any service loss.

The Exalted project addresses the aforementioned issue in several ways, two of them being *traffic aggregation* and *network structuring*. This section explains how a capillary network is structured for the purpose of making a large capillary network a manageable component for its responsible gateway, considering the fact that data aggregation and network structuring are inter-dependent tasks, i.e. forming a network structure also takes into account the necessity to perform data aggregation. Hence, such related aspects are also explained in the text, where necessary.

7.2 Distributed clustering of devices in the capillary network

EXALTED considers a hierarchical network structure with two levels in a capillary network. The higher tier is a collection of *cluster head* (CH) nodes which control their respective *member* nodes within their clusters. CHs are local control points that are responsible for various tasks, such as the management of devices in the cluster, delivery of control information, e.g. device-specific queries, and maintenance tasks like software update.
Furthermore, a hierarchical network structure provides scalability to the capillary network. Device querying, which is a difficult task for a single gateway device to perform in a large-scale network, can be done in a collective way when there are multiple responsible devices scattered in the network. A *backbone* of CHs simply routes the control traffic to related locations and devices. Hence, all these management tasks that are to be performed by a gateway node are distributed to the network, providing scalability by dividing the network into responsibility areas. This reduces the device management load on the gateway and provides faster response to management servers. Therefore, the capillary network is required to have a clearly defined protocol that provides the hierarchical structure of the network. With this structure in place, CH nodes can perform the above mentioned DM tasks.

Section 7.2 describes this protocol. The definition of this protocol is partially included in the public deliverable D4.4, Data Aggregation Protocols, since the protocol also explains how data aggregation is performed over this hierarchical capillary network structure. This document additionally provides in-depth results with respect to key protocol parameters.

7.2.1 Distributed Clustering of Heterogeneous Devices

Exalted envisions a capillary network as a collection of a high variety of devices with various characteristics, such as data generation rate, uptime, battery energy capacity, energy consumption rate, data processing speed (if any), etc. This provides high flexibility of EXALTED's solutions to networks with more specific properties, e.g. a capillary network of similar devices (homogeneous networks). The homogeneity can come from uniform communication capabilities, initial battery energy levels, or device hardware.

In the following, the document describes the **Dis**tributed **C**lustering **(DisC)** protocol, which considers a large capillary network of heterogeneous non-LTE M2M devices deployed in a wide area. Major mechanisms of DisC are briefly explained and some key performance results are provided to show its benefits.

7.2.2 DisC Protocol

DisC is a protocol that dynamically adapts the network's hierarchical structure via re-assignment of its CH nodes. The major parameters to the hierarchy construction problem are (i) multihop data paths to the gateway and (ii) variations in device properties. The protocol shares the common goal of all device clustering protocols: extension of network lifetime. In order to achieve this, DisC rotates the CH role among all network nodes. This is done in a distributed way, and collectively via message exchanges between nodes in a locality. Hence, DisC does not depend on a decision made by a single device that has to first collect all network statistics and then find an optimal solution and distribute the result to the network. Observing the fact that such protocols are not well-tuned and not much adaptable to dynamic networks, making local re-clustering decisions is the main strategy in DisC.

7.2.2.1 Improvement over the state-of-the-art

When compared to prior works in node clustering, the distinguishing property of the protocol is that it does not depend on time synchronisation among all network devices. Despite the current node clustering protocols that simultaneously start a network-wide re-clustering process in well-defined time periods [22][23][24][25][26][27], DisC leaves the decision to the network's current CH nodes and makes local changes when the need arises. Furthermore, DisC avoids various other assumptions that clustering protocols frequently make. For instance, global or local information on device locations, energy values, or inter-node distances is not available at any part of the network. Instead, in order to capture the most recent dynamicity in the network, DisC monitors traffic loads and residual energy levels by introducing minimal overhead to a typical ad hoc network protocol that normally exchanges neighbourhood messages. According to the updates, transmission ranges are adjusted so that cluster sizes can be selected in consideration of local traffic loads and node energy levels. This helps DisC prevent the emergence of network hot-spots and extend network lifetime.

7.2.2.2 DisC Preliminaries

The protocol considers multihop data delivery to the gateway and increasing traffic load on nodes as the hop distance to the gateway gets smaller. To adjust the cluster sizes, the hop distance to the gateway and the energy statistics collected in local neighbourhoods are taken into account.

To distribute the energy demanding CH role, DisC rotates the CH role among all network nodes. Nodes select a probability value that determines whether they will become a candidate for the CH role and get involved in a local election process based on statistical information. This process is triggered by currently assigned CH nodes, as will be explained in Section 7.2.2.7.

7.2.2.3 Protocol variables

The set of variables used by DisC for a node i are provided in Table 7.1: DisC variables.

Term	Definition
R	CH-CH transmission range
r(i)	i's transmission range as a CH or prospective CH
H(x)	Set of nodes that are x hops from the gateway
h(i)	i's hop distance to the gateway
$p_{h(i)}$	The baseline probability of becoming a CH in h(i)
P(i)	i's probability to become a CH

E(i)	Residual energy of i
$E_{cost}(i)$	Energy cost, as a CH, of i
S(i)	The set of all nodes in r(i) range of i
$S_{CH}(i)$	The set of all CHs in R range of i
$S_{prev}(i)$	The set of CHs that has selected i as its next hop
CHlist(i)	The list of CH neighbours of i
bmn(i)	The best member node in i's cluster with the highest residual energy/remaining lifetime.
qn(i)	The node that has sent a $BEACON_Q$ to i
CH_{def} (i)	The CH neighbour with the best SNR among all CH neighbours that have less hop distance to the gateway.
myCH(i)	CH of i
isCH(i)	Boolean indicating whether node i is a CH
nexthop(i)	Next hop CH of a CH node i
M(i)	Set of members of CH i's cluster
rate(i)	Generated data rate at node i
$rate_{out}(i)$	Outgoing data rate from CH i
$rate_{in}(i)$	Incoming data rate from CH i's members
$rate_{ave}(i)$	Average incoming data rate from CH i's members
$rate_{CH}(i)$	Incoming data rate from CH i's previous hop CHs

Table 7.1: DisC variables

7.2.2.4 Control packets in DisC

Figure 7-1 lists the packet types used in DisC, with details on the information content.

Packet Type	Purpose	Key Contents	Size (bits)	From → To	Range
$BEACON_Q$	Node status query	Query Type	64	Query Sender → Query Receiver	R
$BEACON_{-/+R}$	Node status reply	Response Type	64	Query Receiver → Query Sender	R
CHCHANGE	Transfer of CH role to another node	New CH's ID	80	CH i → members of i	$r(i)$
RECLUSTER	Notify cluster members of in-cluster CH reselection	$p_{h(i)}$	96	CH i → members of i	$r(i)$
RECLUSTERCH	Notify neighbour CHs of multi-cluster reclustering	-	64	CH i → CH Neighbours of i	R
JOIN	Request to join the cluster	$rate(i)$	96	New member candidate → CH i	$r(i)$
ACCEPT	Confirmation of cluster membership	-	64	CH i → New member candidate	$r(i)$
HELLO	Neighbourhood packet between all nodes	$E(i)$	128	Node i not in MCR → neighbours	$r(i)$
CHHELLO	Neighbourhood packet between CH nodes	$E(i)$, $\sigma_E(i)$, $\mathcal{E}(i)$	192	CH i not in MCR or ICR → CH neighbours	R
INIT	Initial broadcast packet from the gateway	-	64	Gateway → network nodes (broadcast)	R

Figure 7-1: Control packet types in DisC protocol

7.2.2.5 Neighbourhood messages

HELLO *packets:* Each node periodically transmits a HELLO message to notify its neighbours of its residual node energy. Each node j then updates the following:

- Local energy average: $\varepsilon(j) = \frac{1}{n(j)}\sum_{k=1}^{n(j)} E(k)\,, \quad k \in S(j)$,
- Standard deviation of local energy levels: $\sigma_\varepsilon(j) = \sqrt{\frac{1}{n(j)}\sum_{k=1}^{n(j)}(E(k)-\varepsilon(j))^2}\,,$
- Maximum local node energy level: $\varepsilon_{max}(j) = \max_{k\in S(j)} E(k)$.

CHHELLO *packets:* CH nodes periodically transmit a neighbourhood message, CHHELLO. With this message, a CH notifies its CH neighbours about its residual energy and most

recent outgoing data rate. Upon collection of CHHELLOs from neighbours, each CH j updates its local statistics:

- Maximum residual CH energy among all neighbour CHs: $\varepsilon_{CH_{max}} = \max_{k \in S_{CH}(j)} E(k)$,
- Average of the standard deviation values computed in each neighbour CH: $\overline{\sigma_\varepsilon}(j) = \frac{1}{n_{CH}(j)} \sum_{k=1}^{n_{CH}(j)} \sigma_\varepsilon(k)\,, \quad k \in S_{CH}(j)$,
- Average of the mean energy values E computed in each neighbour CH: $\bar{\varepsilon}(j) = \frac{1}{n_{CH}(j)} \sum_{k=1}^{n_{CH}(j)} \varepsilon(k)\,, \quad k \in S_{CH}(j)$.

7.2.2.6 Node states

A node *i* is found in one of the following four states:

1. *Member:* Node *i* is attached to a CH, collects data that it then delivers to its CH.
2. *CH:* The node may/may not have members, and it is not attached to another CH. It collects member data, summarises and forwards the data towards to the gateway over the CH backbone. The CH-to-CH communication range is R.
3. *Reclustering:* Node has started a re-clustering timer and is neither a CH nor a member node.
4. *Lonely:* The node is not attached to a CH, is not a CH itself, and is not involved in re-clustering.

Figure 7-2 below shows the state transition diagram of DisC. The transitions are marked by letters designating corresponding locations in the algorithms provided in Section 7.2.2.8.

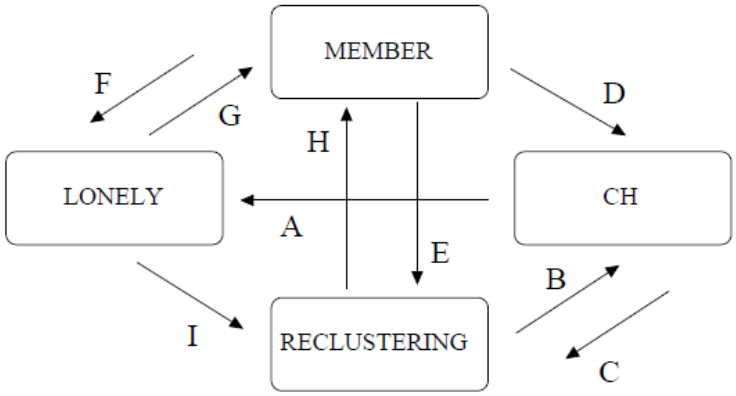


Figure 7-2: Node states.

7.2.2.7 Re-clustering mechanisms

DisC has two re-clustering mechanisms, both initiated by CH nodes, that help the network distribute traffic loads according to node positions and residual energy levels.

7.2.2.7.1 In-cluster Cluster head role Rotation (ICR)

ICR is the mechanism that rotates the CH role within a particular single cluster. Whenever a CH determines the need for transferring its role, which it decides based on energy statistics of the cluster, the CH initiates ICR. Each CH node *j* periodically checks the following condition in its clusters:

$E(k) > \theta_{IN} E(j), \quad k \in S(j)$, where θ_{IN} is a decision threshold. ICR is performed when *in-cluster timer* $t_{IN}(j)$ expires:

In-cluster Timer (t_{IN}): This timer controls when a CH should check the energy situation in its cluster in order to understand whether it is a suitable time to initiate an ICR. A constant value of T_{IN} is added when the timer is restarted.

CH j then picks the member k with the highest residual energy E(k). If the criterion is not met and no "better" member node is found, j resets its *in-cluster timer* as $t_{IN}(j) = t_{IN}(j) + T_{IN}$, and cancels ICR. This process of selecting a better member node is summarized in Figure 7-3.

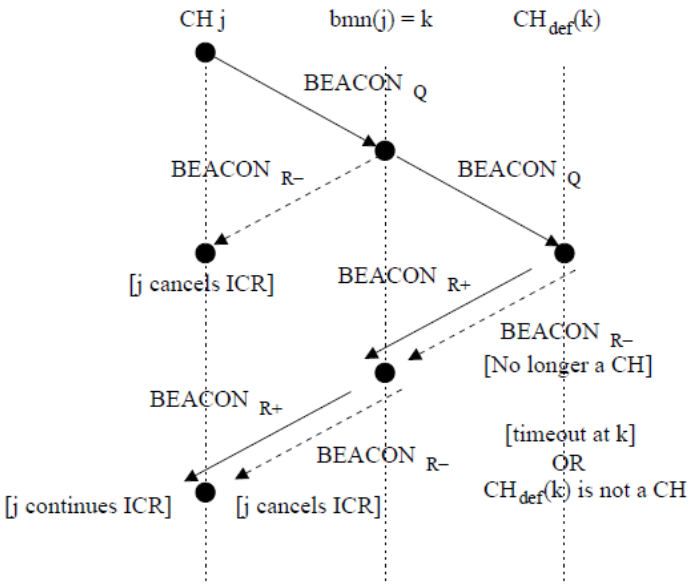


Figure 7-3: ICR's first stage: selection of a better member node.

Upon finding a better member, CH j continues with the transfer of its role to this node. Packet exchanges are performed, and the process is shown in Figure 7-4. The related packet types are listed in Figure 7-1.

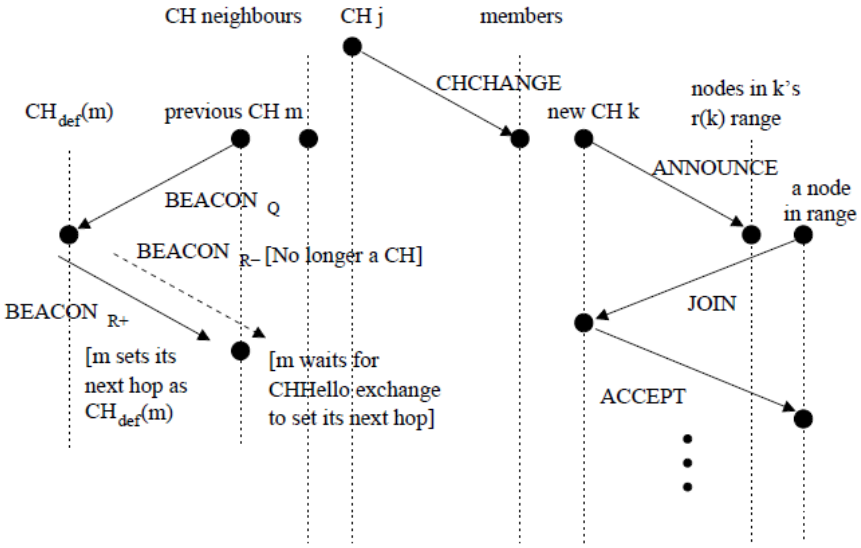


Figure 7-4: ICR's second stage: transferring the CH role.

7.2.2.7.2 Multi Cluster Re-clustering (MCR)

The second type of update of the network's cluster structure is MCR. This mechanism involves the neighbouring clusters of a CH as well, so as to reduce the energy imbalance in a wider area than what a single cluster can cover. Only the neighbouring clusters are triggered since starting a network-wide re-clustering process is costly in terms of the necessary extra time and energy required for transmitting extra control messages.

In DisC, a CH j initiates MCR, when $\overline{\sigma_\varepsilon}(j) > \theta_{STD}.\sigma_0$. Here, σ_0 is the standard deviation of the initial node energy levels of the network, which is a pre-set value at each node at deployment time, and θ_{STD} is a decision threshold.

The CH that initiates an MCR notifies its neighbour CHs. Then, nodes in the neighbour clusters, as well as the initiating cluster, go to RECLUSTERING state and collectively select new CH nodes. This is based on node timers whose expiry periods are calculated according to residual energy levels. Candidate nodes are determined based on a probability value given by:

$$P(i) = p_{h(i)} \frac{E(i)}{\varepsilon_{max}(i)}. \qquad (7.1)$$

The key property of DisC is that cluster sizes are dynamically adapted to preserve energy balance in the network. Whenever nodes update their probability values, they also pick a new value for their individual transmission range that is used when a node has the CH role, This range is effectively the cluster radius when the node is a CH. This is given by:

$$r(i) = \frac{2ln10}{\sqrt{\pi \sigma p_i}}. \qquad (7.2)$$

The CH selection probability P(i) of a node i depends on the baseline probability value $p_{h(i)}$ assigned to its hop distance h(i) to the gateway. This ensures a differentiation between hop distance-based CH densities, which is a measure towards energy equalization. This baseline probability $p_{h(i)}$ is updated according to the following equation:

$$p_{i_{new}} = \sum_{j=i-1}^{i+1} \overline{p_j} \frac{\overline{L_\iota}}{L_j} \left(\frac{N_j}{N_{i-1}+N_i+N_{i+1}}\right) \qquad (7.3)$$

The MCR mechanism is summarised in Figure 7-5. The related packet types can be found in Figure 7-1.

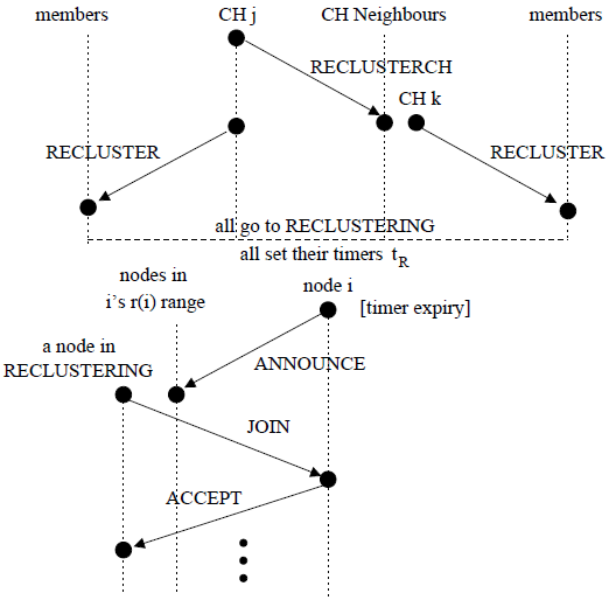


Figure 7-5: MCR mechanism

7.2.2.8 DisC Algorithms

Nodes in DisC operate according to which state they are, as shown in Figure 7-2. Individual state transitions are marked at the algorithms that are provided in this section.

Algorithm 1 CH state

```
1: if MCR criterion is met
2:     send RECLUSTERCH
3:     send RECLUSTER
4:     Update p, P, tR, and r
5:     Start tR
6:     Go to RECLUSTERING state C
7: end if
8: if Packet P received
9:     if P is RECLUSTERCH
10:        send RECLUSTER
11:        Update p, P, tR, and r
12:        Start tR
13:        Go to RECLUSTERING state C
14:    else if P is CHCANCEL
15:        if sender(P) is not nexthop
16:            Ignore P
17:        else
18:            nexthop = 0
19:            Select default neighbour CH CHdef in CHlist
20:            if default neighbour CH has been set
21:                send BEACON_Q to CHdef
22:            end if
23:        end if
24:    else if P is JOIN
25:        send ACCEPT to sender(P)
26:        set sender(P) as member
27:    else if P is BEACON_Q
28:        send BEACON_+R to sender(P)
29:    else if P is BEACON_-R or P is BEACON_+R
30:        if No BEACON_Q was sent to sender(P)
31:            Ignore P
32:        else
33:            if sender(P) is CHdef and P is BEACON_+R
34:                nexthop = sender(P)
35:            else if sender(P) is CHdef and P is BEACON_-R
36:                remove sender(P) from CHlist
37:                CHdef = best CH in CHlist
38:            else if sender(P) is not a CH and is bmn
39:                if P is BEACON_+R
40:                    Send CHCHANGE
41:                    Send CHCANCEL
42:                    isCH = 0
43:                    myCH = 0
44:                    Go to LONELY state A
45:                else if P is BEACON_-R
46:                    tIN = tIN + TIN
47:                end if
48:            else
49:                Ignore P
50:            end if
51:        end if
52:    else
53:        Ignore P
54:    end if
```

Algorithm 2 MEMBER state

```
1: if Packet P received
2:     if P is RECLUSTER
3:         Set p, P, r, and tR
4:         Go to RECLUSTERING state E
5:         Start tR
6:     else if P is CHCHANGE
7:         if I am not bestnode bmn in CHCHANGE
8:             myCH = 0
9:             Go to LONELY state F
10:        else
11:            Update P and r
12:            Send CHNOTIFY
13:            isCH = 1
14:            myCH = 0
15:            Go to CH state D
16:        end if
17:    else if P is BEACON_Q
18:        if CHdef == 0 and CHlist == ∅;
19:            send BEACON-R to sender(P)
20:        else if CHdef ≠ 0
21:            send BEACON_Q to CHdef
22:            qn = sender(P)
23:        end if
24:    else if P is BEACON_+R
25:        send BEACON_+R to qn
26:    else if P is BEACON_-R
27:        send BEACON_-R to qn
28:    else
29:        Ignore P
30:    end if
31: end if
```

Algorithm 3 RECLUSTERING state

```
1: if Packet P received
2:     if P is ANNOUNCE or P is CHNOTIFY
3:         Cancel tR
4:         Send JOIN to sender(P)
5:         Start tJ
6:     else if P is ACCEPT
7:         Cancel tJ
8:         myCH = i
9:         isCH = 0
10:        Go to MEMBER state H
11:    else
12:        Ignore P
13:    end if
14: end if
```

Algorithm 4 LONELY state

```
1: if Packet P received
2:     if P is RECLUSTER or P is RECLUSTERCH
3:         Set p, P, r, and tR
4:         Go to RECLUSTERING state I
5:         Cancel tJ
6:         Start tR
7:     else if P is CHHELLO from a CH i
8:         Insert i in CHlist (ID, SNR, h(i))
9:     else if P is ACCEPT
10:        Cancel tJ
11:        myCH = i
12:        isCH = 0
13:        Go to MEMBER state G
14:    else if P is CHNOTIFY or P is ANNOUNCE
15:        if Has not sent a ACCEPT to another CH
16:            Send JOIN to i
17:            Start tJ
18:        else
19:            Record sender(P) as CHdef and the SNR of P
20:        end if
21:    end if
22: end if
```

7.2.2.9 Performance Results

The DisC protocol is evaluated in networks with uniformly distributed node locations. Network dimensions are 400 units × 400 units and 800 nodes are deployed, corresponding to a node deployment density of $\sigma = 0.005\ nodes/unit^2$. Contention-based random access channel is considered. 10 different types of nodes with data generation rates of 500, 1000, 1500,…, 5000 bits/sec and an average per-bit processing energy cost of E_{bit} = 5 × 10−9J/bit that has a standard deviation of 10−9J/bit are considered. Initial node energy levels have an average of 2J and a standard deviation of 0.25J. For demonstration of results, we take measurements once in every time-unit, which is defined as the period of time required for 10 back-to-back link-level packet transmissions. The periodic neighbourhood messages, HELLO and CHHELLO are transmitted once in every 3 and 5 time-units, respectively. DisC is not dependent on a specific energy model, yet uses a popular one [28][29][30][31].

In this section, the performance of DISC is compared with the following settings:

1) ***No-Cl***: No clusters are formed and nodes deliver collected data to the gateway through multihop routes.
2) ***Fixed***: Clusters are formed, but cluster sizes are fixed and same at all CH nodes. CH role is not rotated among nodes and CH size is also not modified during network operation.
3) ***Variable***: This setting corresponds to the initial network structure used by DisC, which is not changed according to any energy-related local feedbacks. Cluster sizes are not necessarily same at different nodes and are assigned according to the initial estimation of network loads on different hop distance regions, and are not modified during network operation.

Figure 7-6 demonstrates the average node energy level over time. As it can be observed, DisC extends network lifetime compared to the three settings described above. MCR, which makes updates over multiple clusters, extends lifetime as well, however it cannot adjust to local changes in energy levels. In contrast, ICR captures local changes yet cannot address the variations that emerge within a single part of the network.

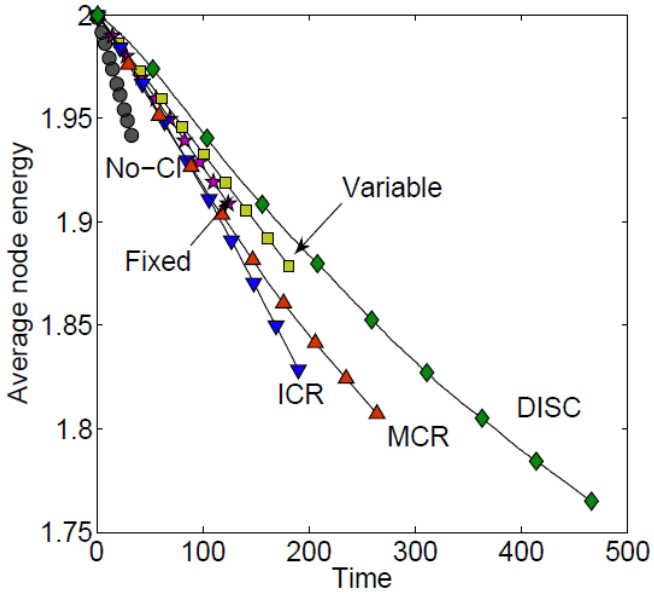


Figure 7-6: Average node energy over time.

In Figure 7-7, the standard deviation of energy levels is shown, which demonstrates the energy equalisation effect of DisC. Both ICR and MCR have a rapid rise showing that they cannot equalise energy levels on their own. ICR obtains lower levels as it makes a direct CH role transfer from a low-energy node to a high energy one; however differences across the network eventually emerge, which ICR does not address.

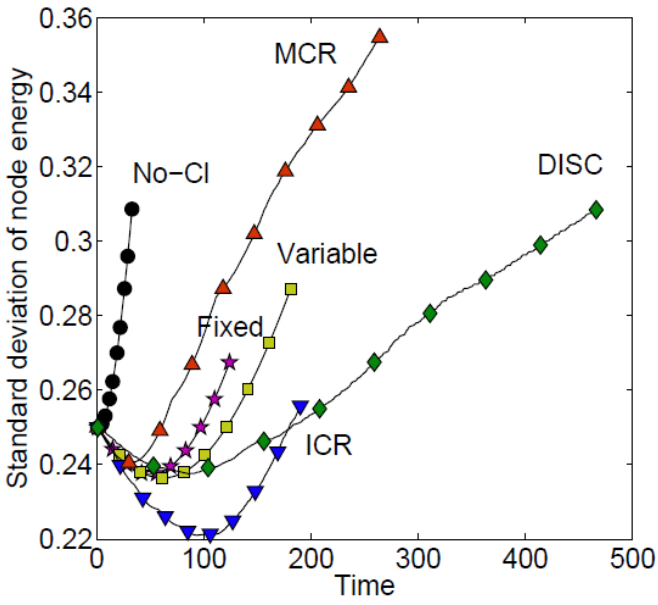
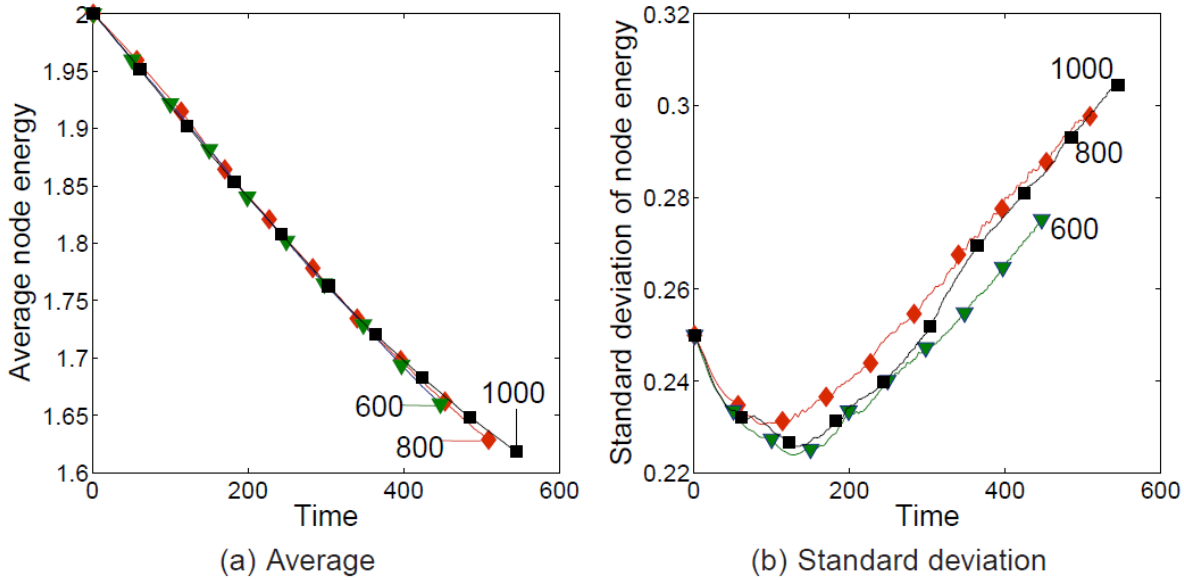


Figure 7-7: Standard deviation of node energy levels over time.

Figure 7-8 (a) shows that networks with different node density (as defined by different number of nodes, i.e. 600, 800, 1000, in the same area) have similar decline in average energy level, which indicates that the protocol consumes the resources of networks of different density in a similar way, without incurring a substantially different overhead. This can also be seen in Figure 7-8 (b), where the standard deviation levels have similar curves.

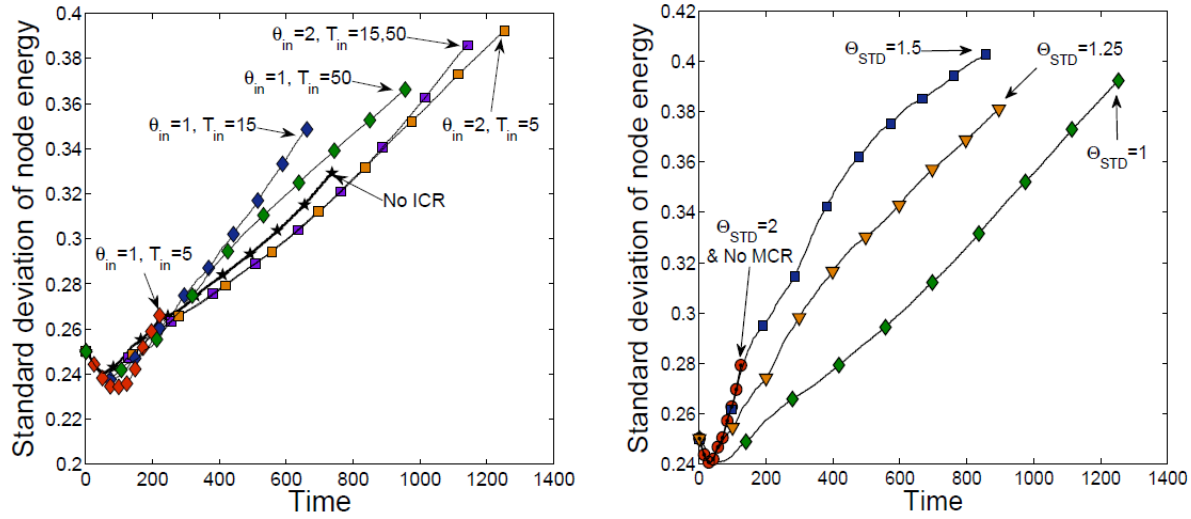
As node density gets less, network lifetime gets smaller, which can be attributed to the fact that larger transmission ranges are required so as to ensure communication on the CH backbone, which leads to larger transmission energy consumption.



(a) Average (b) Standard deviation

Figure 7-8: Scalability of DisC to network node density.

Finally, the effects of key protocol parameters are illustrated in Figure 7-9. As can be observed in Figure 7-9 (a), the best performance is observed when CHs in each cluster require that a candidate node have at least twice the energy level of its own for the CH to transfer its role to that node (θ_{IN}=2). Furthermore, the ICR check period, i.e. T_{IN} should be frequent enough (T_{IN}=5). Figure 7-9 (b) shows that in MCR, a CH should initiate the process of re-selecting CH nodes in the neighbourhood when the standard deviation is higher than the initial setting (θ_{STD}=1).



(a) T_{IN} and Θ_{IN} (b) Θ_{STD}

Figure 7-9: The effect of DisC thresholds on protocol performance.

7.3 Connectivity of capillary network backbone

The strategy of node clustering protocols is to determine the CH nodes either centrally or in a distributed way. In order to ensure the connectivity of the CH backbone, existing node clustering protocols pick a very large CH-to-CH range, either arbitrarily or based on the geometry of their considered network model. However, this causes CHs to consume excessively high transmission energy, even if a shorter range would provide the similar performance. Motivated by this observation, EXALTED provides an analytical computation method that determines a uniform backbone communication range R, which provides connectivity guarantees and at the same time reduces CH transmission power as much as possible. This section outlines this method. In doing so, the section provides the major criterion of selecting the transmission range of the capillary network's wireless backbone. This has critical implications from the DM point of view, as the direct results of using shorter/larger ranges in the backbone are:

1. The change in the delivery time of control and management packets to devices.
2. The number of CH nodes that are selected by the clustering protocol presented in Section 7.2.

Algorithm 1 Algorithm to compute R

1: $R \leftarrow R_0$;
2: $[prob] \leftarrow connect(\lambda, d, R)$;
3: **while** $prob < M$ **do**
4: $\quad R \leftarrow R + \Delta R$;
5: $\quad [prob] \leftarrow connect(\lambda, d, R)$;
6: **end while**
7: **return** R

Figure 7-10: Algorithm to compute the CH-to-CH communication range R

Figure 7-10 outlines the top-level algorithm that computes the value of R that ensures M% of E2E connectivity for a given distance d. The algorithm sequentially increments R until this connectivity is satisfied. As explained later, the distance d is varied to see the accuracy of this method for its different values. The main component of this algorithm is the connect procedure in lines 2 and 5. This procedure returns the probability of connectivity, *prob*, and it is called multiple times until *prob* < *M*.

7.3.1 Computation of the probability "prob"

Calculation of *prob* is based on considering the sequence of transmissions performed by CH nodes while a packet propagates towards the gateway. The combined E2E connectivity is modeled as the product of the individual single hop packet probability values of finding a next hop node towards the gateway. This is depicted in Figure 7-11. In this figure, area R_A is where a node with a shorter distance to the gateway is searched by node A, which represents the CH whose E2E connectivity to the gateway is sought. Node B represents the gateway. Area R_B is considered, so that no detour routes to the gateway are taken into account.

By considering the extreme cases in which the previous hop node to the gateway B are located at the far end locations X and Y of R_B (Figure 7-11(b)), distances to these two points reshape area R_A to make it smaller and change into R_{Next}. This provides a more accurate estimation.

The position of the next hop node A_{Next} of A within R_{Next} determines the distance d_{Next} to the gateway of this next hop node. Basically, the angle of deviation θ and the radial distance r as shown in Figure 7-11(c) are uniformly distributed random variables. Distance is estimated with:

$$d_{Next} \approx \sqrt{d^2 + \bar{r}^2 - 2\bar{r}d\cos(\bar{\theta})}, \qquad (7.4)$$

where $\bar{r}$ is calculated by:

$$f(r) = \frac{2\pi r\lambda e^{-\lambda\pi(R^2-r^2)}\,dr}{1-e^{-\lambda\pi R^2}},$$
$$\bar{r} = E[r] = \frac{R - \int_0^R e^{-\lambda\pi(R^2-r^2)}\,dr}{1-e^{-\lambda\pi R^2}}, \qquad (7.5)$$

and $\bar{\theta}$ is calculated by:

$$\bar{\theta} = E[\theta] = \frac{\int_0^{\beta} \theta e^{-\lambda\left[\theta\bar{r}^2 - \bar{r}^2 sin(2\theta)\right]/2}\,d\theta}{\int_0^{\beta} e^{-\lambda[\theta\bar{r}^2 - \bar{r}^2 sin(2\theta)]/2}\,d\theta}, \qquad (7.6)$$

where $\theta \in [0\ \beta]$, as shown in Figure 7-11(c).

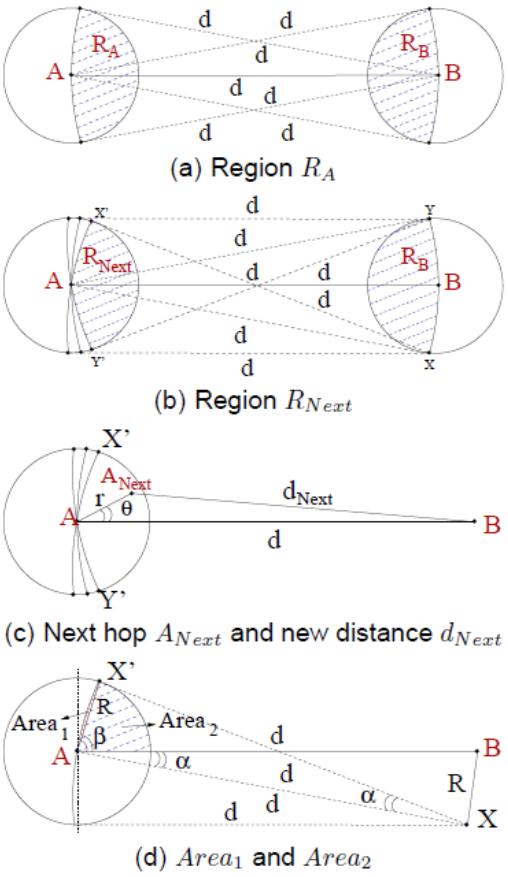


Figure 7-11: Locating the next hop node towards the gateway.

Figure 7-11(d) illustrates calculation of the area of R_{Next}, which is divided into two manageable sectors, $Area_1$ and $Area_2$, calculated by:

$$\begin{aligned} Area_1 &= d^2\alpha/2 - \sqrt{s(s-d)^2(s-R)}, \\ Area_2 &= R^2\beta, \text{ and } \beta = (\pi - 3\alpha)/2, \end{aligned} \tag{8.7}$$

where s is the half circumference of the triangle AX'X and Heron's formula is used to calculate its area.

Although we use an approximation method given by Equation 8.4, the position of the next hop not only determines d_{Next}, but also the rest of the path. Especially when long paths are traversed towards the gateway, the locational probability of all next hop nodes gets extremely complicated, while the search area of finding these nodes gets larger. To reflect this effect, the approximation method illustrated in Figure 7-12 is utilised. With this method, we have:

$$R_{new} = ((2i-3)\bar{r} + 2R)\bar{r}\bar{\theta}. \tag{8.8}$$

Then, we consider the probability to find at least one node in R_{new} at each hop while updating the E2E probability *prob*.

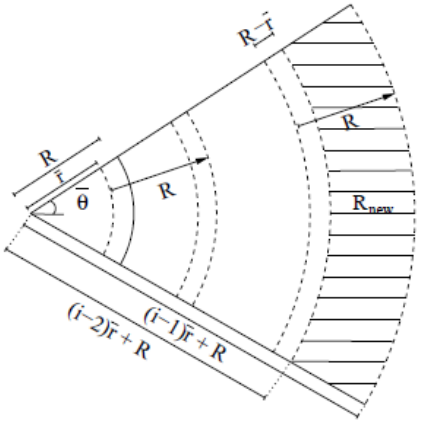


Figure 7-12: Areas to search next hop nodes.

Figure 7-13 is the complete algorithm that computes the probability value *prob* and covers the above-mentioned calculations. The algorithm loops until the current value of d gets less then R, at which point the gateway has been reached. Line 3 is for $\bar{r}$, lines 6-11 are for d_{Next}, lines 12-16 are for the area of R_{Next}, and finally lines 17-21 show the update of *prob* in each iteration of the loop.

Algorithm 2 Procedure $connect(\lambda, d, R)$

$prob \leftarrow 1$;
$K \leftarrow 0$;
$\bar{r} \leftarrow E[r] = \frac{\int_0^R 2\pi r^2 \lambda e^{-\lambda\pi(R^2-r^2)}\mathbf{dr}}{1-e^{-\lambda\pi R^2}}$;
while $d > R$ **do**
  $K \leftarrow K + 1$;
  $\alpha \leftarrow 2sin^{-1}(R/2d)$;
  $\beta \leftarrow (\pi - 3\alpha)/2$;
  $C \leftarrow \int_0^\beta e^{-\lambda[\frac{\theta r^2 - r sin(2\theta)}{2}]}\mathbf{d}\theta$;
  $\bar{\theta} \leftarrow \frac{1}{C}\int_0^\beta \theta e^{-\lambda[\frac{\alpha r^2 - r sin(2\theta)}{2}]}\mathbf{d}\theta$;
  $d_{Next} \approx \sqrt{\bar{r}^2 + d^2 - 2\bar{r}dcos\bar{\theta}}$;
  $d \leftarrow d_{Next}$;
  $s \leftarrow (2d + R)/2$;
  $a \leftarrow \sqrt{s(s-d)^2(s-R)}$;
  $Area_1 \leftarrow d^2\alpha/2 - a$;
  $Area_2 \leftarrow \frac{R^2\beta}{2}$;
  $Area(R_{Next}) \leftarrow 2(Area_1 + Area_2)$;
  if $K == 1$ **then**
    $prob \leftarrow (1 - e^{-\lambda Area(R_{Next})})prob$;
  else
    $Area(R_{new}) \leftarrow ((2K-3)\bar{r} + 2R)\bar{r}\bar{\theta}$
    $prob \leftarrow (1 - e^{-\lambda Area(R_{new})})prob$;
  end if
end while
if $d > 0$ **then**
  $K \leftarrow K + 1$;
end if
return $prob$

Figure 7-13: Procedure "connect".

7.3.2 Simulation results of range R

In this section, first the distance d that provides an accurate estimation of R that gives an M% connectivity probability is determined. To achieve this, estimation results are compared to results of Monte-Carlo simulations for node deployment density values of $\sigma = 0.003, 0.004, 0.005, 0.006\ nodes/unit^2$, in $X\ x\ X$ networks with $X = 1000\ units$. $R_0 = 10$ and $\Delta R = 1$. The target CH deployment density λ is 0.1 that of the node density σ. The gateway is located at the centre of the topology, coordinates (500,500).

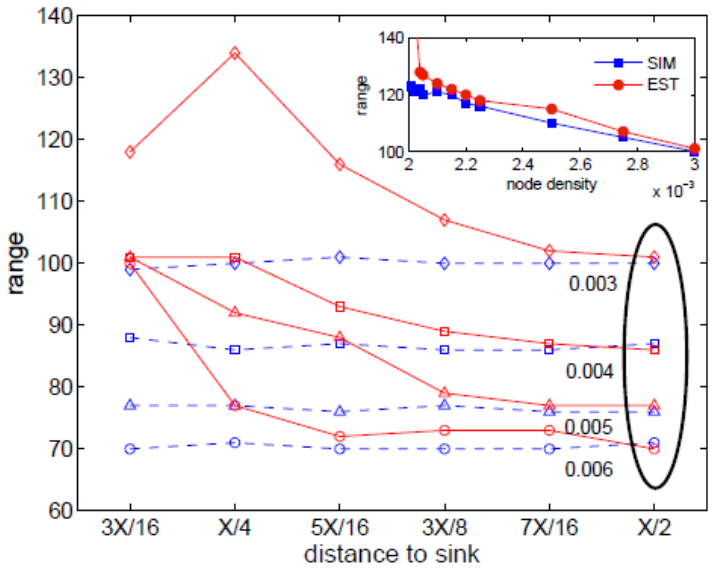


Figure 7-14: Evaluation of estimation accuracy for different values of d.

Figure 7-14 illustrates that when d is set to the maximum horizontal distance, which is X/2, the accuracy is the highest. Hence, the estimation method sets d = X/2 for a $X \times X$ network.

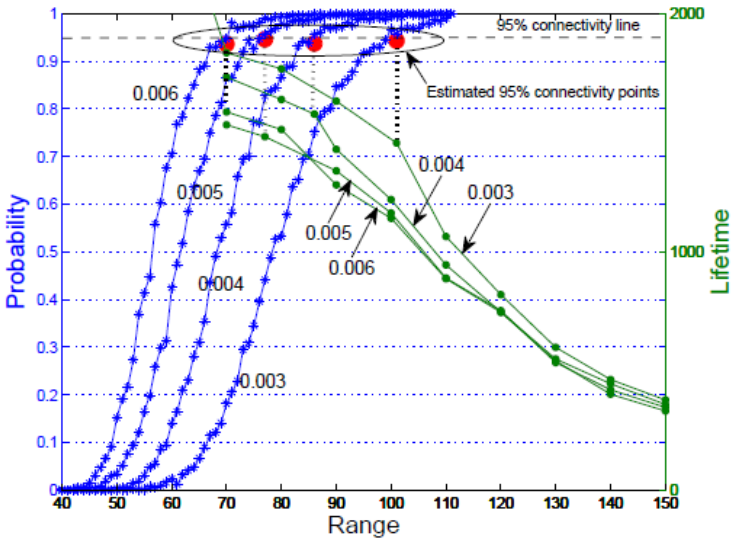


Figure 7-15: Probability of connectivity and network lifetime.

Figure 7-15 shows the connectivity probability for various values of range R (inclining graphs) and the resulting network lifetime by selecting these range values (declining graphs). The estimated M=95% connectivity points are depicted by red dots in the figure. It is observed that the estimation of these points are highly accurate when compared to the M=95% connectivity line. Furthermore, the lifetime of the network monotonically decreases when a larger range R is selected for the communication of the backbone CHs. Hence, picking an arbitrarily large network is destructive while an arbitrarily small R does not provide sufficient backbone connectivity due to rapid partitioning of the network.

7.4 Concluding remarks

In this section, the DisC protocol is presented, which forms a hierarchical capillary network structure. CH nodes in DisC are dynamically changed in order to rotate the energy demanding role of being a CH. Two mechanisms are proposed: ICR and MCR. ICR transfers CH role locally within a cluster, while MCR initiates a region-wide re-selection of CHs based on certain probability values that are adjusted according to residual node energy levels and latest traffic conditions. Nodes in DisC exchange neighbourhood messages to update energy statistics and adjust cluster sizes to network conditions. As opposed to existing node clustering protocols, DisC does not require network-wide clock synchronisation and considers the heterogeneous characteristics of M2M networks instead of homogeneous settings that are widely seen in wireless sensor networks. Results demonstrate that node clustering is beneficial to extend network lifetime. Furthermore, a combination of ICR and MCR can provide the best performance results, which supports the need for updating cluster sizes in a dynamic environment.

The assignment of suitable transmission ranges in the backbone of a hierarchical network play a fundamental role in energy savings in a capillary M2M network. Instead of picking an arbitrary large value which would cause excessive energy consumption, or an arbitrarily small value, which would be insufficient for high backbone connectivity, DisC uses an estimation method before network operation time and assigns a suitable value of uniform CH-to-CH communication range to devices.

8. Managing Capillary Devices

The previous section 7. details how Non-LTE-M devices can dynamically form capillary networks with self-organizing capabilities. Clusters are thus formed by electing dynamically and automatically CHs. This self-organization approach aims to extend device battery life and to enhance spectrum efficiency via traffic aggregation.
Provided the capillary network has been structured, this section covers how M2M Server can manage Non-LTE-M devices within the capillary network.
In this entire chapter, end devices refer to Non-LTE-M devices.

8.1 Principle

Per EXALTED system architecture concept (section 2.3), Non-LTE-M devices do not have direct communication with M2M Server. They communicate with M2M Server through the gateway. Likewise, M2M Server manages indirectly Non-LTE-M devices behind the gateway. M2M Gateway is thus managed by M2M Server using ELFOMA. However, the gateway does not use this protocol to manage Non-LTE-M device. Protocol adaptation is therefore needed at the gateway.
M2M Server leverages ELFOMA solution developed within EXALTED (detailed in section 4.2.1) and OMA Gateway Management Object [39] to manage capillary devices deployed behind M2M Gateway. This approach extends OMA GwMO usage beyond the single level hierarchy of child end devices, to include multi-level hierarchy level of CHs and end devices. DM related signalling is reduced over the LTE-M link. The following DM functions can be fulfilled: Device bootstrapping, device configuration, software component management and software update. In addition, the gateway has the ability to keep track of devices being attached to it. The gateway has therefore the ability to multicast commands to a group of devices.

8.2 Functional components and interfaces of Non-LTE-M devices

Device Management architecture diagram Figure 3-2: Indirect Management of non-LTE-M Devices through Gateway, can be further expended as shown in the figure below. Figure 8-1 shows the different capillary network topologies that the gateway needs to be considered in DM:

1. Non-LTE-M devices are managed directly by Gateway without CH
2. Non-LTE-M devices are managed by CHs
3. For the latter case, CHs may be organized in a hierarchy structure, a chain of linked CHs

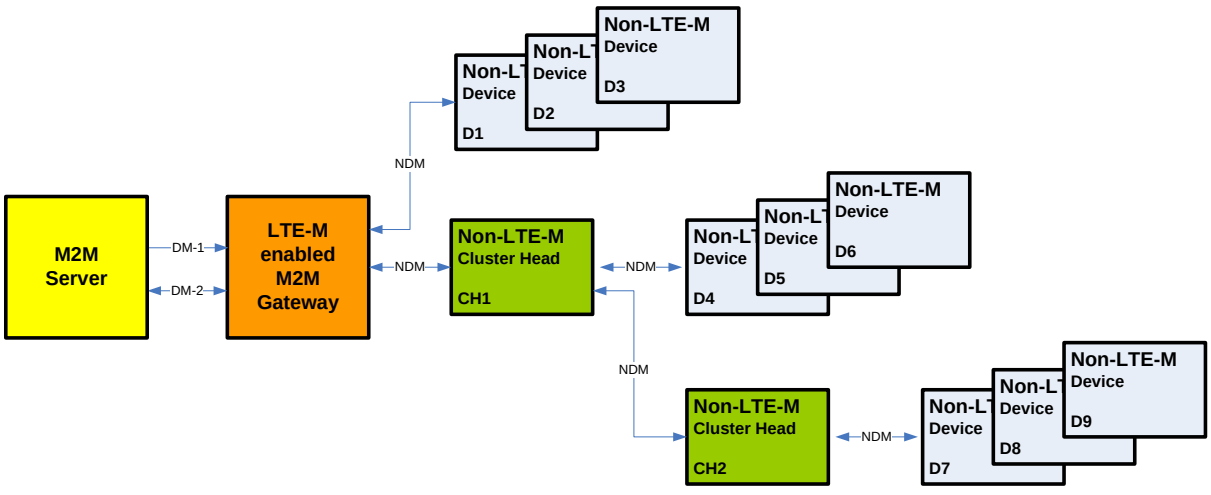


Figure 8-1: DM consideration in capillary network

Gateway, CHs and end devices communicate with each other over the NDM interface. This latter does not implement EXALTED Lightweight DM protocols developed within EXALTED. For interoperability purpose, various device native protocols (e.g. Zigbee, Bluetooth, KNX…) implement the same NDM interface over the corresponding air interface.

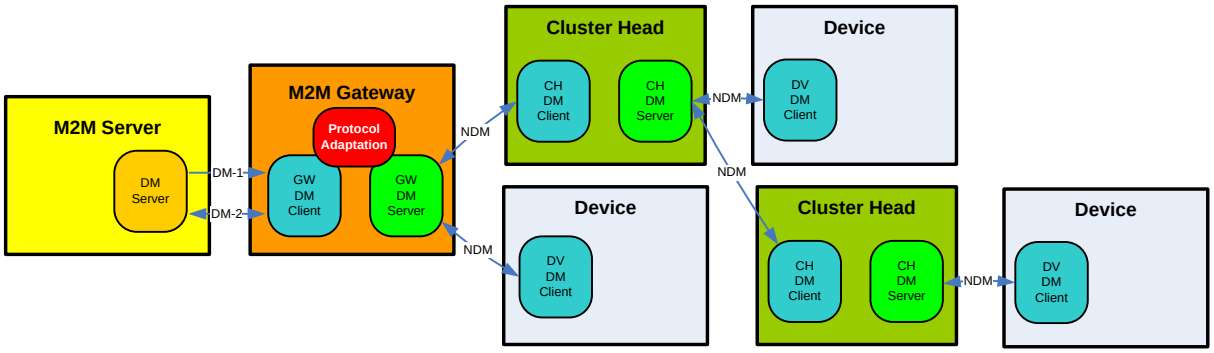


Figure 8-2: DM client and server components in capillary devices

DM client and server components (as shown in Figure 8-2) expose functionalities on the same NDM interface. It should be noted that for clarity purposes, application component and service component are not depicted in Figure 8-2.

CH DM Client and Device DM Client expose elementary DM functions that can be invoked over the NDM interface, for instance:

- doAction along with function selector and parameters. The function selector is used to define the action to be performed, such as, bootstrapping, update firmware, install/remove software, self-diagnostic, reset, reboot, message available, turn light on/off, read meter, etc. Provided parameters specify the target device/group, input arguments for requested action.
- getStatus along with function selector and parameters. The function selector is used to specify the component whose status is to be requested.
- setConfig to configure the device. Selector and parameters can be used to specify a particular component to be configured.
- getConfig to retrieve the current configuration. Selector and parameters can be used to specify a particular component to be retrieved.

Note that other control functions not related to DM are not listed.
Upon invocation of one of the aforementioned functions and depending on the specified targeted, CH DM client may:

- Execute the requested function locally (e.g. configuration of the CH)
  or/and
- Fan out the requested action to devices that are managed by the CH

To achieve the latter fan out process, CH's DM Server subsequently calls the same requested function towards all managed devices' DM Client. This function cascading call is only performed by CHs, end device does not. A top level DM function call made by M2M Gateway can thus be recursively propagated through CH chains down to all end device leaves.

CH DM Server and Gateway DM Server expose elementary DM functions with uniform interface that can be invoked by CHs DM Client or by end devices DM Client, over the NDM interface, for instance:

- Notify function in order to notify the DM server of an event. For instance, device attach or detach event. In the former case, a list of devices is passed as parameters to the server along with other information, for instance, device ID, associated CH ID.

This function can also be used as a response to a previous call to doAction, in order to return asynchronously the execution status, or requested status.

- getData along with a selector and parameters. The function selector specifies the type of the requested data. Parameters can be provided for further filtering purposes. For instance, DM Client (end devices or CHs) can request firmware or software image, pending messages, self-diagnostic or self-healing rules, etc
- setData along with a selector and parameters. The function selector specifies the type of data to be set or posted. Parameters contain sensor and also specify for instance, the recipient to whom the data is intended to, e.g. posting device-to-device message, smart meter data or other sensor data.

It should be noted that other control functions not related to DM are not listed.
Upon invocation of one of the aforementioned functions and depending on the specified recipient, DM Server within CH or M2M Gateway may:

- Execute the requested function locally (e.g. update device inventory upon receiving device attach event)

  or/and

- Relay the requested action to parent node (e.g. report the device attach event to parent node; a parent CH or M2M Gateway)

To achieve the latter recursive relay process, CH's DM Client calls the same requested function exposed by parent node's DM Server. This function upstream recursive call is only performed by CHs and M2M Gateway. The information may be relayed up to M2M Server by the M2M Gateway. A bottom level DM function call made by an end device can thus be recursively propagated through CH chains up to M2M Gateway and ultimately to M2M Server. To achieve energy and spectrum efficiencies, CHs and M2M Gateway may momentarily retain the collected information. The upstream relaying is postponed to a later stage to take advantage of the data aggregation process. This approach is depicted in Figure 8-3

In Figure 8-3, provided that a Non-LTE-M device is elected to be CH2. As result of this self-organizing, Device D1 and Device D2 are managed by CH2, which in turn is managed by CH1. And this latter is managed by the gateway. D1 and D2 both send data to their associated Application App1. Sensor data is posted to CH2 by invoking setData function exposed by DM Server of the targeted CH over the device legacy protocol. CH2 redirect the function call to upper parent node's DM Server. This relay call has the same signature and is performed by CH2's DM Client. As CH1 has the same behaviour as CH2, the function is now relayed to the upper parent node which is the M2M Gateway. Collected data is retained and aggregated at the gateway. To enhance the spectrum efficiency the aggregated data will be sent to M2M Server at later stage. M2M Server collects the data and made it available to the owner Application App1. Only App1 is allowed to retrieve this set of data.

It should be noted that the aforementioned DM functions are to be implemented as request/response data frames and transmitted over the native protocol of the device (e.g. Zigbee, Bluetooth…).

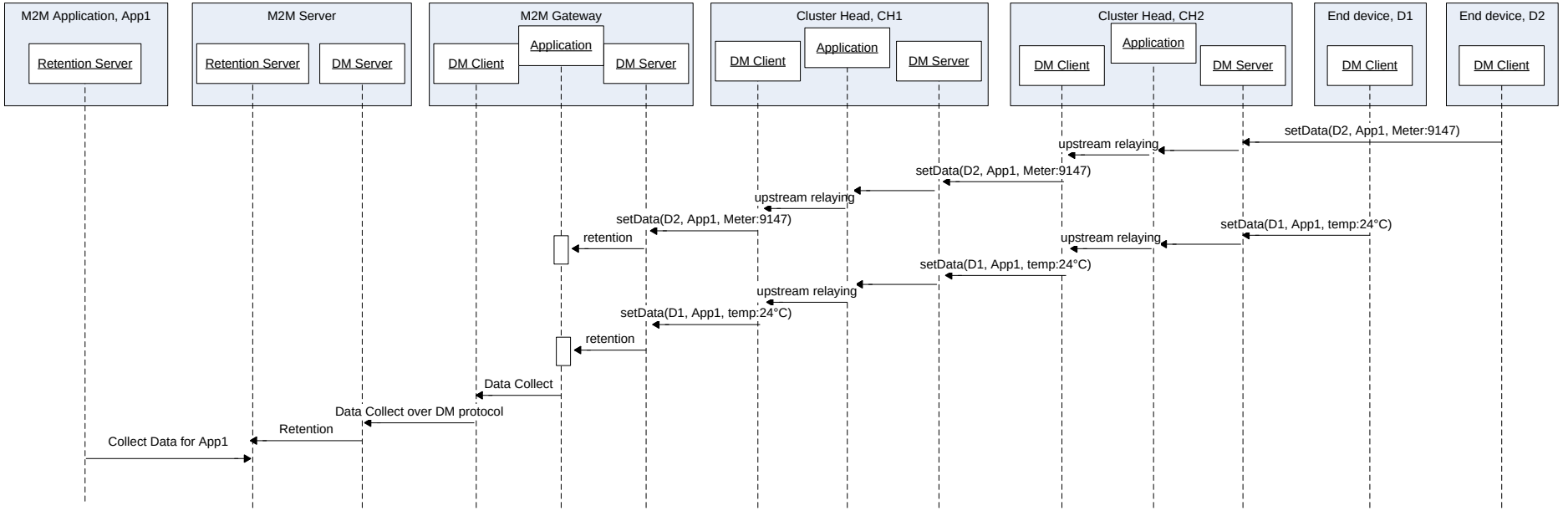


Figure 8-3: Recursive message upstream escalation in capillary network

8.3 DM Protocol Adaptation

M2M Gateway is managed by M2M Server over the EXALTED Lightweight DM protocol. And M2M Gateway, in turn, manages Non-LTE-M devices over native device protocols. Therefore a DM protocol adaptation component is required in the gateway as depicted in Figure 8-2. The main functionality of this component consists in translating the incoming DM commands (received from M2M Server) onto outbound function calls along with parameters. Function calls are described in the previous section 8.2. Gateway's DM Server invokes these functions exposed by the end device's DM Client over the NDM interface.

8.4 Device Inventory

Before any DM actions can be performed by M2M Server, end devices must be detected, paired then registered to a M2M Gateway. OMA Gateway Management Object enabler [39] has defined various MOs for realizing DM functions in M2M Gateway. Device Inventory MO resides in the management tree of the M2M Gateway and it maintains a list of devices in the capillary network that are managed through the M2M Gateway. This MO is updated when the M2M Gateway becomes aware of new Non-LTE-M devices being detected and attached to the capillary network, or the M2M Gateway becomes aware that a previously subtending end device is no longer present in the capillary network. In accordance with the approach described in the previous sub-section, end devices can use the "notify" function to report "device attach" or "device detach" event to the parent CH node, which in turn will propagate this notification up to the M2M Gateway. This event notification could carry useful information such as device ID, device type, CH to which the device is attached, network type, etc. The Device Inventory MO can therefore be updated accordingly at the gateway to reflect the actual fleet of devices within the capillary network.
During a DM session started by the M2M Gateway, M2M Server can remotely query this Device Inventory MO over the EXALTED Lightweight DM protocol. Consequently, M2M Server becomes aware of Non-LTE-M devices deployed behind a M2M Gateway. From this point, M2M Server can start managing a particular end device or a group of end devices.

A pictorial representation of the Device Inventory MO is given in Figure 8-4: Device Inventory MO.
This MO tree is persisted in the M2M Gateway. Information in this tree can be retrieved by M2M Server, over the DM protocol. The "DevCount" leaf node gives the number of Non-LTE-M end devices being managed by the gateway. Each end device has a placeholder node "<x>/Inventory/Records/<x>" to store its connectivity information such as, Device ID, device type, type of address (e.g. Non routable, IPv4, IPv6), device address (e.g. Zigbee address for non-routable address type), reference to the LAN in which the end device is deployed (e.g. Zigbee within cluster CH2). As mentioned in 8.1 and 8.3, DM protocol adaptation is required by the gateway to manage Non-LTE-M devices, therefore the "Mode" should be set to Adaptation mode.

The "LAN" node lists all available LAN in the capillary network. Based on the information provided by "device attached" event, M2M Gateway can create a placeholder node for each cluster. For instance, "<x>/Inventory/LAN/CH1" and "<x>/Inventory/LAN/CH2" placeholder nodes contain connectivity information of clusters managed by cluster head CH1 and cluster head CH2. The type of network can also be specified within these placeholder nodes. For example:

LAN/CH1/NetworkType = 3 (Zigbee)
LAN/CH2/NetworkType = 2 (Bluetooth)

Heterogeneous aspect of capillary network can thus be handled by the M2M Gateway.

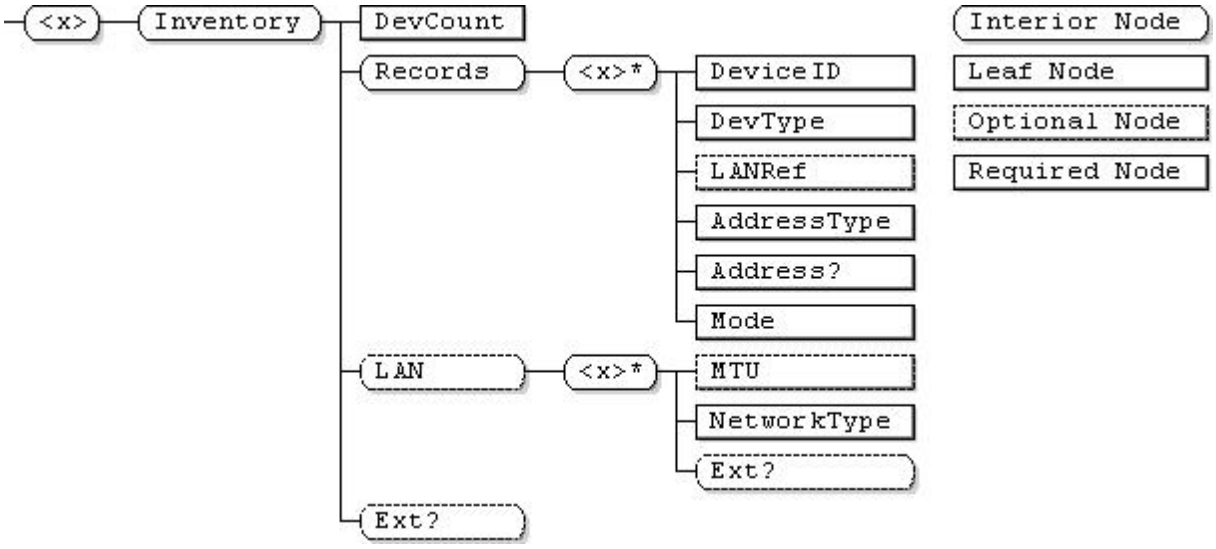


Figure 8-4: Device Inventory MO

8.5 Device Addressing

Device addressing enables a M2M Server in a public addressing domain to send messages to a M2M end device in a private addressing domain, behind a gateway. As oppose to the first device addressing approach, as described in section 5.2, which operates at the transport level, the addressing scheme used in this DM enabler is handled at the DM protocol level. That is, the identity of the recipient end device is specified in the payload of the DM message using the “TargetRef” element type.
Upon receiving DM messages originated by M2M Server over the LTE-M link, the M2M Gateway first relies on the Device Inventory MO (section 8.4) to perform the address translation. Therefore based on the specified recipient device identifier (in the DM message payload), the Gateway can look up the private address within the capillary network and the network type of the recipient end device (refer to Figure 8-4). The gateway then applies the protocol adaptation process as described in section 8.3: DM commands are translated onto parameters and functions to be invoked on the NDM interface exposed by the recipient device as described in section 8.2. This invocation is realized towards the device address over the native air interface as defined by the network type (e.g. Zigbee, Bluetooth…).
A device unique identifier is used to uniquely identify an M2M device. Device Inventory MO and EXALTED Lightweight DM protocol are handling identifier as a string of characters; thus they are agnostic to the type of the provided unique identifier being used. For the reasons described in section 5. , at the present time, this unique identifier is most likely defined by M2M Service provider. Future naming and addressing standard can be supported.

8.5.1 Full Exposure

As described in section 8.4, M2M Gateway holds a complete inventory list of end devices in the Device Inventory MO. M2M Server has access to this Device Inventory MO over the DM protocol. Thus, this full exposure enables the M2M Server to manage individually each end devices.
This high granularity exposure is required by some scenarios. For instance, in eHeath scenarios, the medical staff or M2M Server may want to have direct control on end devices (e.g. blood pressure, spirometer…).
The full device exposure approach consumes memory resources on the M2M Gateway in order to keep track of device inventory information. The required storage capacity grows as the number of end devices increases. The traffic in capillary network is also getting busier.

Based on Figure 8-1: DM consideration in capillary network, the corresponding device inventory MO with full list of end devices is depicted in Figure 8-5: Full device inventory.

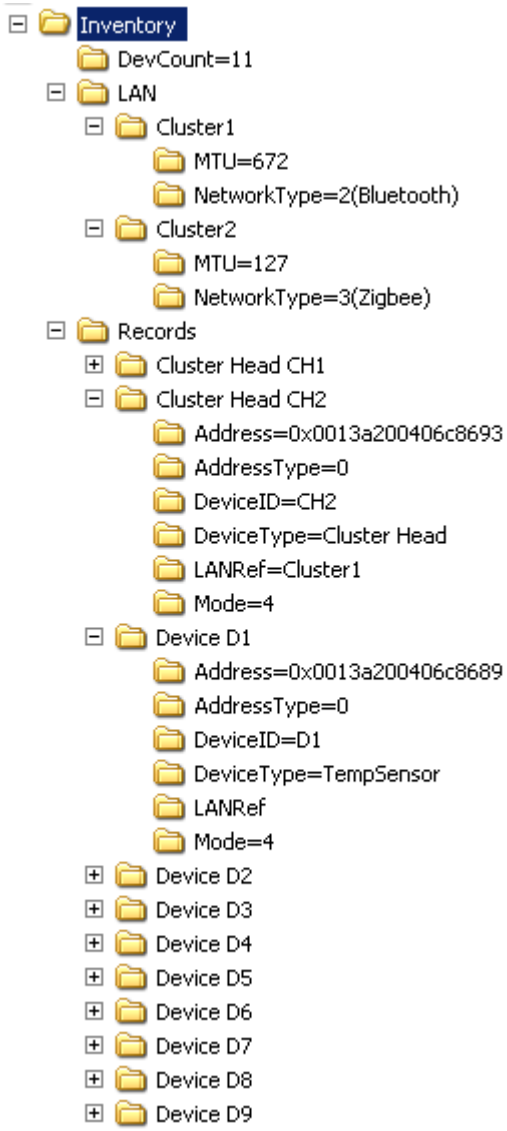


Figure 8-5: Full device inventory

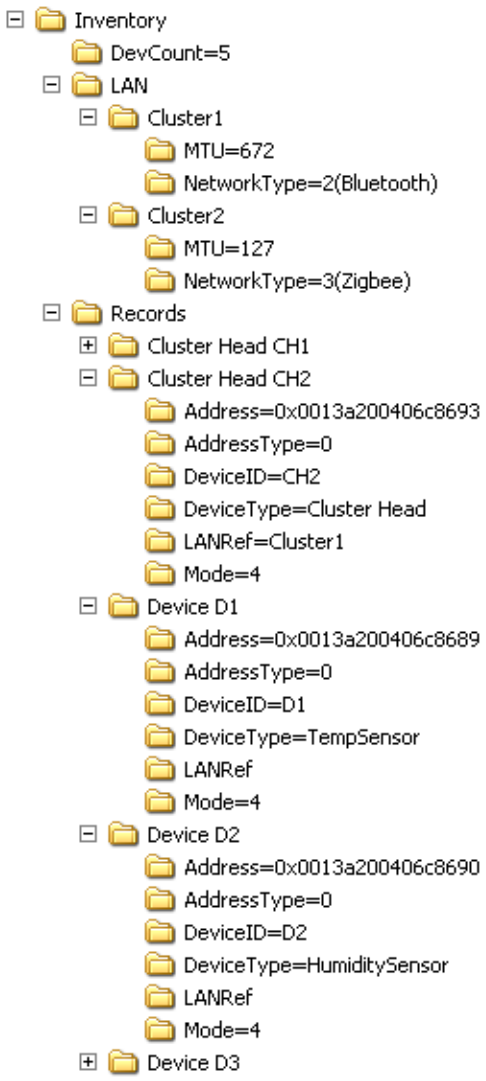


Figure 8-6: Minimum device inventory

8.5.2 Minimum Exposure

Other scenarios, such as remote monitoring, full exposure of devices to M2M Server is not required. For instance, remote monitoring of air pollution which relies on sensors spread over a large area. In this scenario, M2M Server needs to collect sensor data that has been aggregated by M2M Gateway, but does not need to have direct control on each single sensor. Gateway and CHs still have the responsibility to ensure usual DM functions such as configuration, software update.

For scenarios requiring minimal device exposure to M2M Server, a lightweight approach can be adopted which consists in reusing the GwMO's Device Inventory MO as described in section 8.4. And this MO only holds a complete list of CHs in the capillary network. End devices that are directly managed by the gateway are also captured in this MO. End devices managed by CHs are not listed in the Device Inventory MO. DM related traffic in the capillary network can be significantly reduced, as most "device attach" and "device detach" events

originated by end devices are no longer forwarded and propagated up to the M2M Gateway. Therefore the required energy power for the transmission of said DM related traffic can also be saved.
Based on Figure 8-1: DM consideration in capillary network, the corresponding device inventory MO with minimal end devices exposure to M2M Server is depicted in
Figure 8-6: Minimum device inventory. Note that compared to the full device exposure approach Figure 8-5, devices D4, D5, D6, D7, D8, D9 are not listed in the MO as they are managed by cluster heads CH1 and CH2. The DevCount is now 5, versus 11 in the full exposure approach.

8.6 Multicast and Broadcast

This section describes multicasting and broadcasting performed at the DM protocol level. These features contributes to enhance the system scalability as it enables M2M Server to send DM commands once, over the LTE-M link, to M2M Gateway. The gateway then forwards the received DM commands to multiple end devices.
It should be noted that DM sessions are initiated by devices before M2M Server can issue DM commands.

8.6.1 Multicasting to group of end devices

OMA Gateway Management Object enabler [39] has defined various MOs for realizing DM functions in M2M Gateway. Gateway Config MO resides in the management tree of the M2M Gateway and it maintains information regarding the handling of different types of end devices by the M2M Gateway. Figure 8-7 gives the pictorial description of the Gateway Config MO. It contains the following sub-trees:

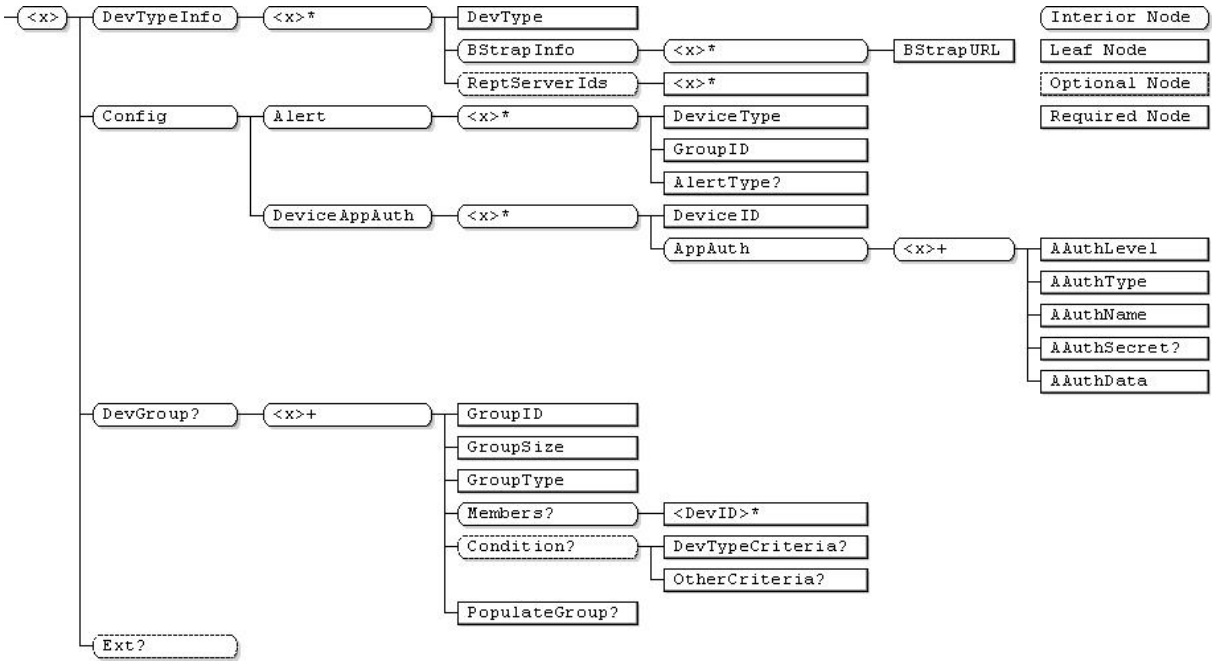


Figure 8-7: Gateway Config MO

- DevTypeInfo: This sub-tree is used to hold control information for reporting the Device Attach/Detach alerts and providing bootstrapping information depending on the device type of the End Devices. The device inventory process described earlier makes use information in this sub-tree.
- Config: This sub-tree is used by the M2M Server to configure the M2M Gateway for features such as the reporting of alerts that the M2M Gateway might send to the M2M

Server. It is also used to store End Device credentials on the DM Gateway for those End Devices that require DM Server assisted bootstrapping [39].

- DevGroup: This sub-tree is used to assign attached End Devices into groups, which can then be addressed for command fan-outs or notification fan-outs.

8.6.1.1 Defining device group

M2M Server uses the group identifier to fan-out commands to multiple end devices, over the EXALTED Lightweight DM protocol.
DevGroup sub-tree can handle multiple device groups, each group has a placeholder node within this sub-tree, e.g. “<x>/DevGroup/group1”, “<x>/DevGroup/group2”, etc.
Each device group has the following attributes, represented as leaf nodes or interior nodes in the sub-tree:

- GroupID, the unique identifier of the device group. “<x>/DevGroup/group1/GroupID”
- GroupSize, specifies the number of devices within the group.
- GroupType specifies the membership rule. If the M2M Server sets this value to be ‘0’, then the M2M Server must add individual end Devices under “<x>/DevGroup/<x>/Members” sub-tree. If the M2M Server sets this value to be ‘1’ or ‘2’, then the M2M Gateway must add individual end Devices under “<x>/DevGroup/<x>/Members” sub-tree according to the conditions specified in “<x>/DevGroup/<x>/Condition” node. GroupType of “1” is used to group all devices having the same specified DeviceType together. Other grouping option can be defined with GroupeType set to be “2”. Once the M2M Server has configured the device grouping conditions (GroupType set to “1” or “2”, rule specified in Condition node), it can trigger the M2M Gateway to start populating the “<x>/DevGroup/<x>/Members” sub-tree. This action is initiated by executing the “<x>/DevGroup/<x>/PopulateGroup” node.
- All end devices belonging to this group are listed in “<x>/DevGroup/<x>/Members” sub-tree, through unique device identifiers

8.6.1.2 Fanout

Once the device grouping is setup, M2M Server uses Fanout MO to multicast DM commands to group of devices. The Fanout MO resides in the management tree of the M2M Gateway and maintains information regarding the handling of DM command fan-out and response aggregation. Figure 8-8 gives the pictorial description of the Fanout MO.

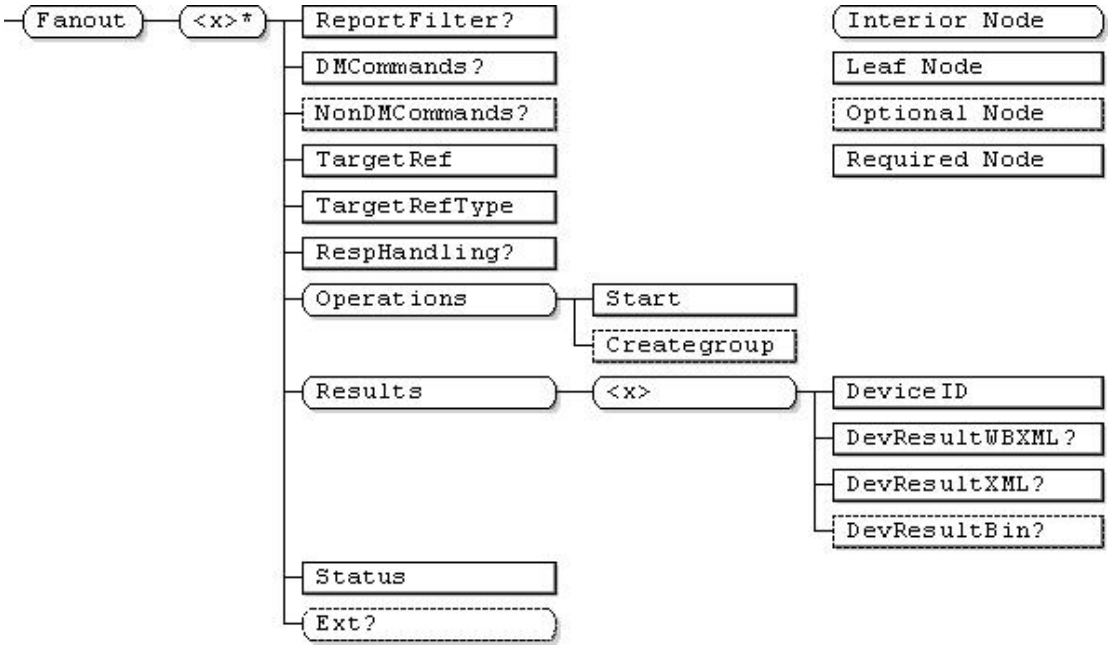


Figure 8-8: Fanout MO

This sub-tree can hold multiple fanout operations.
To multicast DM commands to group of devices, M2M Server proceeds as follow:

- Configure a Fanout operation over the DM protocol by manipulating nodes of Fanout MO:
  - Define the command to be executed in “Fanout/<x>/DMCommands” or in “Fanout/<x>/NonDMCommands” node. The latter node serves to specify non-OMA DM commands.
  - Indicate whether the previously specified commands are intended to a device or a group of devices. “Fanout/<x>/TargetRefType” node is set to “0” to indicate that “Fanout/<x>/TargetRef” points to a unique device group identifier, and the value “1” indicates the “Fanout/<x>/TargetRef” node is pointing to unique device identifier.
  - Set the device group identifier or device identifier onto “Fanout/<x>/TargetRef” node
  - Configure the response handling. Set the “Fanout/<x>/RespHandling” node to be “0” for no report. Set this value to “1” to have the M2M Gateway to report the overall status of the command upon completion of the fan-out. M2M Server can then retrieve the complete results from the Results sub-tree. M2M Server set this value to be “2” in order to have the M2M Gateway to report the overall status of the command along with the aggregated results.
- Start the fanout process
  - M2M Server executes the “Operations/Start” node causing the M2M Gateway to send the commands (specified in “Fanout/<x>/DMCommands” or in “Fanout/<x>/NonDMCommands” node) to target end devices (members of the specified group)
- Receive results from Gateway
  - Depending on the value of “Fanout/<x>/RespHandling”, M2M Server may be notified about the completion of the fan-out operation, see above.
  - M2M server can retrieve the complete results in the Results sub-tree, or
  - M2M Server can optionally specify the filter condition for aggregated results, in the “Fanout/<x>/ReportFilter” node

8.6.2 Broadcast to group of devices

The multicast approach as described in the previous section 8.6.1 is a standardized OMA-DM method using OMA GwMO enabler. This multicasting scheme is necessary for scenarios that require full exposure of end devices to M2M Server, e.g. eHealth scenarios as described in 8.5.1. However, this approach is not optimized for scenarios that comprise a large number of devices using one gateway. For instance, let's consider a remote control scenario which consists in turning on/off 100,000 street lights of a city. In the previous approach, the Gateway would have to create 100,000 sub-nodes in the Device Inventory MO, and in the Config MO's DevGroup sub-tree. Not only this consumes large amount of memory in the Gateway, the fanout operation can be lengthy. Furthermore, grouping of devices per area basis requires important M2M Server configuration activities over the LTE-M link.
To address the above street light scenario, it's better to consider a multi-hierarchy level topology built up with distributed clusters using CHs. Management of 100,000 light bulbs is delegated to a chain of CHs. A minimal device exposure model can therefore be leveraged, refer to 8.5.2. In this approach the number of end devices entries in Gateway's Device Inventory MO and Config MO is significantly reduced. The overhead on the gateway is thus significantly reduced and the usage of LTE-M bandwidth for configuring groups of devices is also minimized. Broadcasting DM commands to an entire branch of end devices can be achieved by sending DM commands to the top level node (CH) of the branch. Upon receiving the DM commands, the top level cluster head of the branch applies the recursive command forwarding approach as described in section 8.2 a broadcast scheme can thus be realized. The branch represents an area of the city, sub-branches represent segments of streets and so forth. This breakdown makes use of CHs which handle a manageable amount of light bulbs.

Provided a distributed capillary network topology as depicted in Figure 8-9: Broadcasting commands to end devices, to broadcast DM commands to all end devices in a tree hierarchy, M2M Server proceeds as follow:

a) The gateway only exposes top level CHs (A, B & C) in the Device Inventory MO. These CHs are managed directly by the Gateway.
b) M2M Server may configure a device group G to include CHs B and C, as described in section 8.6.1.1
c) Apply the fanout operation as described in section 8.6.1.2 steps a) to c)
    - It should be noted that with this approach, M2M Server can broadcast DM commands to all end devices within a branch by applying the fanout operation to cluster head A, the top level node of the branch. Cluster head A then relays commands to lower level CHs, as described in section 8.2, which in turn relay the commands recursively down to end devices. In this case, defining group is not necessary to target branch A, thus further resource saving can be achieved.
    - M2M Server can broadcast DM commands to all end devices within branches B and C. To achieve this, it configures a fanout operation to target the group G (created in step b). The gateway fanout the commands to members (cluster heads B and C) of group G. CHs B and C handle the same process as described in the previous point for CH A.

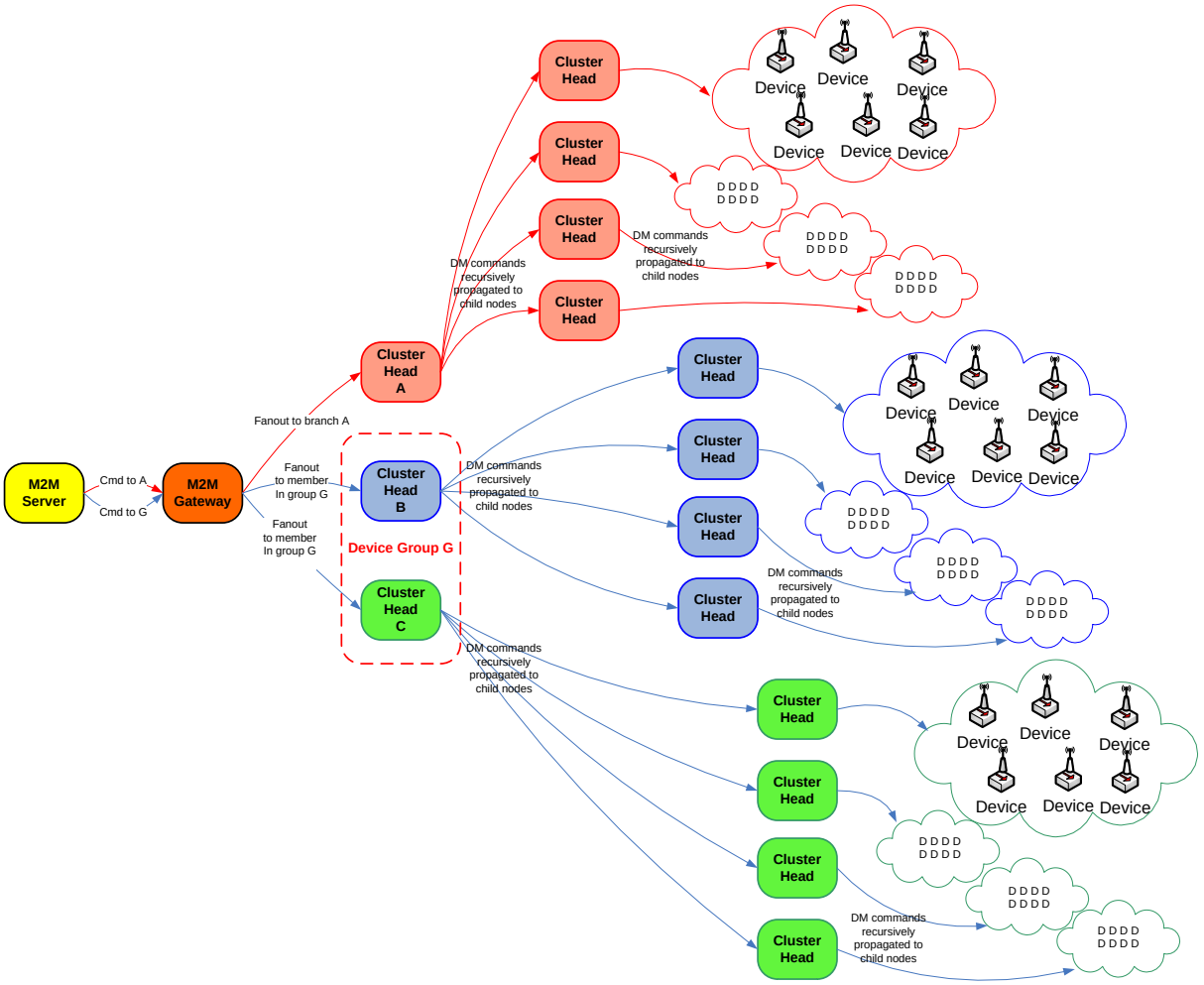


Figure 8-9: Broadcasting commands to end devices

8.6.3 Multicast End-to-end Workflow

Broadcasting in a multi-level hierarchy capillary network topology is described in the previous section. Multicasting can also be applied to this type of network topology.

For instance, let's consider a firmware update scenario using multicast, the high level workflow would be as follow:

1. M2M Applications, management authority, specifies a firmware update operation:
    a. Connect to M2M Server to retrieve a list of Device Groups
    b. M2M server returns a list of Groups as persisted in the Device Inventory MO, back to M2M Application.
    c. Management authority view (further information may be displayed e.g. number of devices in the selected Group, location, device types, current software/firmware version, etc)
    d. Admin selects the targeted Groups, the total number of devices is displayed.
    e. Admin selects the version of firmware to upgrade to.
    f. The initial FOTA deadline is calculated based on the number of LTE-M device and M2M Gateway involved in the FOTA operation. Management authority may adjust this proposed deadline.
2. M2M Server acknowledges the FOTA operation defined by management authority in step 1 and proceed to start the FOTA campaign (handled by automation):
    a. URLs of selected firmware packages are specified in FUMO
    b. SMS may be issued to LTE-M devices and M2M Gateway involved in this operation, to initiate DM sessions

   c. Upon connection of LTE-M devices and M2M Gateway to DM Server, FUMO server pushes FOTA commands (Replace and Exec on FUMO) to update the FUMO tree of gateway. Acknowledgments are collected and logged.
3. M2M Gateways multicast FOTA commands to targeted device Groups
   a. M2M Gateways forward FOTA commands to the first hierarchy level of CHs and end devices which are members of the targeted Group, specified in the DM payload issued in step 2.
   b. Upon receiving the FOTA commands, CHs, in turn, forward them to devices in lower hierarchy level. This recursive downstream propagation is repeated by CHs to reach devices in the lowest hierarchy level.
   c. As CHs may have different air interfaces (e.g. Zigbee, Bluetooth), FOTA commands can be fan-out to heterogeneous capillary networks.
   d. It should be noted that firmware packages are also pushed down to CHs and made available to adjacent end devices, to avoid multiple download of firmware packages via multi-hops.
   e. Upon execution of FOTA, status is collected recursively and aggregated at the M2M Gateway and ultimately sent to M2M Server
4. M2M server's FOTA campaign manager monitors the overall progress of pending FOTA operations
   a. Campaign manager collects and analyses the aggregated FOTA status posted by M2M Gateway and LTE-M devices
   b. Retry policy is applied to unsuccessful upgrades operation, until full completion or abortion in the case the number of retries is exhausted.

8.7 Mobility Management

Config sub-tree in Gateway Config MO (Figure 8-7) can be configured to report device attach/detach notifications to M2M Server. M2M Application can leverage this option along with information residing in Device Inventory MO to support mobility management at the application level.

Some eHealth scenarios need mobility management at the application level. For instance, let's consider a remote patient monitoring scenario: the application needs to monitor an elder patient's blood pressure and oxygen saturation of hemoglobin. The oximeter and the sphygmomanometer electronic appliances are used by the elder patient for the measurement. The readings are collected by a gateway through Bluetooth connectivity. This smart assistive gateway can provide the following services to elderly or disabled patient: voice/visual reminder to make the required measurements, voice/visual instruction to guide the patient on installing/wearing the appliance, and to trigger the worn appliance to start the reading. Smart oximeter and sphygmomanometer both have a presence sensing capability, the smart gateway can therefore be informed which appliance is being worn by the patient, so the gateway can remotely trigger the reading. Collected data is then sent to M2M Server by the smart M2M gateway over the LTE-M connectivity. The M2M medical application makes regular connection to the M2M Server in order to retrieve the collect data (stored in the retention component). The medical staffs use M2M medical application to access the patient's personal health data and to deploy "monitoring prescriptions" to gateways. This personalized monitoring prescription contains a list of reading to be made along with the reading frequency, the type of reading and associated appliance to be used. The smart gateway uses this electronic prescription to create reminder, to guide the patient through the reading process and to automatically trigger the reading on devices.

This medical application must take the mobility of patient into account. For instance, the patient is on holiday and is visiting relatives. The patient only travels with oximeter and sphygmomanometer, the smart gateway is left at home. These devices should connect to M2M Server by the way of another smart gateway, located at patient relative's house. As

such, M2M application must be automatically aware of this mobility in order to deploy the "monitoring prescription" to the new gateway. The continuation of the aforementioned assistive services can thus be maintained.

The required mobility feature is enabled at the application layer by leveraging components described earlier:

- Device inventory, as described in section 8.4
- Full exposure of devices to M2M Server. Notify M2M Server whenever a device is: (i) attached to gateway (ii) detached from gateway. This function is described in section 8.5.1.
- M2M Server relays the device attach/detach notifications to M2M Application.
    - Upon device attaching, monitoring prescription can thus be pushed to the newly attached gateway
    - Device attach/detach events fired by an electronic wristlet can be used to determine whether the patient is at home or not

8.8 Device & Secure Element

8.8.1 Association

To prevent any misuse of the Secure Element it must be strongly associated to the device. Without this association an attacker could recycle a valid Secure Element and solder it on another device with adequate probes to introspect the system. This association is also called "pairing" and must not be confused with the process of establishing a shared key between devices without relying on pre-shared key or PKI.

The pairing between the device and its secure element must be done offline in a secure environment. A unique device identifier is sent and store into the Secure Element and the couple (Secure Element Identifier, unique device identifier) is stored in a database to be exploited later by the M2M server to enable any required control during the life of the device.

During this setting the very early group key is also set in the secure element. This setting is called pairing in the parlance of the project but it will need these keys to be replaced when the device is deployed on the field.

8.8.2 Key setting

Taking into account the fact that the Factory key is prepared before the deployment of the device, the group key setting algorithm is protected against the MITM attack. As introduced in 4.3.3.2 the principle of the protocol is first to set a shared session key and then to send the group key to the device protected by this session key. In the following Crypto-MAC refers to Message Authentication Code. A Crypto-MAC is similar to a signature in a way that a message is hashed and then this hash is encrypted with a key. With the protocol below a symmetric key is used to encrypt the hash and with this regard is different form a digital signature where is must be the private key to be used to sign the hash.

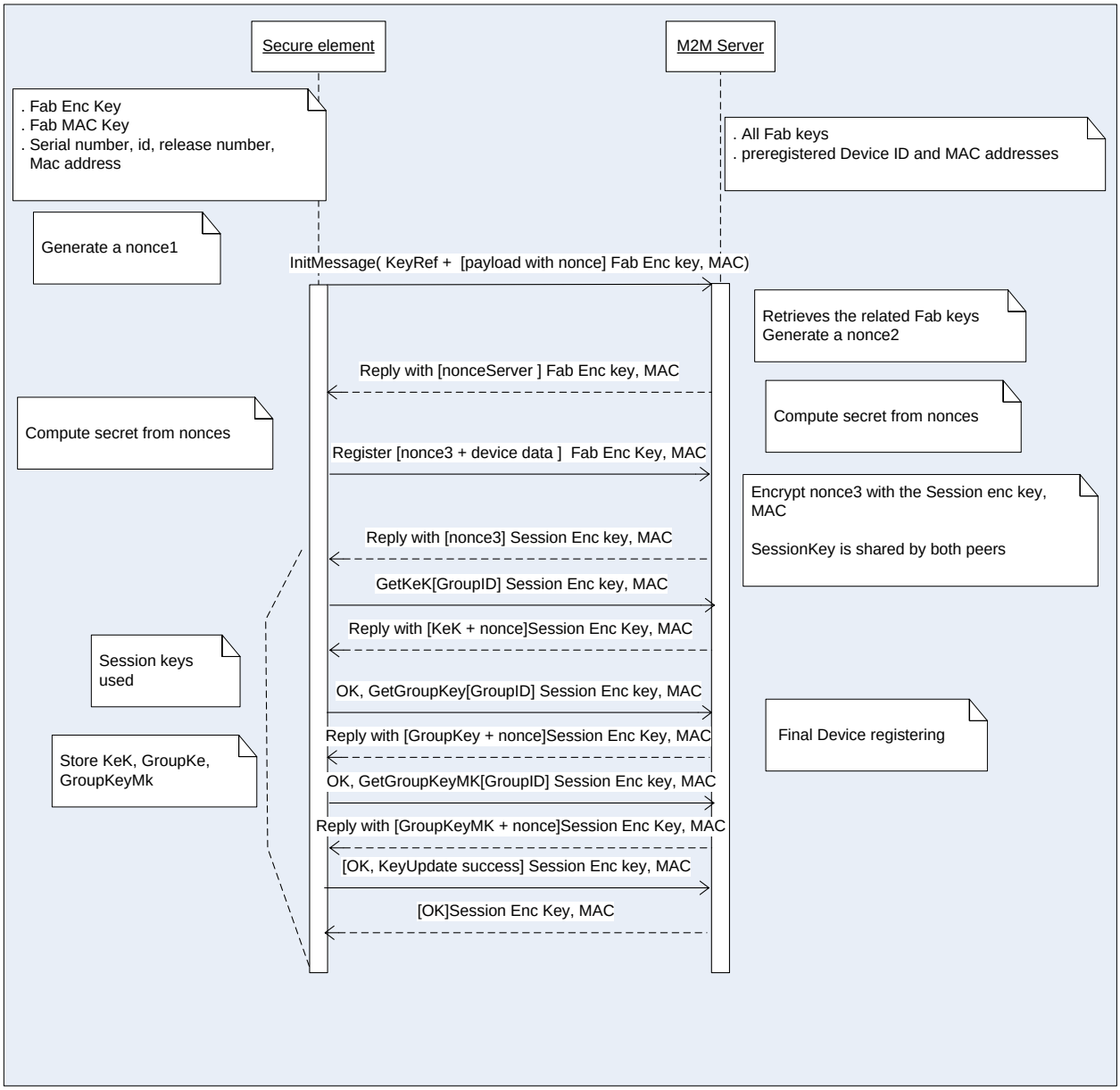


Figure 8-10: Group key setting in the Secure Element

First a nonce is sent to the M2M server protected with the Fab Keys. The device knows the server address either because it directly received this address with an SMS or because the gateway booted the handshake and provided the IP address of the M2M server.

The server answers to this InitMessage by providing its nonce. These two nonces enable to obtain both Session Encryption and Crypto-MAC key. The device sends a Register message with another nonce - called nonce3 in the diagram- and all devices data like the device ID, maybe the release number of the software, the physical address. Reply to this register message is the nonce3 protected with the session keys generated by both peers. Both the secure element and the M2M server share the same secret.
Next message is the GetKeK. KeK is the Key encryption Key that is given in the answer by the server. Both the message and the answer are protected with the session keys. The KeK is in the Secure Element. Next the M2M server will transmit to the SE the Group Key Management Key (Group Key MK) and Group Key. All messages exchanged from now on can be Crypto-MACed using this group key.

The Group Key MK will serve to all management operation of the SE and the Group Key is for encrypt and sign data provided by the device.

8.8.3 Group Key Master Key

As described in the previous section a Group Key MK is also send by the M2M server to the Secure Element to be stored for a future usage. This Group Key MK is used to encrypt a new Group key value that is sent to the Secure Element. This key replacement process is initiated by the M2M server like any Device Management action as described in the previous section: a Device Management notification action is sent to the device together with an URL address on a final acknowledge message at the end of an application message exchange. This process is depicted in the figure below.

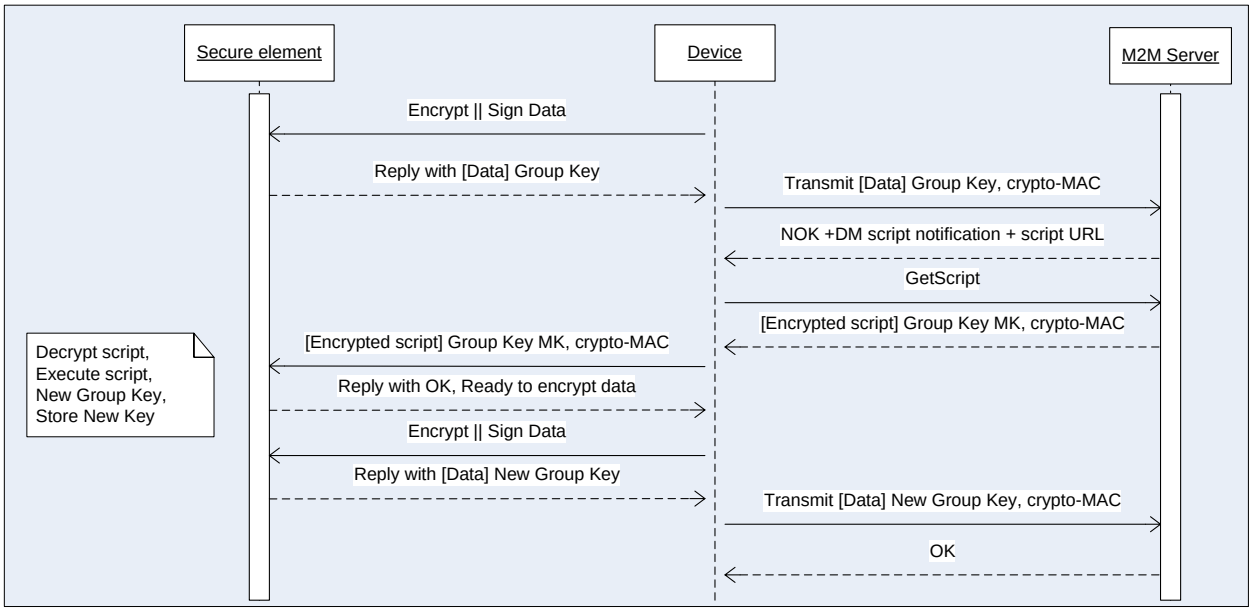


Figure 8-11: Key replacement handshake.

The Device Management Agent of the device is able to download the Device Management script using the URL passed. The received script is encrypted with Group Key MK and then, it is send to the Secure Element. The SE decrypts the script, interprets it and runs it. As a result the new Group Key is stored. Same process might be applied to the Group Key MK to replace it if required, except that the key itself has an extra encryption and needs to be deciphered using the KeK.

9. Conclusion

Recapitulating DM solutions disclosed in this report, key achievements are summarized below:

DM solutions disclosed in this report, comprise 4 mains items:

1. EXALTED Lightweight DM protocols
2. DM functions and DM related service capabilities
3. Architecture that enables extension of additional services in the future
4. Management of devices in capillary network, including forming and structuring capillary network
5. End-to-end security

50 technical requirements have been defined and listed in deliverable D2.1 [1]. DM solutions presented in this report can address 27 of them.

Algorithms developed by other EXALTED tasks can be mapped onto the DM architecture. Values created by various algorithms can be summed up to further achieve high level objectives.

ELFOMA extends OMA-DM v1.x specifications so that existing OMA-DM v1.x servers can be reused as is to continue supporting existing mobile devices and to incrementally support new constrained M2M devices. Operators and service platform providers may be interested by this cost saving option.
ELFOMA is spectrum efficient as the size of DM messages is reduced by 85% compared to OMA-DM v1.x messages
OMA-DM GwMO is reused and adapted to manage Non-LTE-M devices deployed in multi-level hierarchical heterogeneous capillary network.

EXALTED Lightweight DM protocol is used to support both management functions and service capabilities. This merge contributes to lower the cost of devices due to simplifications.

ELFOMA method could be applied to other Lightweight related M2M standardization activities. The tree-based to flat data model mapping principle could help to leverage existing OMA MOs in DM Next Generation (section 4.1.2.2) Work Items. ELFOMA approach combined with envisaged protocol simplifications in DM Next Generation (DM NG) will further reduced OMA-DM 2.x payloads. JSON-based messages are more verbose than ELFOMA. Although OMA-DM v1.x protocol is stateful, namely, connection oriented, ELFOMA concept can be applied to connectionless oriented protocols, such as CoAP which is considered by DM NG.
An implementation option onto ETSI M2M architecture is disclosed in the annex. It shows how ELFOMA can be integrated onto OpenMTC platform. This platform is a prototype that implements ETSI M2M architecture.

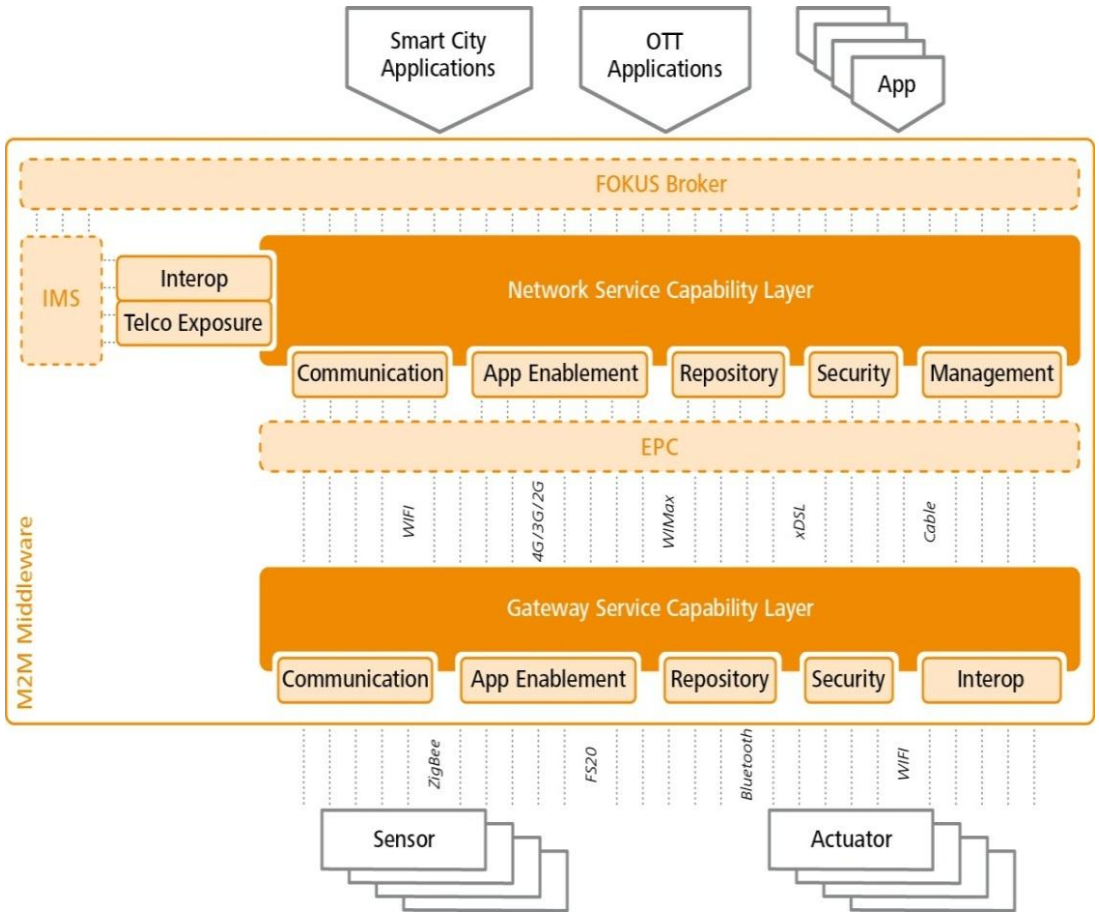
ELFOMA is being implemented in Device Management Testbed [10]. This testbed will also be used to numerically assess Key Performance Indicators (KPIs) pertaining to signaling reduction, i.e. Transmission payload size (K30), Actual payload size (K32). A preliminary list of KPIs can be found in Deliverable D7.2 [10]. KPIs measured in tesbed will be compared to analytical assessment unveiled in section 4.2.1.5. Analysis on potential deviations will be disclosed in the upcoming deliverable D7.3.

As a summary, it is claimed that DM solutions presented in this report provide a significant contribution towards the achievement of the following EXALTED objectives: (i) Scalability, (ii) Energy efficiency, (iii) Cost efficiency, (iv) Reliability, (v) Security

Annex. Implementation Option Onto ETSI M2M M2M Architecture Using The Fraunhofer OpenMTC Prototype

Fraunhofer is Europe's largest application-oriented research organization with 60 Fraunhofer Institutes at different locations in Germany. The Fraunhofer Institute for Open Communication Systems (FOKUS) in Berlin develops – among other fields – a generic M2M platform that is meant to act as a facilitator between different service platforms and the core network to enable the seamless communication management of different terminals, sensors and actuators. The goal is a flexible support of urban spaces in different areas, for example building surveillance and control, electronic health care systems, smart metering or localization services. In 2012, Fraunhofer FOKUS published the first release of the OpenMTC platform [78].

OpenMTC is a middleware platform enabling M2M communication. It has been designed to act as a horizontal convergence layer for machine type communication that supports multiple vertical application domains. Such domains are the typical M2M market segments (verticals) such as transport and logistics, utilities, automotive, eHealth, etc. which can be deployed as part of a common platform.
OpenMTC consists of two service capability layers, a gateway service capability layer (GSCL) and a network service capability layer (NSCL). Many design aspects and features of those have been specified by the ETSI Technical Committee M2M in [75] and [76]. Additionally, a device service capability layer (DSCL) is available for selected devices, designed as extension of the GSCL. Figure A.1 depicts the OpenMTC architecture. Optionally, OpenMTC can be combined with our OpenEPC, OpenIMS, and FOKUS Broker platforms enabling extensive integration with other operator core network functionality.



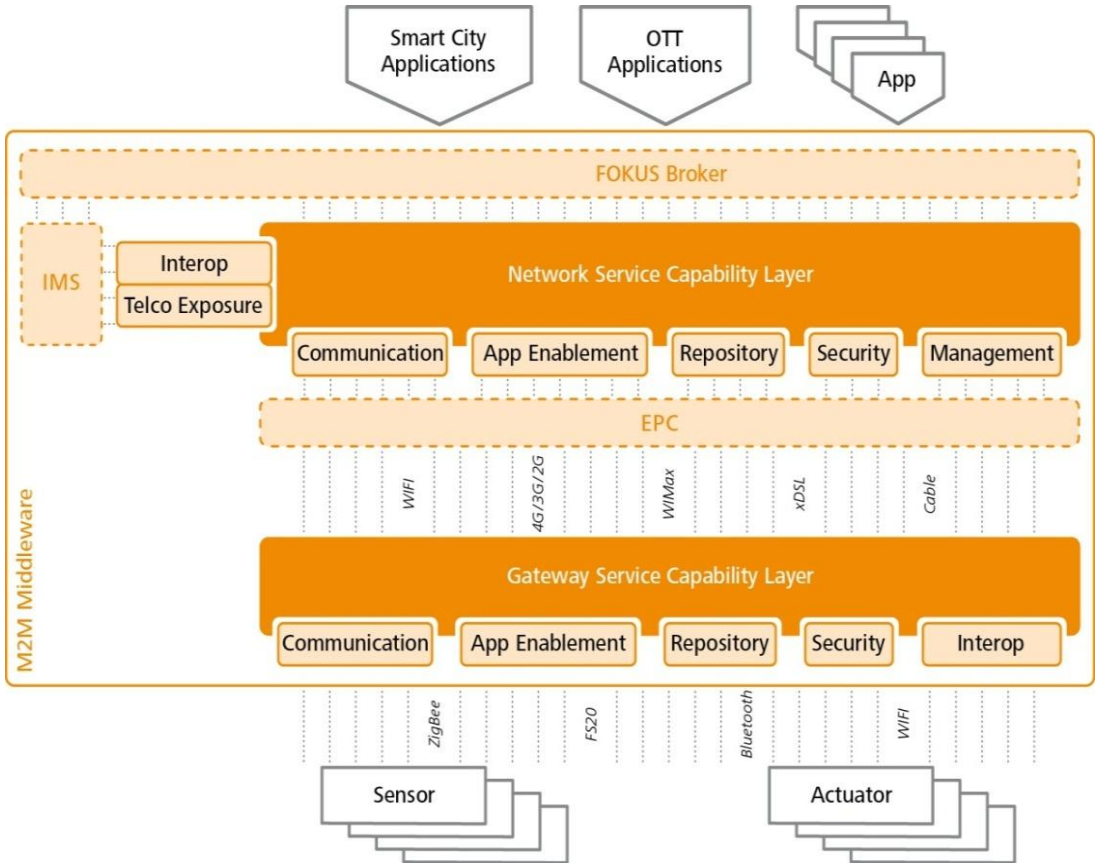


Figure A.1: High level OpenMTC architecture

A. OpenMTC Features

The first release focuses on the fundamental communication features of an M2M middleware enabling fast hands-on experimentation. OpenMTC currently supports the following features on both service capability layers (G/NSCL):

- Generic Communication
- Application Enablement
- Reachability, Addressing and Repository
- Interworking Proxies
- Security (as part of Generic Communication)
- Remote Entity Management

Generic Communication (G/NGC) includes:

- mId reference point implementation
- Support for synchronous, semi-synchronous and asynchronous communication between the network service platform and the M2M gateways and devices
- Transport session establishment and teardown based on HTTP: establishment of communication data paths between the devices and the network
- Secure delivery (HTTPS) of application data from/to M2M gateways or M2M devices and the NSCL. The communication is independent of the HTTP/S transport which may be replaced upon demand with other protocols such as CoAP.

Application Enablement (G/NAE) includes:

- mIa reference point implementation: network applications are able to discover, register and access the OpenMTC network service platform over the mIa interface. Through this reference point, the network enables applications to communicate efficiently and in a secure manner with a large number of devices while remaining transparent to the actual M2M communication mechanisms.
- dIa reference point implementation: enables applications on the gateways and on the devices to register to the corresponding OpenMTC middleware and through it to communicate with the network.
- Registration of applications in the respective SCL: in order to use the capabilities offered by OpenMTC, the various sensors and actuators on the device side are registered through the M2M middleware to the network side enabling the remote control and communication according to the specific service requirements.
- Routing mechanism between applications and capabilities in the SCL: through this functionality, the service requirements and the specific data exchanged between the applications and the OpenMTC are forwarded to the appropriate local handling functionality in both the devices and in the network platform.
- Routing towards different capabilities, retargeting, 1-hop, 2-hop: additional to the local routing mechanism between applications and capabilities, this functionality enables to forward or delegate control requests and information data to other functions both locally or remote.

Reachability, Addressing and Repository (G/NRAR)

- Subscriptions and notifications management: OpenMTC features an extended subscriptions management system in which the applications as well as the remote M2M middleware functionality are notified when specific data is created, modified or deleted. Through this feature, the applications are able to receive information on specific sensors and actuators events which enable the triggering of various services callbacks providing the means for managing large groups of devices.
- Through the subscription management, the OpenMTC network platform is able to receive events notifications from the devices and gateways, enabling it to manage and control devices which pertain to the same service provider or from the same technology family transparently to the actual applications. Through this means, the M2M middleware service provider is able to further optimize its network usage through dynamic responding to specific events.
- Storage of application (G/NA) and G/NSCL related registration information – in order to reach the appropriate remote device, network platform or application, this information has to be available at the remote parties. Through this functionality, the network is aware which devices are connected at a specific moment in time and thus is able to forward the data traffic to the appropriate entity.
- Storage of application (D/G/NA) and SCL data – in order to enable a trustworthy and consistent communication, the application communication parameters and data is stored in the devices and in the network. This enables efficient communication scenarios where information is cached for a specific time for aggregation and processing.

Interworking Proxy (IP)

- Gateway Interworking Proxy (GIP): Interworking between non-ETSI compliant devices or gateways and the NSCL. In order to be able to interact with off-the-shelf sensors and actuators and to offer the OpenMTC functionality also for devices which are not compliant to the ETSI standard, OpenMTC features a Gateway Interworking Proxy (GIP) translating control and data requests to the ETSI model.
- Network Interworking Proxy (NIP): Interworking with other platforms e.g. IP Multimedia Subsystem (IMS). The NIP is able to translate messages from other platforms to the

ETSI standard for example between the Session Initiation Protocol (SIP) that is widely used in IMS and the HTTP-based generic communication specified by ETSI TC M2M.

Network Remote Entity Management (NREM)
In OpenMTC rel. 1, the focus of the remote entity management is on the network side. Through this functionality the network is able to transmit communication parameters to the devices in the form of Managed Objects (MO). The specific MOs are transmitted to the mobile devices themselves which modify their parameters and through this adapt the communication. The following section explains in more detail how ELFOMA could be used in upcoming releases of OpenMTC to optimize its device management capabilities.

B. OpenMTC ELFOMA integration and implementation options

We identified 4 different options for extending the OpenMTC prototype with the Device Management solutions outlined in section 4.2.1 and 4.2.2:

1. The ELFOMA device management procedures are completely transparent to OpenMTC
2. Implementation of the ELFOMA solution as part of an alternative G/NREM service capability
3. Extend the current transport with a CoAP solution (can also be part of option 1 and/or 2)
4. Combined implementation of option 1 and 2 by supporting OMA-DM based mobile devices as well as EFOMA based M2M devices.

Option 1: ELFOMA device management procedures are transparent to OpenMTC
The ETSI TC M2M specifications leverage on OMA-DM mechanisms by defining a mgmtObj resource instance as specified by the ETSI M2M functional architecture (TS 102.690). Such MO maintains information regarding the management of the M2M service capabilities in M2M devices and gateways. [77] describes the OMA-DM compatible management objects.
Since ELFOMA supports a proxy mechanism where existing OMA-DM v1.x servers can be used as is to support both the former OMA-DM v1.x enabled mobile devices and the new constrained M2M devices, the ELFOMA mechanisms can be realized transparently by an OMA-DM adapter proxy. Such proxy is introduced between constrained M2M devices and the OMA-DM server converting messages in both directions.

Since the API that is exposed by M2M servers (i.e. the NSCL) to M2M application servers is outside the scope of EXALTED, we describe here the mIa/dIa/mId reference points specified by ETSI as those interfaces can be used by DM applications (or other SCLs) to communicate with the DM system independent of the four different options discussed here. This description is seen as a useful add-on to this deliverable that primarily targets solutions for the EXALTED core problems.

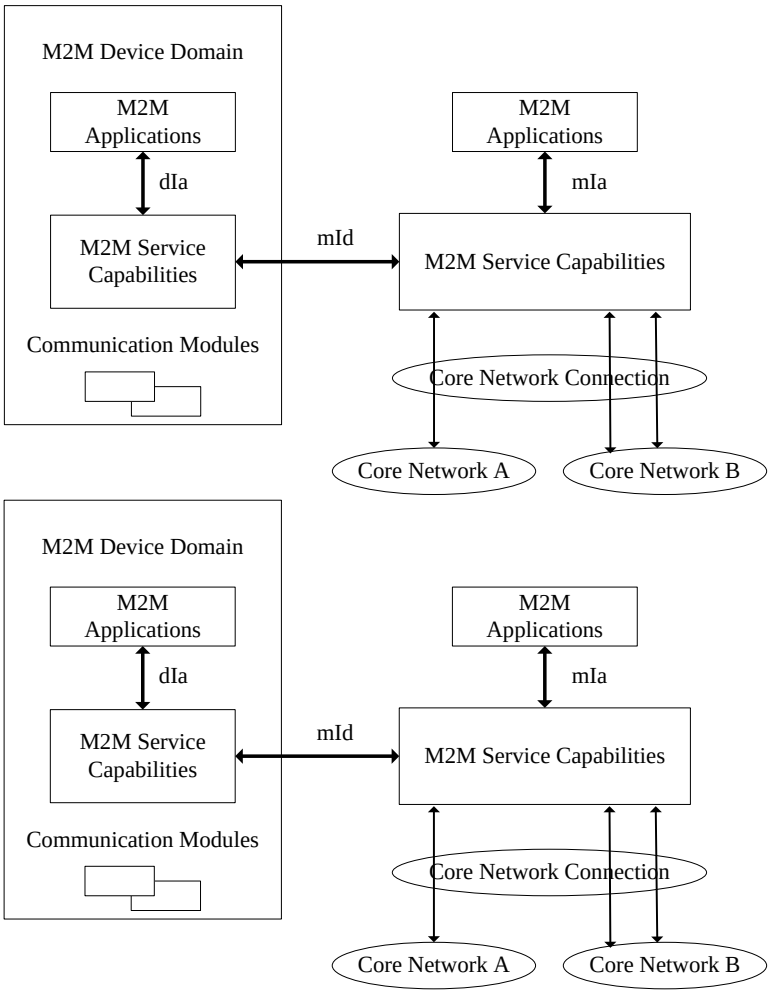


Figure A.2: The ETSI defined reference points [76]

The mIa interface allows an application to access the M2M Network Service Capabilities via defined primitives.
mId allows an M2M Device or M2M Gateway to communicate with the M2M Service Capabilities in the Network. mId uses core network connectivity functions as an underlying layer.
dIa allows an application residing in an M2M Device to access the different M2M Service Capabilities in the same M2M Device or in an M2M Gateway; and allows an application residing in an M2M Gateway to access the different M2M Service Capabilities in the same M2M Gateway.
In the following, we describe the SCL resource is described, as an example for what can be manipulated via the reference points shown in Figure A.2.

The <scl> resource shall represent a remote SCL that is authorized to interact with the hosting SCL. The D/GSCL has to register to the hosting SCL (NSCL), to get authorized to interact with it. The result of a successful registration process is the creation of two <scl> resources, one in the Issuer-SCL and one in the Hosting-SCL storing the information that each SCL needs to know about the other one.

Path	<sclbase>/scls/<sclid>
`<?xml version="1.0" encoding="UTF-8"?>`	

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://uri.etsi.org/m2m" targetNamespace="http://uri.etsi.org/m2m"
elementFormDefault="qualified" xmlns:Q1="http://uri.etsi.org/m2m">
    <include schemaLocation="attachedDevices.xsd"></include>
    <include schemaLocation="m2mPocs.xsd"></include>
    <include schemaLocation="ComplexTypes.xsd"/>
  <include schemaLocation="notificationChannels.xsd"/>
  <include schemaLocation="mgmtObjs.xsd"/>
  <include schemaLocation="subscriptions.xsd"/>
  <include schemaLocation="accessRights.xsd"/>
  <include schemaLocation="applications.xsd"/>
  <include schemaLocation="groups.xsd"/>
  <include schemaLocation="containers.xsd"/>
  <complexType name="scl">
    <sequence>
      <element ref="tns:containers" minOccurs="1" maxOccurs="1" />
      <element ref="tns:groups" minOccurs="1" maxOccurs="1" />
      <element ref="tns:applications" minOccurs="1" maxOccurs="1" />
      <element ref="tns:accessRights" minOccurs="1" maxOccurs="1" />
      <element ref="tns:subscriptions" minOccurs="1"
        maxOccurs="1" />
      <element ref="tns:mgmtObjs" minOccurs="0" maxOccurs="1" />
      <element ref="tns:notificationChannels" minOccurs="1"
        maxOccurs="1" />
      <element ref="tns:integrityValResults" maxOccurs="1"
        minOccurs="0">
      </element>
      <element ref="Q1:m2mPocs" maxOccurs="1" minOccurs="0"></element>
      <element ref="tns:attachedDevices" maxOccurs="1"
        minOccurs="0">
      </element>
    </sequence>
    <attribute name="pocs" type="tns:anyURIList" />
    <attribute name="remTriggerAddr" type=" anyURIList " />
    <attribute name="onlineStatus" type="tns:onlineStatus" />
    <attribute name="serverCapability" type="boolean" />
    <attribute name="link" type="anyURI" />
    <attribute name="schedule" type="string" />
    <attribute name="expirationTime" type="dateTime" />
    <attribute name="accessRightID" type="anyURI" />
    <attribute name="searchStrings" type="tns:searchStrings" />
    <attribute name="creationTime" type="dateTime" />
    <attribute name="lastModifiedTime" type="dateTime" />
    <attribute name="locTargetDevice" type="string" />
    <attribute name="mgmtProtocolType" type="tns:mgmtProtocolType" />
    <attribute name="id" type="string"></attribute>
  </complexType>
  <element name="scl" type="tns:containers"/>
</schema>
```

Resource description

Attribute	Type	Description
containers	sub-resource	
groups	sub-resource	
applications	sub-resource	
accessRights	sub-resource	
subscriptions	sub-resource	

mgmtObjs	sub-resource	Only applicable and mandatory in the NSCL.
notificationChannels	sub-resource	
m2mPocs	sub-resource	
attachedDevices	sub-resource	
pocs	anyURIList	List of points of contact that can be used for out-of-band communication with an SCL.
remTriggerAddr	anyURIList	triggering address of the remote entity
onlineStatus	onlineStatus	Indicates the reachability of the SCL
serverCapability	boolean	Set to TRUE if this SCL is able to issue connections towards the registered SCL.
link	anyURI	the URI of the <sclBase> resource of the registered SCL
accessRightsID	anyURI (RW)	URI of an accessRight resource which defines permissions to be applied.
creationTime	dateTime (RO)	Time of creation of the resource.
lastModifiedTime	dateTime (RO)	Last modification time of a resource.

Procedures

Name	Description
Retrieve	Get the full representation of the resource, or specific attributes. Method: sclRetrieveRequestIndication Response (successful case): 200 Ok Response (unsuccessful case): Error code
Update	Update some or all attributes of the resource. Method: sclUpdateRequestIndication Response (successful case): 200 Ok Response (unsuccessful case): Error code
Delete	Delete the resource. Method: sclDeleteRequestIndication Response (successful case): 200 Ok Response (unsuccessful case): Error code
Subscribe	subscribe to the changes in the resource representation Method: SubscriptionCreateRequestIndication Response (successful case): 201 CREATED Response (unsuccessful case): Error code

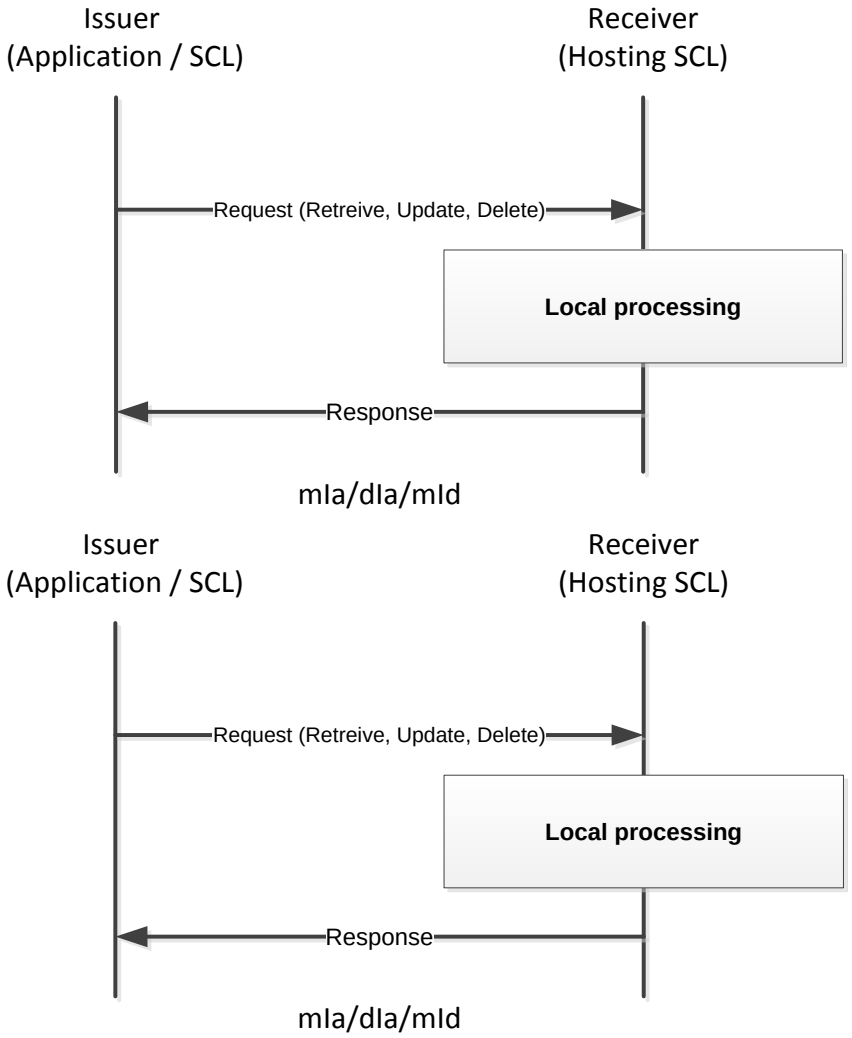


Figure A.3: Procedures for <scl> resource

One of the sub-resources is the mgmObjs resource which is the parent resource for <mgmObj> and <mgmtCmd> resources. It is created whenever the parent resource is created. A <mgmObj> resource holds the management data which represents a certain type of M2M remote entity management function, while <mgmtCmd> resource shall be only used to model non-RESTful management commands.

Option 2: Implementation of ELFOMA as part of a G/NREM service capability
Figure A.4 shows the remote management architecture. This is mapped according to the following table to the OMA and BBF management specifications.

ETSI M2M	OMA DM	BBF TR069
REM Server	DM Server	ACS
REM Client	DM Client	CPE
REM Interface (mId)	DM-1, DM-2	TR069-CWMP

Alternatively to the remote entity management (REM) Server being integrated as a part of the NREM, the REM Server may be external to the NREM but interface with the NREM via an implementation-specific interface exposed by the REM Server.

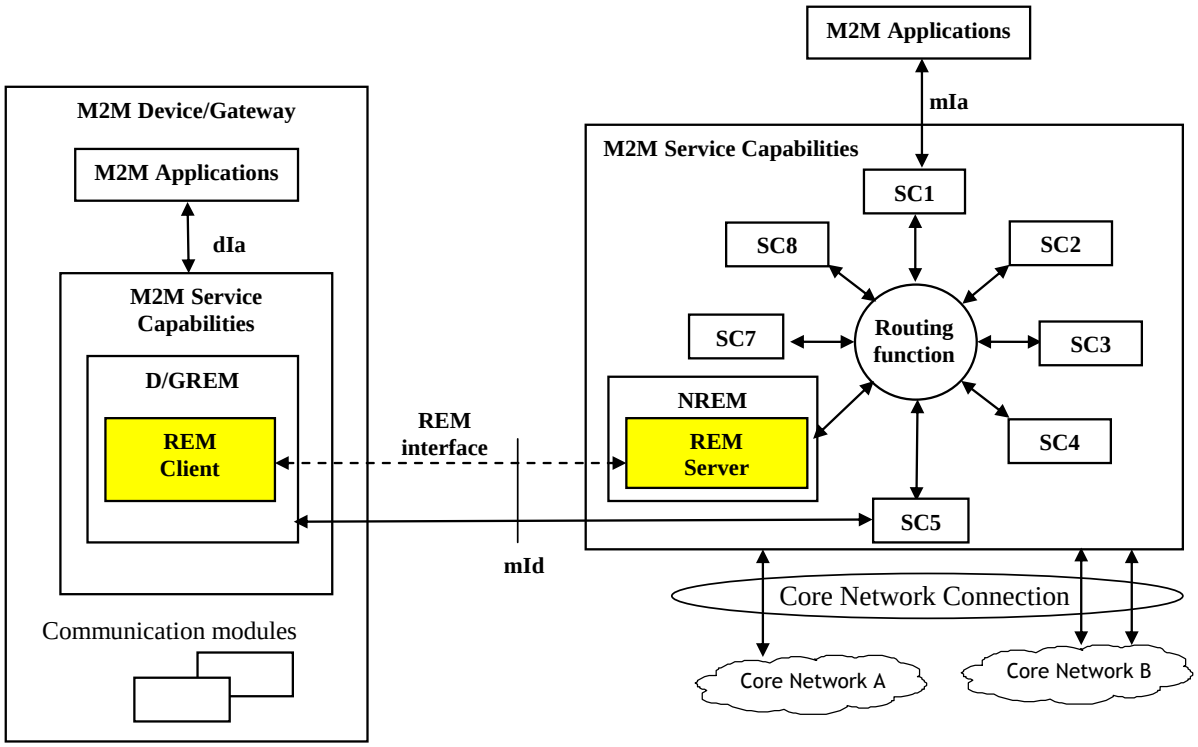


Figure A.4: ETSI Remote Management Architecture

Since the ELFOMA approach mainly touches transport and document compression, the MO (OMA DM in this case) would not have to be adapted for implementing the option presented here. This means that for the implementation of a new ELFOMA G/NREM as part of OpenMTC, the ETSI defined mgmtObj and mapping can be kept while the transport (HTTP/CoAP/headers) as well as the compression of content (XML documents) would need adaptation. Such adaptation could be implemented in the REM service capability (SC) and the GC (generic communication) SC.
In the following, we give some insights into the REM architecture, resources, general procedures, and capabilities exposed over mIa, are given. Most of this example has been extracted from the ETSI TC M2M tutorial draft. Figures Figure A.5 and Figure A.6 show the Device Management Resources.

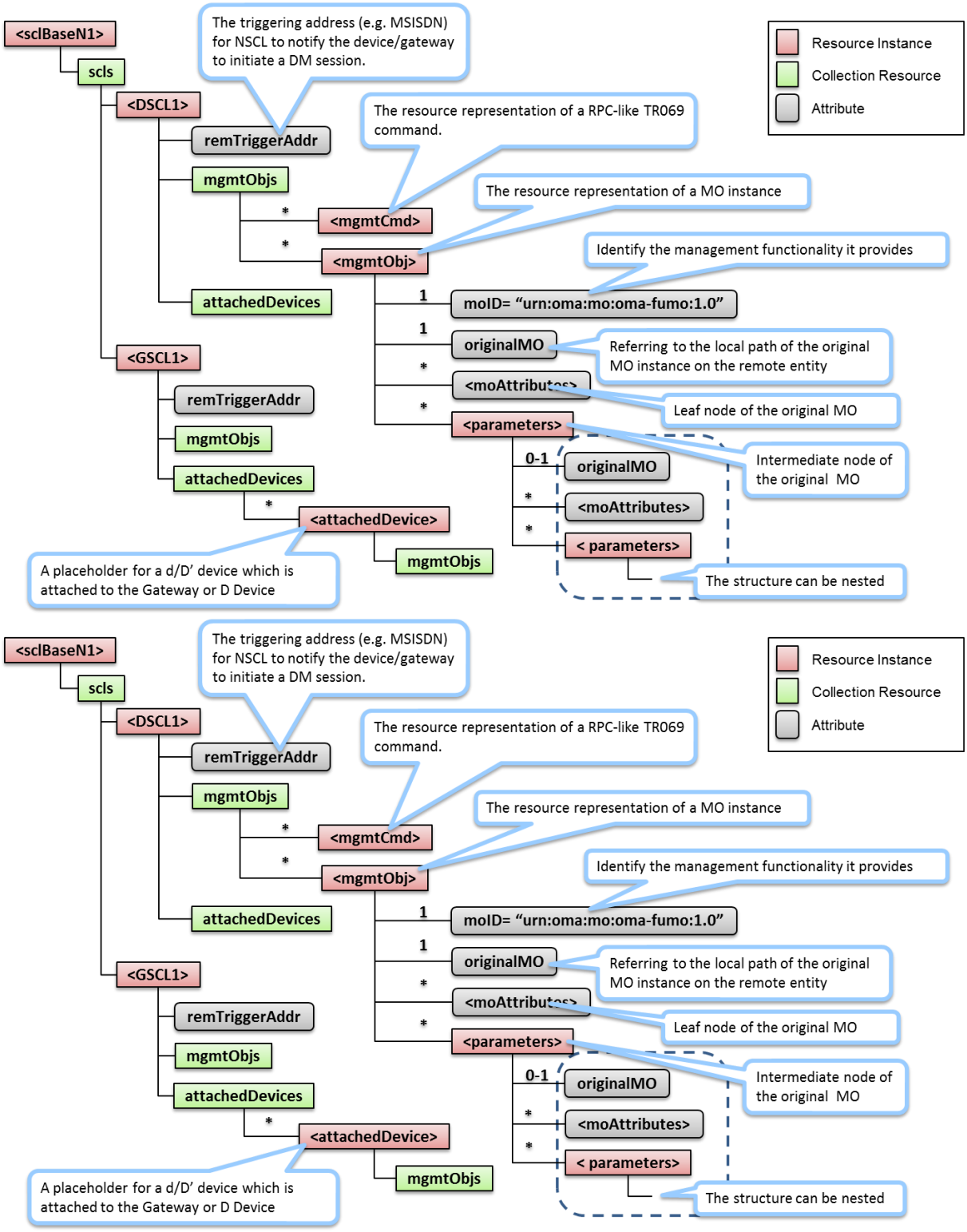


Figure A.5: ETSI Device Management Resources

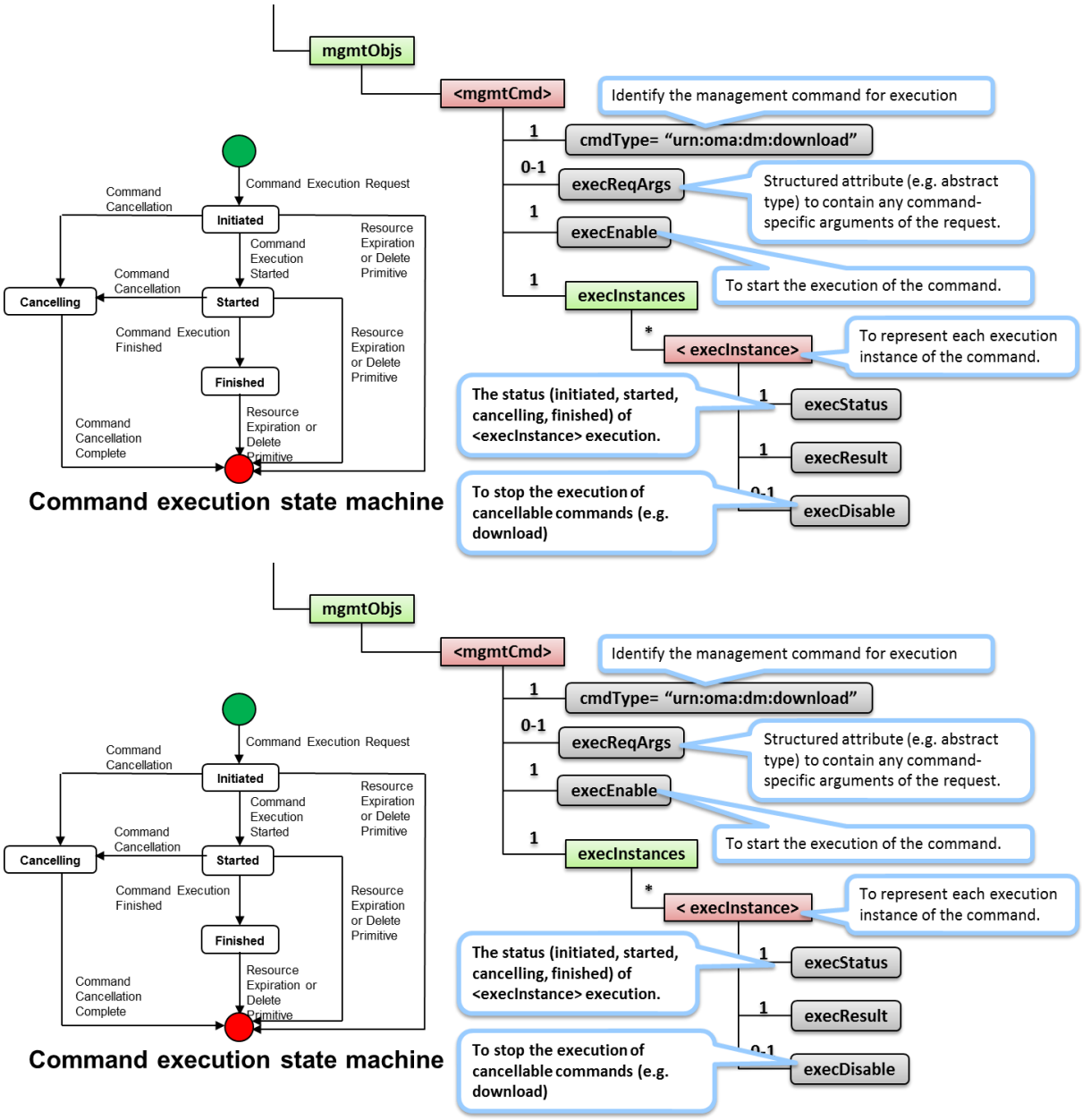


Figure A.6: ETSI Device Management Resources

ETSI M2M provides the following management functions via the RESTful APIs (mIa) by data model and primitive mapping:

- General Management
- Configuration Management
- Diagnostic & Monitoring Management
- Software/Firmware Management
- Gateway Management
- SCL Management

For each remote entity (D/G), one or multiple <mgmtObj> resources may be created in NSCL for remote management purpose. The NSCL (NREM) shall translate the RESTful methods performed on <mgmtObj> resources from a network application (NA) into corresponding OMA DM procedures performed on the MOs in the D/G as shown by Figure A.7, Figure A.8, and Figure A.9.

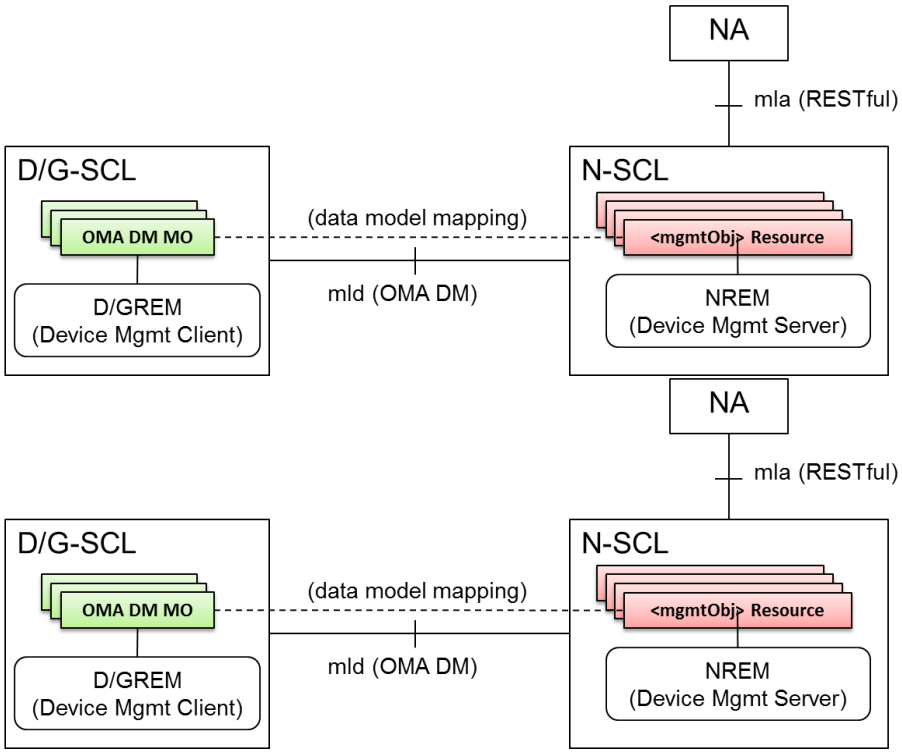
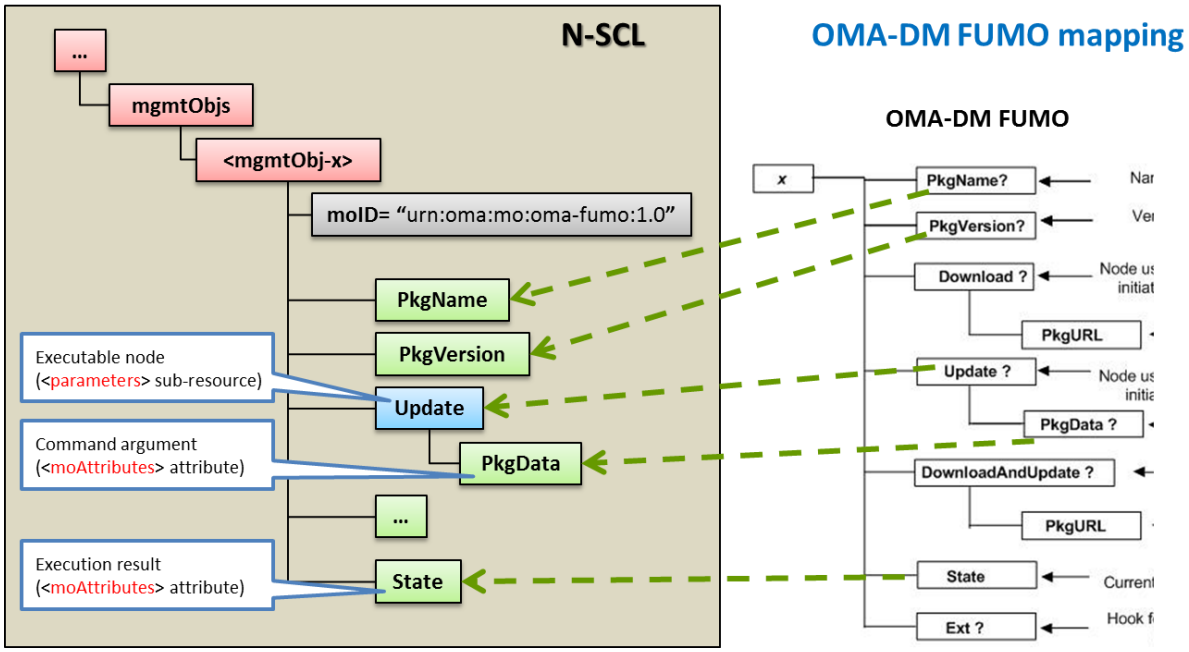


Figure A.7: REM Capabilities exposed over mIa



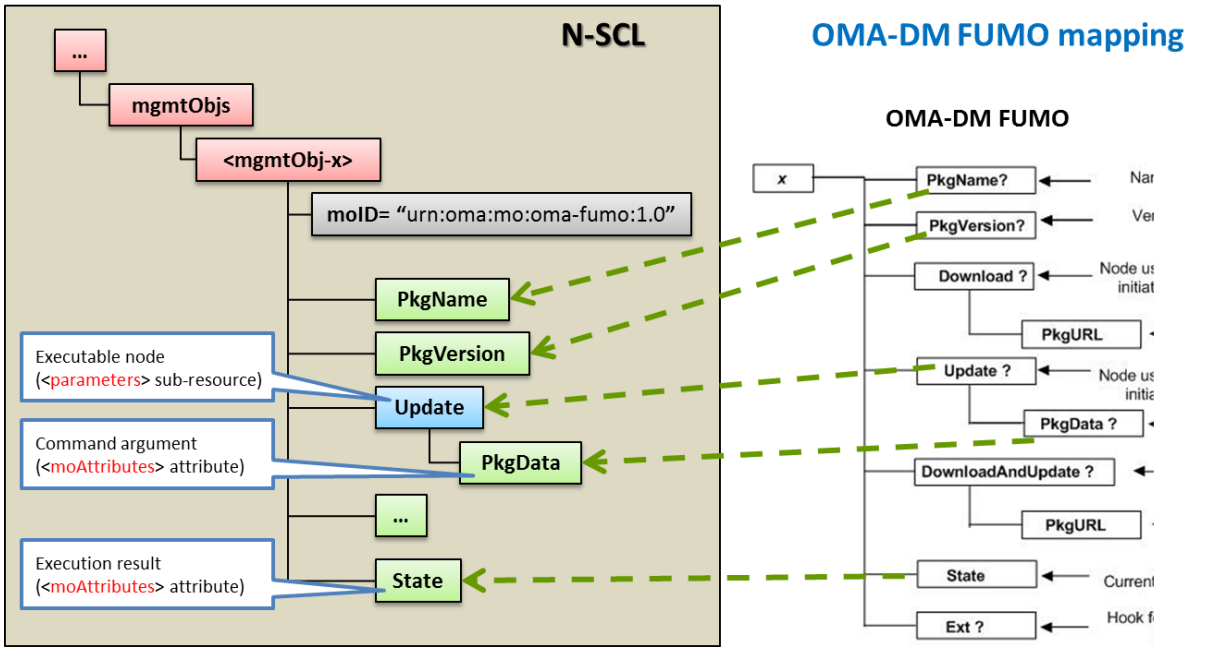
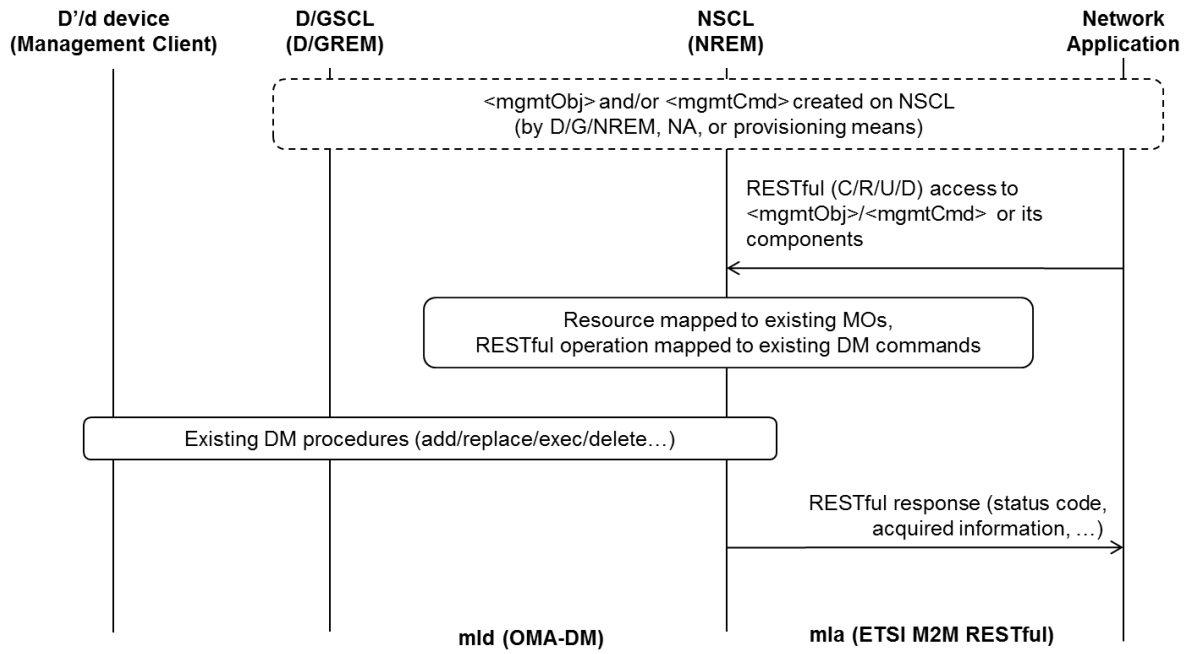


Figure A.8: MO-Resource Mapping



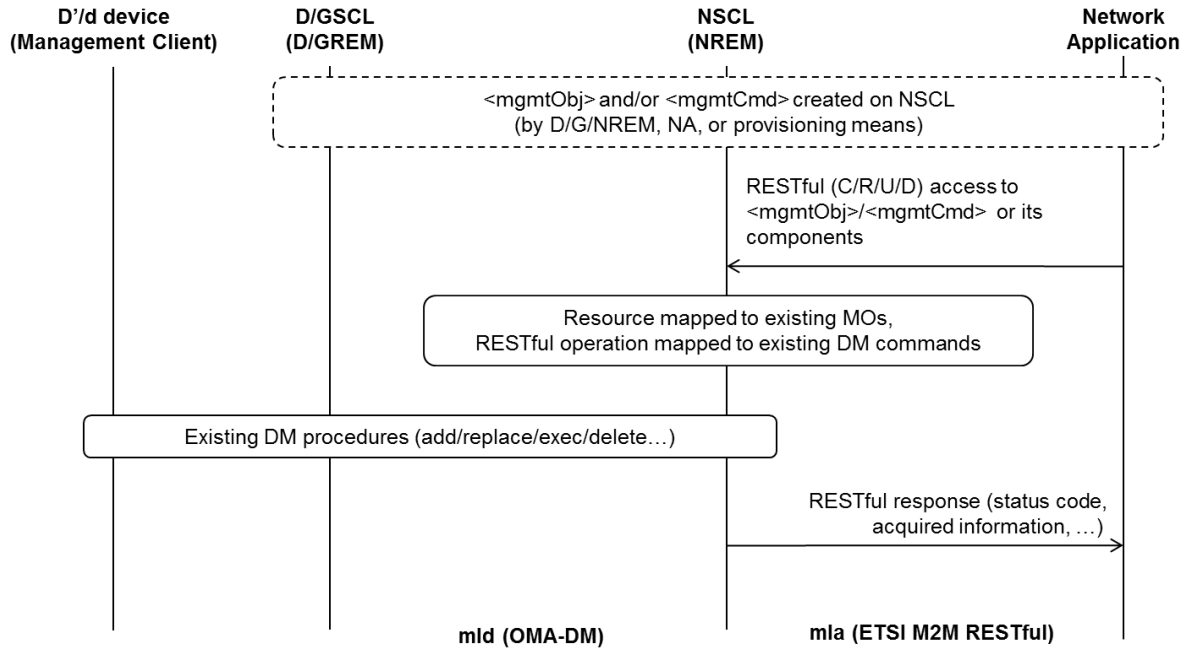


Figure A.9: ETSI REM procedures on the mIa and mId reference points

Option 3: CoAP transport extensions

Currently, OpenMTC does not implement a CoAP transport. However, the CoAP mapping is aligned with the ETSI TC M2M specifications and yields further signalling reduction. In order to implement such an approach the OpenMTC REM and GS service capabilities would need to be extended with a CoAP stack.

Option 4: Combined implementation of option 1 and 2

This option would allow supporting both OMA-DM-based mobile devices as well as ELFOMA-based M2M devices. From an implementation complexity perspective this option can be compared to option 2 since option 1 can be handled transparently. However, some additional logic would be needed as part of the REM SC in order to decide whether the standard OMA-DM or ELFOMA procedures shall be used depending on the capability of the managed device.

List of Acronyms

Acronym	Meaning
ACS	Auto Configuration Server
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
BNF	Backus-Naur Form
CCITT	International Telegraph and Telephone Consultative Committee
CH	Cluster Head
CoAP	Constrained Application Protocol
CPE	Customer Premises Equipment
CP	Control protocol
CSV	Comma Separated Values
CWMP	CPE WAN Management Protocol
DD	Device and Gateway Domain
DER	Distinguished Encoding Rules
DiagMO	Diagnostic Management Object
DM	Device Management
DTD	Document Type Definition
DTLS	Datagram Transport Layer Security
E2E	End-To-End
ELFOMA	EXALTED Lightweight DM For OMA
EPC	Evolved Packet Core
ESC	Encapsulating Security Payload
FOTA	Firmware update Over The Air
FUMO	Firmware Update Management Objects
G/NIP	Gateway/Network Interworking Proxy
G/NAE	Gateway/Network Application Enabler
G/NGC	Gateway/Network Generic Communication
G/NRAR	Gateway/Network Reachability, Addressing and Repository
G/NREM	Gateway/Network Remote Entity Management
G/NSCL	Gateway/Network Service Capabilities Layer
GwMO	Gateway Management Objects
HTTP	Hypertext Transfer Protocol
ICR	In-cluster Cluster head role Rotation
IKE	Internet Key Exchange Protocol
IMSI	International Mobile Subscriber Identity
ITS	Intelligent Transportation System
ITU	International Telecommunication Union
KeK	Key encryption Key
FIFO	First in first out
LDAP	Lightweight Directory Access Protocol
LIFO	Last in first out
LTE	Long Term Evolution
M2M	Machine-to-Machine
MAC	Message Authentication Code
MCR	Multi Cluster Reclustering
MDN	Mobile Directory Number
MITM	Man-In-The-Middle
MNO	Mobile Network Operator
MO	Management Object

MSISDN	Mobile Subscriber Integrated Services Digital Network Number
NAT	Network Address Translation
ND	Network Domain
OMA	Open Mobile Alliance
OPEX	OPErational Expenditure
PDA	Personal Digital Assistant
PER	Packed Encoding Rules
PKI	Public-Key Infrastructure
REST	Representation State Transfer
RFC	Request for Comments
RPC	Remote Procedure Call
ROI	Return On Investment
SCOMO	Software COmponent Management Objects
SMS	Short Message Service
SMS-C	Short Message Service Center
SNMP	Simple Network Management Protocol
STUN	Session Traversal Utilities for NAT
SLA	Service Level Agreement
SyncML	Synchronization Markup Language
TTL	Time To Live
TLS	Transport Layer Security
TLV	Tag Length Value
UE	User equipment
UDP	User Datagram Protocol
UDH	User Data Header
UICC	Universal Integrated Circuit Card
UMTS	Universal Mobile Telecommunication System
URI	Uniform Resource Identifier
USIM	Universal Subscriber Identity Module
HTTP	Hyper text transfer protocol
VIN	Vehicle Identification Number
WBC	White-Box Cryptography
WIMAX	Worldwide Interoperability for Microwave Access

References

[1] FP7 EXALTED: “D2.1 – Description of baseline reference systems, scenarios, technical requirements & evaluation methodology,” project report, May 2011.
[2] FP7 EXALTED; “D2.3 – EXALTED System Architecture” project report, August 2012.
[3] FP7 EXALTED: "D3.3 – Final report on LTE-M algorithms and procedures", project report, July 2012.
[4] FP7 EXALTED: “D4.1 – M2M Packet Data Protocols between LTE-M and Capillary Networks” project report, June 2012.
[5] FP7 EXALTED: “D4.2 – IP Networking System for M2M communications for EXALTED use cases” project report, June 2012.
[6] FP7 EXALTED: “D4.4 – Traffic aggregation,” project report, October 2012.
[7] FP7 EXALTED: “D4.5 – Monitoring of network sensors and actuators” project report, October 2012.
[8] FP7 EXALTED: “D5.1 – Security, Authentication & Provisionning,” project report, January 2012
[9] FP7 EXALTED: “D6.3 – Final specification of the reliable device implementation” project report, February 2012.

[10] FP7 EXALTED: “D7.2 – Integration of selected algorithms into platforms & interfaces finalization,” project report, August 2012
[11] M2M Consumer electronics forecast and research. http://wirelessnoodle.blogspot.com/2011/07/new-report-from-machina-research.html
[12] Cisco’s vision on IoT. http://share.cisco.com/internet-of-things.html
[13] CEO to shareholders: 50 billion connections 2020. http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf
[14] T. Sakurai and H. L. Vu, "MAC access delay of IEEE 802.11 DCF,” IEEE Transactions on Wireless Communications, vol. 6, no. 5, pp. 1702-1710, May 2007.
[15] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma, “Performance of reliable transport protocol over IEEE 802.11 wireless LANs: analysis and enhancement,” in Proc. IEEE INFOCOM 2002, New York, Jun. 2002.
[16] K. Schwieger, A. Kumar, and G. Fettweis, “On the impact of the physical layer on energy consumption in sensor networks,” in Proc. European Workshop on Sensor Networks, Istanbul, Jan. 2005.
[17] H. L. Vu and T. Sakurai, “Accurate delay distribution for IEEE 802.11 DCF,” IEEE Communications Letter, vol. 10, no. 4, pp. 317-319, Apr. 2006.
[18] e-SENSE, European FP6 project, Deliverable D3.3.2: Novel Cross Optimization, Available online http://www.ist-e-sense.org, last accessed on 28 Aug. 2010.
[19] IETF RFC 5166, “Metrics for the evaluation of the congestion control mechanisms,” S. Floyd, Mar. 2008.
[20] 3GPP TS 23.203 v. 8.8.0, “Policy and charging control architecture,” Dec. 2009.
[21] H. Holma and A. Toskala, “WCDMA for UMTS,” 4th edition, Willey, 2008.
[22] G. Chen, C. Li, M. Ye, and J. Wu, “An unequeal cluster-based routing protocol in wireless sensor networks,” *Springer Wireless Networks*, 2007.
[23] D. Wei, Y. Jin, S. Vural, and R. Tafazolli, “An energy-efficient clustering solution for wireless sensor networks,” *IEEE Transactions on Wireless Communications*, vol. 10, no. 11, November 2011.
[24] O. Younis and S. Fahmy, “HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks,” *IEEE Transactions on Mobile Computing*, vol. 3, no. 9, pp. 366–379, December 2004.
[25] L. Qing, Q. Zhu, and M. Wang, “Design of a distributed energy-efficient clustering algorithm for heterogeneous wireless sensor networks,” *Elsevier Computer Communications*, vol. 29, pp. 2230–2237, March 2006.
[26] M. Qin and R. Zimmermann, “VCA: An energy-efficient voting-based clustering algorithm for sensor networks,” *Journal of Universal Computer Science*, vol. 13, no. 1, pp. 87–109, January 2007.
[27] M. Ye, C. Li, G. Chen, and J. Wu, “EECS: An energy efficient clustering scheme in wireless sensor networks,” in 24th *IEEE International Performance, Computing, and Communications Conference*, 2005, pp. 535–540.
[28] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, October 2002.
[29] I. Matta, G. Smaragdakis, and A. Bestavros, “SEP: A stable election protocol for clustered heterogeneous wireless sensor networks,” in *SANPA*, August 2004.
[30] D. Kumar, T. C. Aseri, and R. B. Patel, “EEHC: Energy efficient heterogeneous clustered scheme for wireless sensor networks,” *Computer Communications*, vol. 32, no. 4, pp. 662–667, March 2009.
[31] S. D. Muruganathan, D. C. F. Ma, R. I. Bhasin, and A. O. Fapojuwo, “A centralized energy-efficient routing protocol for wireless sensor networks,” *IEEE Radio Communications*, vol. 43, no. 3, pp. 8–13, March 2005.
[32] D4.1, M2M packet data protocols between LTE-M and capillary networks, http://www.ict-exalted.eu/publications/deliverables.html, Jun. 2012

[33] Constrained Application Protocol (CoAP), CoRE Working Group, http://tools.ietf.org/pdf/draft-ietf-core-coap-10.pdf, Jun. 2012
[34] TR-069, CPE WAN Management Protocol v1.1, Issue 1, Amendment 2. http://www.broadband-forum.org/technical/download/TR-069_Amendment-2.pdf
[35] OMA: Device Management V1.2; Approved Enabler-Release Date 17 June 2008. http://www.openmobilealliance.org/Technical/release_program/dm_v1_2.aspx
[36] OMA DM Representation Protocol; Approved Version 1.2.1, 17 June 2008. http://www.openmobilealliance.org/Technical/release_program/docs/DM/V1_2_1-20080617-A/OMA-TS-DM_RepPro-V1_2_1-20080617-A.pdf
[37] OMA Device Management Next Generation: http://member.openmobilealliance.org/ftp/Public_documents/TP/Permanent_documents/OMAWID_0172-DM_NG-V1_1-20110407-A.zip
[38] OMA Lightweight M2M Work Item: http://member.openmobilealliance.org/ftp/public_documents/dm/LightweightM2M/2011/OMA OMADM- LightweightM2M-2011-0001-INP_LightweightM2M_RD_Kick_Off.zip
[39] OMA Gateway Management Object (GwMO) v1.0. http://www.openmobilealliance.org/Technical/release_program/GwMO_V1_0.aspx
[40] Fastinfo Set. http://www.itu.int/ITU-T/asn1/xml/finf.htm
[41] Efficient XML Interchange (EXI). http://www.w3.org/XML/EXI/
[42] Binary XML Content Format Specification V1.3, July 2001. http://www.openmobilealliance.org/technical/release_program/docs/browsing/v2_1-20061020-a/wap-192-wbxml-20010725-a.pdf
[43] Sherif Sakr, “XML compression techniques: A survey and comparison”, Journal of computer sciences 75 (2009) 303-322
[44] Wilfred Ng, Lam Wai Yeung, James Cheng, “Comparative analysis of XML compression technologies”
[45] Sherif Sakr, “Investigate state-of-the-art XML compression techniques”, http://www.ibm.com/developerworks/xml/library/x-datacompression/index.html?ca=drs-
[46] Christopher J Augeri, Barry E Mullins, Dursun A Bulutoglu, Rusty O Baldwin, Leemon C Baird III, “An analysis of XML compression efficiency”, http://xml.coverpages.org/AugeriExpCS-2007.pdf
[47] SyncML Representation protocol v1.1. SyncML RePro: http://www.openmobilealliance.org/tech/affiliates/LicenseAgreement.asp?DocName=/syncml/syncml_represent_v11_20020215.pdf
[48] EXIfficient. Open source implementation of the W3C Efficient XML Interchange (EXI) format specification. http://exificient.sourceforge.net/
[49] KXML2, XML parser with WBXML support. http://kxml.sourceforge.net/kxml2/
[50] W3C Working draft 7 April 2009, « Effcient XML Interchange Evaluation », http://www.w3.org/TR/exi-evaluation/
[51] 3GPP Release 11 V0.1.1, 2012.
[52] M. Becker, K. Kuladinithi, T. Poetsch, “Transport of CoAP over SMS and GPRS” http://tools.ietf.org/html/draft-becker-core-coap-sms-gprs-00
[53] C. Bormann, Z. Shelby, “Blockwise transfers in CoAP“, http://tools.ietf.org/html/draft-ietf-core-block-04
[54] E. Wilde, A. Vaha-Sipila, “URI Scheme for Global System for Mobile Communications (GSM) Short Message Service (SMS)”, 2010.
[55] K. Li, “CoAP over SMS”, http://tools.ietf.org/html/draft-li-core-coap-over-sms-00
[56] 3GPP TS 03.48 V8.9.0 3GPP TS 03.48 Security mechanisms for the SIM application toolkit
[57] Liu Pu, “An Improved Short Message Security Protocol For Home Network”, International Conference on Future Computer and Communication, FCC 2009.
[58] S2-112899 “M2M: Small data transmission using optimised SMS”; Vodafone; available online: http://www.3gpp.org/ftp/tsg_sa/WG2_Arch/TSGS2_85_XiAN/Docs/

[59] OMA Firmware Update Management Object ; FUMO V1.0.4; http://www.openmobilealliance.org/Technical/release_program/fumo_v1_0.aspx
[60] OMA Diagnostics and Monitoring; DiagMon Management Object v1.1; http://www.openmobilealliance.org/Technical/release_program/diagmon_V1_1.aspx
[61] OMA Device Management Bootstrap; V1.2.1 – 17 June 2008 ; http://www.openmobilealliance.org/Technical/release_program/docs/DM/V1_2_1-20080617-A/OMA-TS-DM_Bootstrap-V1_2_1-20080617-A.pdf
[62] OMA Device Management Notification Initiated Session; V1.2.1, 17 June 2008, http://www.openmobilealliance.org/Technical/release_program/docs/copyrightclick.aspx?pck=DM&file=V1_2_1-20080617-A/OMA-TS-DM_Notification-V1_2_1-20080617-A.pdf
[63] Group Communication for CoAP, http://tools.ietf.org/html/draft-rahman-core-groupcomm-07
[64] OMA Software Component Management Object; V1.1; http://www.openmobilealliance.org/Technical/release_program/scomo_v1_1.aspx
[65] SIMalliance Open API Workgroup, http://www.simalliance.org/en/about/workgroups/open_mobile_api_working_group/
[66] International Telecommunication Union (ITU), « Information technology- Abstract Syntax Notation One: Specification of basic notation, X680 and followers, http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf, July 2002
[67] ITU, “Introduction to ASN.1 “, http://www.itu.int/ITU-T/asn1/introduction/
[68] Darren Mundy and David W. Chadwick, “An XML alternative for performance and Security: ASN.1”, IT Professional Volume 6 Issue 1, January 2004
[69] OSS Nokalva, Inc., release 6.0.4, revision 5344, available for download from http://www.oss.com/asn1/products/asn1-download.html
[70] OMA DM Tree and Description; v1.2.1; June 2008; http://www.openmobilealliance.org/Technical/release_program/docs/copyrightclick.aspx?pck=DM&file=V1_2_1-20080617-A/OMA-TS-DM_TND-V1_2_1-20080617-A.pdf
[71] OMA Lock And Wipe Management Object; V1.0; April 2012; http://www.openmobilealliance.org/Technical/release_program/law_mo_v1_0.aspx
[72] OMA Device Capability Management Object; V1.0; April 2012; http://www.openmobilealliance.org/Technical/release_program/dcmo_v1_0.aspx
[73] ISO/IEC 7816-4, Identification circuit cards – Part 4 Organization, security and commands for interchange, second edition, Jan. 2005
[74] Brecht Wyseur, Page on White-Box cryptography theory available on http://www.whiteboxcrypto.com/research.php
[75] ETSI TS 102.690, “Machine-to-Machine communications (M2M); Functional architecture”
[76] ETSI TS 102.921, “Machine- to- Machine communications (M2M); mIa, dIa and mId interfaces”
[77] ETSI TS 103 092, “Machine-to-Machine communications (M2M); OMA DM compatible Management Objects for ETSI M2M”
[78] OpenMTC website, www.open-mtc.org