ETSI TR 102 966 V1.1.1 (2014-02)

Technical Report

**Machine-to-Machine communications (M2M);
Interworking between the M2M Architecture
and M2M Area Network technologies**

Reference
DTR/M2M-00014ed111

Keywords
interworking, M2M, service

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Machine-to-Machine communications (M2M).

1 Scope

The present document collects and evaluates implementation profiles for interworking with M2M Area Network technologies.

An implementation profile is defined, for the purpose of the present document, as the description on how the ETSI M2M architecture can be used to achieve interworking. Each implementation profile is evaluated against deployment scenarios and applicable technologies in order to identify the most suitable for the specific conditions.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

Not applicable.

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TS 102 690: "Machine-to-Machine communications (M2M); Functional architecture".

[i.2] ETSI TS 102 921: "Machine-to-Machine communications (M2M); mIa, dIa and mId interfaces".

[i.3] IEEE 802.15.4-2003: "IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)".

[i.4] CEN EN 13757: "Communication systems for meters and remote reading of meters".

[i.5] CEN EN 13757-3: "Communication systems for and remote reading of meters - Part 3: Dedicated application layer".

[i.6] ISO 8601:2004: "Data elements and interchange formats -- Information interchange -- Representation of dates and times".

[i.7] IETF RFC 1006: "ISO Transport Service on top of the TCP Version: 3".

[i.8] IETF RFC 5023: "The Atom Publishing Protocol".

[i.9] ISO/IEC 14543-3-10:2012: "Information technology -- Home Electronic Systems (HES) -- Part 3-10: Wireless Short-Packet (WSP) protocol optimized for energy harvesting -- Architecture and lower layer protocols".

[i.10] OASIS.OBIX_1_1: "OASIS oBix semantic conventions, version 1.1".

[i.11] ASHRAE.CSML_1_0: "ASHRAE 135 annex am Control System Modelling Language (CSML) semantic conventions".

[i.12] IETF RFC 4287: "The Atom Syndication Format".

3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AES	Advanced Encryption Standard
AN	Area Network
ASHRAE	American Society of Heating, Refrigeration, and Air-conditioning Engineers
ATOM	XML document format defined in RFC 4287 [i.12]
COV	Change Of Value
CSML	Control System Modelling Language
DA	Device Application
DDNS	Dynamic DNS
DHCP	Dynamic Host Configuration Protocol
DIF	Data Information Field
DIP	Device Interworking Proxy
DNS	Domain Name System
DPT	Data Point Type (KNX™ standard)
DSCL	Device Service Capability Layer
GA	Gateway Application
GENA	General Event Notification Architecture
GIP	Gateway Interworking Proxy
GSCL	Gateway SCL
HAN	Home Area Network
IN	INPut
IP	Internet Protocol
IPA	Interworking Proxy Application
IPU	Interworking Proxy Unit
ISO	International Standard Organization
KNX™	Konnex protocol maintained by the KNX™ Association
LAN	Local Area Network
MAC	Medium Access Control (Layer)
MAN	M2M Area Network
NA	Network Application
NA/DA	Network Application/Device Application
NIP	Network Interworking Proxy
NSCL	Network SCL
OASIS	Organization for the Advancement of Structured Information Standards
OUT	OUTput
PAN	Personal Area Network
PC	Personal Computer
PDA	Personal Digital Assistant
REST	REpresentational State Transfer
RF	Radio Frequency
SCL	Service Capability Layer
SLIP	Serial Line IP
TBD	To Be Defined
TC	Technical Committee
UCP	User/Universal Control Point
UDN	Unique Device Name
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
UTC	Coordinated Universal Time
VIF	Value Information Block

WAN	Wide Area Network
XML	Extensible Markup Language
ZC	ZigBee® Coordinator
ZCL	ZigBee® Cluster Library
ZED	ZigBee® End Device
ZGD	ZigBee® Gateway Device
ZR	ZigBee® Router

4 Scenarios for interworking

Interworking is one of the main caracteric of the exploitation of the usage of the ETSI M2M solution. The scenarios for interworking described in the present document make use of the ETSI TC M2M architecture as shown below. The interworking makes use of GIP, NIP and DIP capabilities which are seen by the SCLs as specialized applications dedicated to the semantic data model interworking.

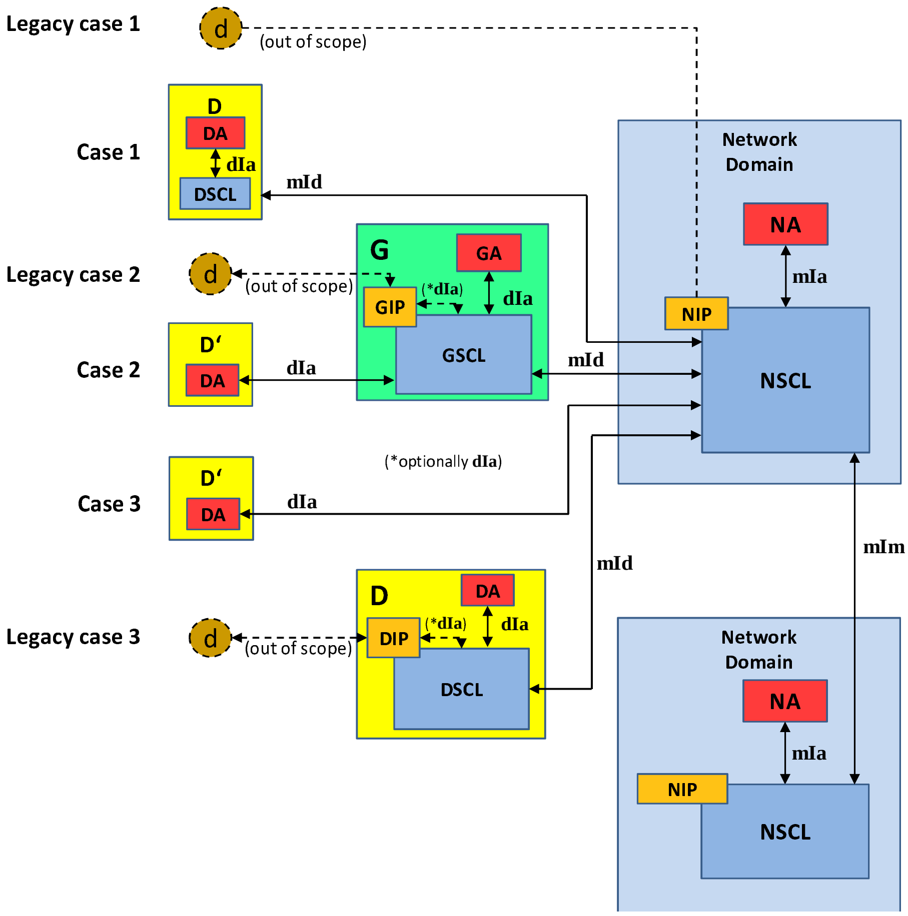

Figure 4.1: Mapping of reference points to different deployment scenarios

5 scenarios for interworking have been identified in the following table, each one applicable to different deployment context. Guidelines for data model interworking between ETSI M2M and specific area network technologies are provided in the rest of the present document.

Scenario 1: transparent retargeting		
Type of device	**d**	**Notes**
Application on device (d, non using dIa)	Specific technology aware	
Application on Network	Specific technology aware	The application is specific for the interworked technology, A specific adaptation is needed to use mIa.
Mechanism	Interworking at the G/D with simple retargeting	
Security impact	Use technology-specific security to the Gateway, independently of M2M Service layer security	Decryption(/reencryption) required at Gateway, i.e. Gateway acts as Aggregator, to extend Service Layer security end-to-end.
Leverage on M2M architecture capabilities	minimum	GIP/DIP is an application using standard dIa towards the G/DSCL.
Deployment scenario example	This interworking scenario is using ETSI M2M as a sort of pipe to carry the specific protocols, so the level of interaction with the ETSI M2M resource management capabilities (Access Rights, security, management, etc.) is limited by visibility on the objects in the interworked technology, but with the relevant advantage but that interworked protocols are preserved. One typical scenarios is the deployment of a specific technology on top of a consolidated ETSI M2M deployment, to leverage of already massively installed ETSI M2M D/G.	

Scenario 2: Retargeting with elements interworking		
Type of device	**d**	**Notes**
Application on device (d, non using dIa)	Specific technology aware	
Application on Network	Specific technology aware	The application is specific for the interworked technology, A specific adaptation is needed to use mIa.
Mechanism	Interworking at the G/D based on retargeting and use of ETSI compliant resources	GIP/DIP is an application using standard dIa towards the G/DSCL.
Security impact	Use technology-specific security to the Gateway	Decryption(/reencryption) required at Gateway, i.e. Gateway acts as Aggregator, to extend Service Layer security end-to-end.
Leverage on M2M architecture capabilities	Yes, level depends on specific solutions	
Example of applicability	This interworking scenario is similar to scenario 1 but is leveraging on the functionality offered by ETSI M2M by means of a more detailed mapping of elements (sensors. Actuators, etc) on ETSI M2M resources. It also allows other applications (e.g. native ETSI M2M application) to interact actively with the elements of the interworked technology that are stored and manipulated by the SCLs. Also in this case the interworked protocols are preserved. One typical scenario is the deployment of a specific technology that leverages on ETSI M2M for the interaction with the communication system.	

Scenario 3: Interworking at the Device/Gateway		
Type of device	**d**	**Notes**
Application on device (d, non using dIa)	Specific technology aware	
Application on Network	Independent from Specific technology	The application is ETSI M2M native and independent for the interworked technology.
Mechanism	Full Interworking at the G/D	GIP/DIP is an application using standard dIa towards the G/DSCL.
Security impact	May rely on technology-specific security over MAN, but interworking with M2M service layer security is possible	Gateway may act according to any of the scenarios in clause 6.
Leverage on M2M architecture capabilities	Full	
Example of applicability	This interworking scenario is making the network applications independent from the area network technologies. One typical scenarios is the case of an application that has to deal with multiple area network technologies (e.g. in case of long term deployments when the available technologies are changing), so the interworking is confinated to the new deployments.	

Scenario 4: Native interworking on dIa		
Type of device	**D'**	**Notes**
Application on device	Independent from Specific technology	The application is ETSI M2M native and independent for the interworked technology.
Application on Network	Independent from Specific technology	The application is ETSI M2M native and independent for the interworked technology.
Mechanism	dIa transport on binding layer between D' and G	Natively supported in ETSI M2M.
Security impact	Enables End-to-end encryption to/from D' devices	Gateway may act according to any of the scenarios in clause 6.
Leverage on M2M architecture capabilities	Full	
Example of applicability	This is the case of a technology supporting HPPT/COAP in case of deployment of ETSI M2M compliant DA and NA. It allows a complete independence of applications from area network technology. Typical.	

Scenario 5: Network based interworking		
Type of device	**d**	**Notes**
Application on device (d, non using dIa)	Specific technology aware	
Application on Network	Independent from Specific technology	The application is ETSI M2M native and independent for the interworked technology.
Mechanism	NIP interworking	Application needs to be able to handle encrypted data in containers.
Security impact	Use technology-specific security over MAN	Limited confidentiality as interworking may require application-related information to be exposed in the service layer.
Leverage on M2M architecture capabilities	low, level depends on specific solutions	
Example of applicability	This is to interwork with completely specific solutions already deployed without touching the G/D. One typical scenario is the introduction of ETSI M2M compliant solution for new services reusing already deployed legacy D/G.	

5 Interworking with legacy devices (d)

5.1 Implementation profile 1

5.1.1 Entity-relation representation of the M2M area network

Figure 5.1 provides a resource-entity model that represents an M2M area network as well as its relationship to an Interworking Proxy Application (IPA).

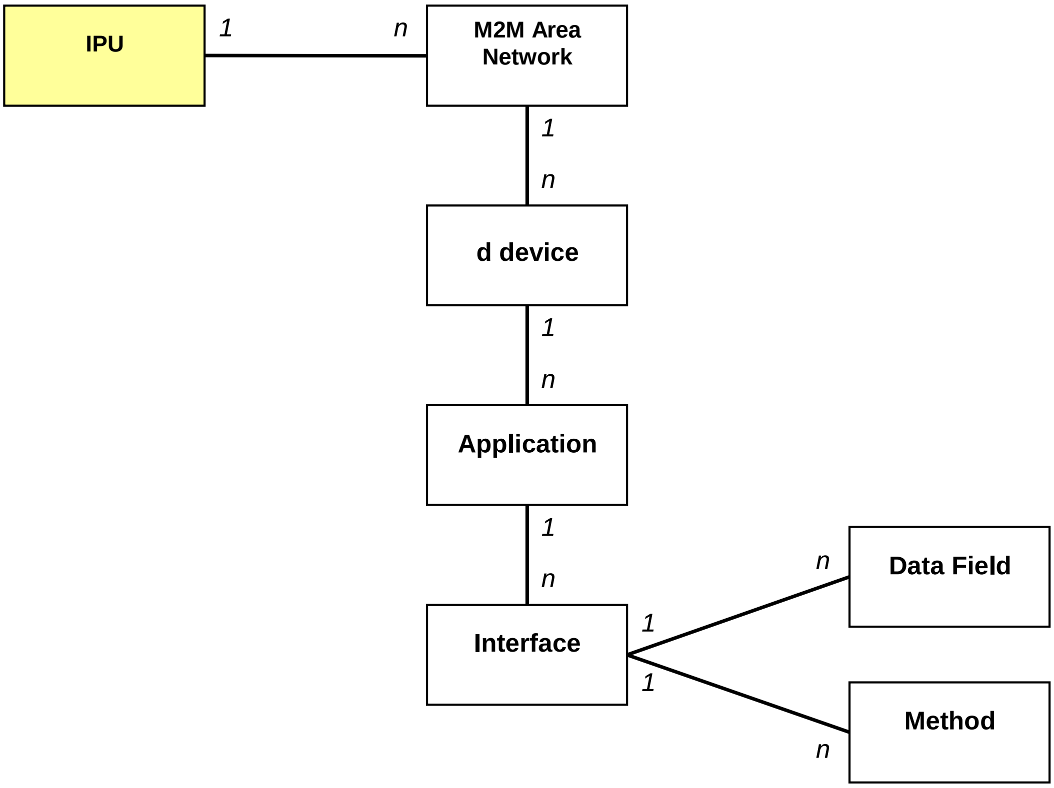

Figure 5.1: Generic entity-relation diagram for an IPU and an M2M Area Network running legacy d devices

This entity-relation diagram is applicable to the following M2M Area Networks:

- ZigBee®
- DLMS/COSEM
- Zwave
- BACnet
- ANSI C12
- mBus

5.1.2 Mapping principles

NOTE: The mapping principles proposed in the present document are initial ones, some others may exist.

This clause describes the mapping principles that are used to map a generic M2M Area Network into a structured tree of ETSI M2M resources in this implementation profile.

More specifically, the IPU is responsible to:

- discover the M2M Area Network structure;
- create an ETSI M2M resource structure representing the M2M Area Network structure in the ETSI M2M Service Capability Layer; and
- manage the ETSI M2M resource structure in case the M2M Area Network structure changes.

In order to facilitate the navigation through the various resources representing the M2M Area Network structure, created by the IPU, a specific format for the searchString attribute of the resources is used. This specific format is referred to as a Tag, and it is specified in annex A. These tags help locate M2M Area Network elements modeled as ETSI M2M resources.

The rules the IPU follows to create the ETSI M2M resource structure are the following:

- The IPU is modeled with an ETSI M2M <application> resource. The "searchString" attribute of this resource contains an ETSI.ObjectType/ETSI.IP tag which identifies it as an IPU. The URI used to access this <application> resource has the following format:

  <sclBase>/applications/< interworking_proxy_application>

  The <application> resource contains an ETSI M2M <container> sub resource. The "searchString" attribute of this sub resource contains a tag of category ETSI.ObjectSemantic which indicates the semantic conventions used in the representation of this object. The URI used to access this <container> resource has the following format:

  <sclBase>/applications/< interworking_proxy_application>/containers/descriptor

  The <container> resource contains one or more <contentInstance> sub resource. The "content" attribute of this sub resource contains the representation of the IPU. In particular, since a single IPU can give access to multiple M2M Area Networks, each of them modeled with an ETSI M2M resource (see next bullet for description), the "content" attribute of the <contentInstance> resource may contain the URIs of the ETSI M2M resources representing these M2M Area Networks. The URI used to access the <contentInstance> resource containing the current representation of the IPU has the following format:

  <sclBase>/applications/< interworking_proxy_application>/containers/descriptor/contentInstances/latest

  The reason is that a new <contentInstance> resource is created each time the IPU representation changes (e.g. a new M2M Area Network is created, or an old one is deleted). So, in case a new <contentInstance> resource is created and the old ones are kept in order to maintain an history, there can be more than one <contentInstance> resources. But, in any case, the <contentInstance> resource pointed by the "latest" attribute of the contentInstances resource contains always the current representation of the IPU.

- Each M2M Area Network controlled by an IPU is modeled with an ETSI M2M <application> resource. The "searchString" attribute of this resource contains an ETSI.ObiectType/ETSI.AN_NWK tag which identifies it as an M2M Area Network. The URI used to access this <application> resource has the following format:

  <sclBase>/applications/<networkX>

  The <application> resource contains an ETSI M2M <container> sub resource. The "searchString" attribute of this sub resource contains a tag of category ETSI.ObjectSemantic which indicates the semantic conventions used in the representation of this object. The URI used to access this <container> resource has the following format:

  <sclBase>/applications/<networkX>/containers/descriptor

The <container> resource contains one or more <contentInstance> sub resource. The "content" attribute of this sub resource contains the representation of the M2M Area Network. In particular, since a single M2M Area Network can be composed by several Devices (N.B.: they are not ETSI M2M Devices), each of them modeled with an ETSI M2M resource (see next bullet for description), the "content" attribute of the <contentInstance> resource may contain the URIs of the ETSI M2M resources representing these Devices. The URI used to access the <contentInstance> resource containing the current representation of the M2M Area Network has the following format:

<sclBase>/applications/<networkX>/containers/descriptor/contentInstances/latest

The reason is that a new <contentInstance> resource is created each time the M2M Area Network representation changes (e.g. a new Device is created, or an old one is deleted). So, in case a new <contentInstance> resource is created and the old ones are kept in order to maintain an history, there can be more than one <contentInstance> resources. But, in any case, the <contentInstance> resource pointed by the "latest" attribute of the contentInstances resource contains always the current representation of the M2M Area Network.

- Each Device belonging to an M2M Area Network (N.B.: they are not ETSI M2M Devices) is modeled with an ETSI M2M <application> resource. The "searchString" attribute of this resource contains an ETSI.ObiectType/ETSI.AN_NODE tag which identifies it as a Device belonging to an M2M Area Network. The URI used to access this <application> resource has the following format:

  <sclBase>/applications/<networkX_deviceY>

  The <application> resource contains an ETSI M2M <container> sub resource. The "searchString" attribute of this sub resource contains a tag of category ETSI.ObjectSemantic which indicates the semantic conventions used in the representation of this object. The URI used to access this <container> resource has the following format:

  <sclBase>/applications/<networkX_deviceY>/containers/descriptor

  The <container> resource contains one or more <contentInstance> sub resource. The "content" attribute of this sub resource contains the representation of the Device. In particular, since a Device can contain several Applications (N.B.: they are not ETSI M2M Applications), each of them modeled with an ETSI M2M resource (see next bullet for description), the "content" attribute of the <contentInstance> resource may contain the URIs of the ETSI M2M resources representing these Applications. The URI used to access the <contentInstance> resource containing the current representation of the Device has the following format:

  <sclBase>/applications/<networkX_deviceY>/containers/descriptor/contentInstances/latest

  The reason is that a new <contentInstance> resource is created each time the Device representation changes (e.g. a new Application is created, or an old one is deleted). So, in case a new <contentInstance> resource is created and the old ones are kept in order to maintain an history, there can be more than one <contentInstance> resources. But, in any case, the <contentInstance> resource pointed by the "latest" attribute of the contentInstances resource contains always the current representation of the Device.

- Each Application belonging to a Device (N.B.: they are not ETSI M2M Applications) is modeled with an ETSI M2M <application> resource. The "searchString" attribute of this resource contains an ETSI.ObiectType/ETSI.AN_APP tag which identifies it as an Application belonging to a Device. The URI used to access this <application> resource has the following format:

  <sclBase>/applications/<networkX_deviceY_applicationZ>

  The <application> resource contains an ETSI M2M <container> sub resource. The "searchString" attribute of this sub resource contains a tag of category ETSI.ObjectSemantic which indicates the semantic conventions used in the representation of this object. The URI used to access this <container> resource has the following format:

  <sclBase>/applications/<networkX_deviceY_applicationZ>/containers/descriptor

The <container> resource contains one or more <contentInstance> sub resource. The "content" attribute of this sub resource contains the representation of the Application. In particular, since an Application can implement several Interfaces, each of them modeled with ETSI M2M resources (see next bullet for description), the "content" attribute of the <contentInstance> resource may contain the URIs of the ETSI M2M resources representing these Interfaces. The URI used to access the <contentInstance> resource containing the current representation of the Application has the following format:

<sclBase>/applications/<networkX_deviceY_applicationZ>/containers/descriptor/contentInstances/latest

The reason is that a new <contentInstance> resource is created each time the Application representation changes (e.g. a new Interface is created, or an old one is deleted). So, in case a new <contentInstance> resource is created and the old ones are kept in order to maintain an history, there can be more than one <contentInstance> resources. But, in any case, the <contentInstance> resource pointed by the "latest" attribute of the contentInstances resource contains always the current representation of the Device.

- Each Data Field and each Method belonging to an Interface implemented by an Application can be mirrored or retargeted. Mirroring is defined as the set of mechanism to keep a data field synchronized with its representation in the M2M resource structure. Retargeting is defined as the mechanism that allows fetching the data directly from the device, that is without storing the data in the M2M resource structure.

  If the Data Field or the Method is mirrored the ETSI M2M <application> resource modeling the Application contains an ETSI M2M <container> sub resource for each interface element mirrored (either Data Field or Method). The URI used to access this <container> resource has the following format:

  <sclBase>/applications/<networkX_deviceY_applicationZ>/containers/<interfaceW_datafieldN>

  or

  <sclBase>/applications/<networkX_deviceY_applicationZ>/containers/<interfaceW_methodM>

  The <container> resource contains one or more <contentInstance> sub resource. The "content" attribute of this sub resource contains the representation of the Data Field or the Method; for the Data Field it is its value, for the Method it is the actual parameters used for a Method invocation or the result of a Method invocation. The URI used to access the <contentInstance> resource containing the current representation of the Data Field or the Method has the following format:

  <sclBase>/applications/<networkX_deviceY_applicationZ>/containers/<interfaceW_datafieldN>/ contentInstances/latest

  or

  <sclBase>/applications/<networkX_deviceY_applicationZ>/containers/<interfaceW_methodM>/ contentInstances/latest

  For the <container> resources representing Data Fields the IPU creates a new <contentInstance> resource each time the value of the Data Field changes in the M2M Area Network, and subscribes for the creation of <contentInstance> resources by M2M Applications; when a new <contentInstance> resource is created the IPU changes the value of the Data Field into the M2M Area Network. For the <container> resources representing Methods the IPU subscribes for the creation of <contentInstance> resources by M2M Applications; when a new <contentInstance> resource is created the IPU invokes the Method into the M2M Area Network with the specified parameters; the result of the Method invocation will be contained in another <contentInstance> resource created by the IPU with the same name but different "creationTime" attribute.

  If the Data Field or the Method is retargeted the "content" attribute of the <contentInstance> resource of the <container> sub resource of the <application> resource modeling the Application contains a URI pointing to the IPU in which the path identifies the specific Data Field or Method. In this way the IPU is able to forward the operation to the original Area Network resource acting as a back-to-back proxy.

Figure 5.2 provides an overview of the resources used to model an example of M2M Area Network.

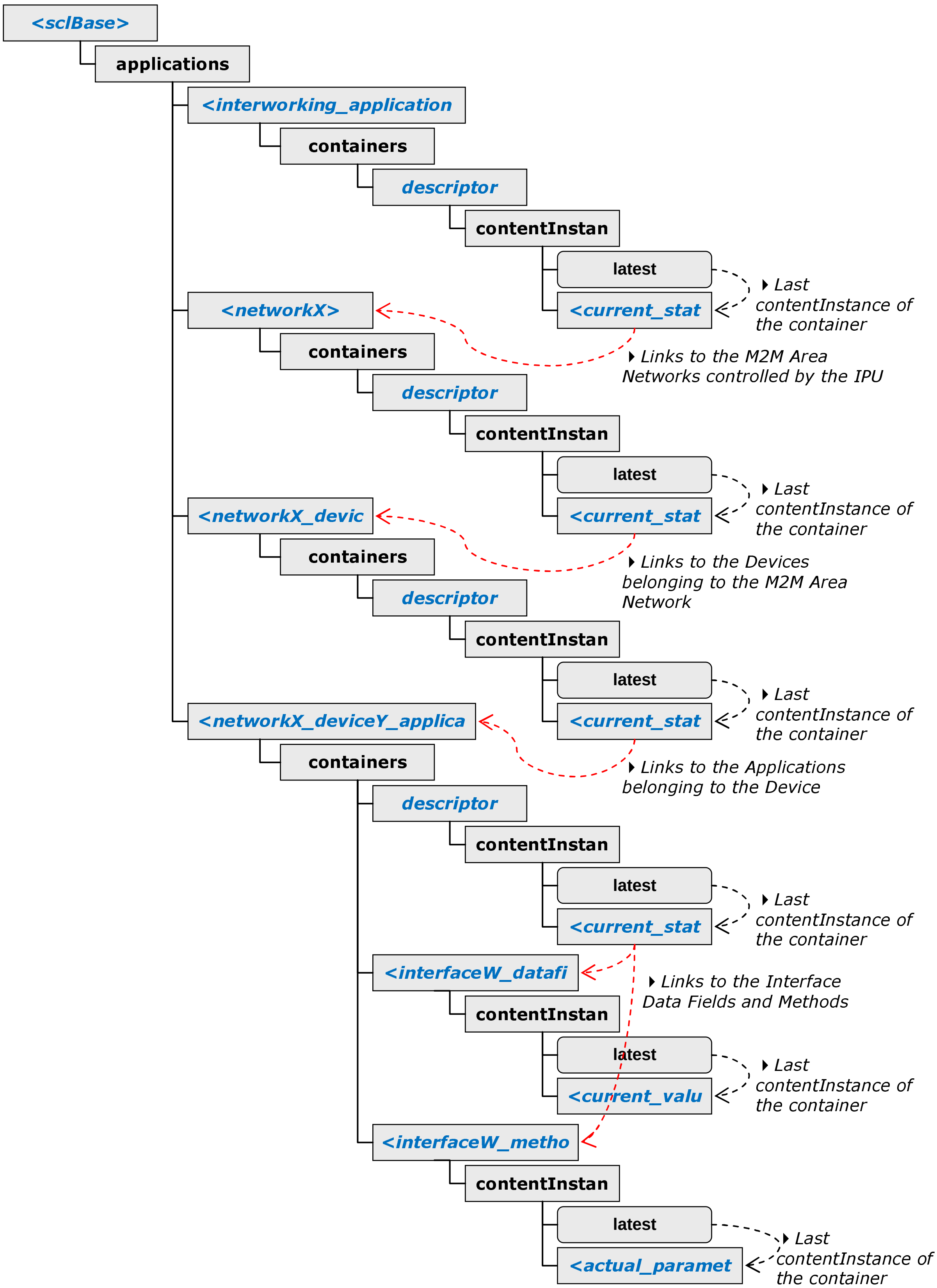

Figure 5.2: Mapping of an M2M Area Network to the ETSI M2M resource architecture

5.1.3 M2M Area Network specific technologies interworking

In the following clauses the interworking between the ETSI M2M Architecture and specific M2M Area Network technologies will be described. The generic mapping principles presented in the previous clause are applied to specific cases and peculiar guidelines are given.

5.1.3.1 ZigBee® Alliance

ZigBee® is a low-power wireless mesh network. Release 1.x of the standard is based on IEEE 802.15.4-2003 [i.3] which provides physical and MAC layer specifications. ZigBee® specifications provide support for the network layer as well as the application framework layer.

ZigBee® specifies 3 kinds of devices:

- ZigBee® Coordinator (ZC): The most capable device, the coordinator forms the root of the network tree and might bridge to other networks. There is exactly one ZigBee® coordinator in each network since it is the device that started the network originally.
- ZigBee® Router (ZR): As well as running an application function, a router can act as an intermediate router, passing on data from other devices.
- ZigBee® End Device (ZED): Contains just enough functionality to talk to the parent node (either the coordinator or a router); it cannot relay data from other devices. This relationship allows the node to be asleep for a significant amount of the time thereby enabling longer battery powered operation.

The ZigBee® specifications define multiple types of profiles which represent classes of devices that differ based on the application layer they implement. The following profiles have been specified by the ZigBee® Alliance:

- Smart Energy: specifies functionalities of devices that pertain to energy management (e.g. thermostat, meter, etc.).
- Health Care: specifies functionalities of devices that pertain to eHealth devices.
- Building Automation: specifies functionalities of devices that pertain to building automation (light, air conditioning, etc.).
- Home Automation: specifies functionalities of devices that pertain to home automation (light, air conditioning, etc.).
- ZigBee® Telecommunication Services: specifies functionalities pertaining to telecommunications (e.g. advertisement, gaming, social networks, etc.) that can be implemented in devices such as cellular phones, tablets, netbooks, etc.

The ZigBee® Cluster Library (ZCL) lists a number of standardized application interfaces, referred to as clusters. Each standardized Zigbee® Profile relies on specific set of clusters specified in the ZCL. Vendors can also define their own clusters. A particular ZigBee® device typically runs one or more Applications (modeled as ZigBee® endpoints) that implement one or more clusters belonging to a particular profile. Each cluster is a collection of attributes (that represent e.g. a value of thermometer), and commands, which together define a communications interface between two ZigBee® devices implementing the server and client sides of the interface respectively.

In the remaining of the present document a Cluster (logical grouping of attributes, commands and events) will be referred to as an Interface which is a more generic term that can be used for mapping of other technologies.

A ZigBee® network can be represented as follows:

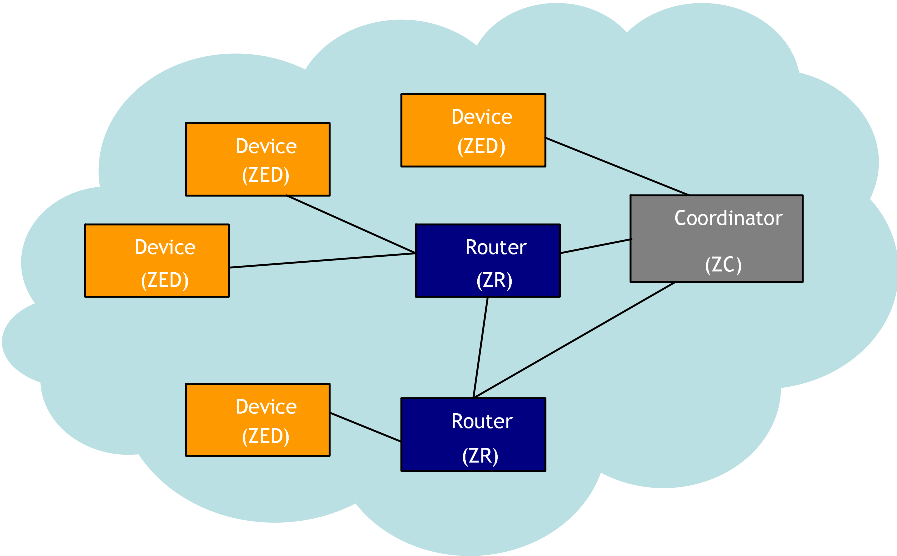

Figure 5.3: A typical representation of ZigBee® network

The mapping between a ZigBee® device, ZigBee® Applications and Interfaces (Clusters) is provided in figure 5.4.

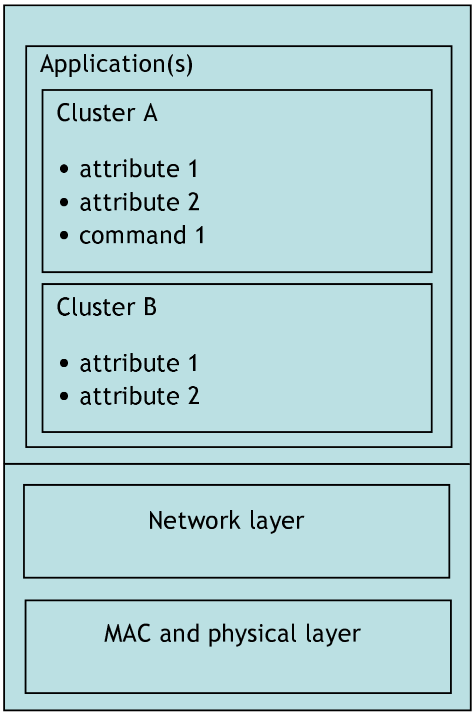

Figure 5.4: A ZigBee® Device Reference architecture

5.1.3.1.1 Implementation profile 1 for ZigBee® PAN interworking with ETSI M2M

This clause specifies a mapping of ZigBee® PAN entities to the ETSI M2M SCL resource structure. The generic mapping principles are applied to this specific M2M Area Network technology. The following ZigBee® PAN entities (with reference to figure 5.1) are considered and mapped to the ETSI M2M resource structure:

- **ZigBee® interworking proxy application (ZigBee IPA):** this is the implementation of an IPU for the ZigBee technology

- **ZigBee® network:** this is equivalent to M2M Area Network
- **ZigBee® node:** this is equivalent to M2M Area Network Device
- **ZigBee® application:** this is equivalent to M2M Area Network Application
- **ZigBee® cluster:** this is equivalent to M2M Area Network Interface
- **ZigBee® cluster attribute:** this is equivalent to M2M Area Network Data Field
- **ZigBee® cluster command:** this is equivalent to M2M Area Network Method

5.1.3.1.2 ZigBee® Interworking Proxy Application resource structure

The <application> resource representing a ZigBee® IPU has a searchString containing the tag ETSI.ObjectType/ETSI.IP.

It contains one <container> sub-resource which has a searchString containing the tag ETSI.ObjectType/ETSI.IP, and a tag of category ETSI.ObjectSemantic. The ZigBee® IPU may update its representation (e.g. add or remove a network) by creating newer <contentInstance> resources.

This implementation profile does not limit the representations of a ZigBee® IPU resource. Published representations are listed in annex B. An example of the content of the contentInstance with the list of ZigBee® networks is shown here:

```
...
<list name="networks"/>
  <ref href="/<sclBase>/applications/<networkX>/">
</list>
...
```

Figure 5.5: Example of ZigBee® Interworking Proxy Application

5.1.3.1.3 ZigBee® network resource structure

The <application> resource representing a ZigBee® network has a searchString containing the tag ETSI.ObjectType/ETSI.AN_NWK.

It contains one <container> sub-resource which has a searchString containing the tag ETSI.ObjectType/ETSI.AN_NWK, and a tag of category ETSI.ObjectSemantic. The network representation may be updated (e.g. add or remove a node) by creating newer <contentInstance> resources.

This implementation profile does not limit the representations of a ZigBee® network resource. Published representations are listed in annex B. An example of the content of the contentInstance with the list of ZigBee® nodes is shown here:

```
...
<str name="extendedPanID" val"0x685B3C34"/>

<list name="nodes">
  <ref href="/<sclBase>/applications/<networkX_nodeY>/"/>
</list>
...
```

Figure 5.6: Example of ZigBee® Network

5.1.3.1.4 ZigBee® node resource structure

The <application> resource representing a ZigBee® node has a searchString containing the tag ETSI.ObjectType/ETSI.AN_NODE.

It contains one <container> sub-resource which has a searchString containing the tag ETSI.ObjectType/ETSI.AN_NODE, and a tag of category ETSI.ObjectSemantic. The node representation may be updated (e.g. add or remove an application) by creating newer <contentInstance> resources.

This implementation profile does not limit the representations of a ZigBee® node resource. Published representations are listed in annex B. An example of the content of the contentInstance with the list of ZigBee® applications is shown here:

```
...
<str name="ieeeAddress" val="0x685B3C88"/>
<enum name="type" val="endDevice"/>

<list name="applications">
  <ref href="/<sclBase/applications/<networkX_nodeY_applicationZ>/"/>
</list>
...
```

Figure 5.7: Example of ZigBee® Node

5.1.3.1.5 ZigBee® application resource structure

The <application> resource representing a ZigBee® application has a searchString containing the tag ETSI.ObjectType/ETSI.AN_APP.

It contains at least one <container> sub-resource which has a searchString containing the tag ETSI.ObjectType/ETSI.AN_APP, and a tag of category ETSI.ObjectSemantic. The application representation may be updated (e.g. add or remove an interface) by creating newer <contentInstance> resources.

The ZigBee® application resource may have a ZigBee.ApplicationProfile tag and a ZigBee.DeviceIdentifier tag, with values matching those of the node ZigBee® Simple Descriptor.

This implementation profile does not limit the representations of a ZigBee® application resource. Published representations are listed in annex B. An example of the content of the contentInstance with the description of an application is shown below, which illustrates the representation of a ZigBee® on/off light application.

```
...
<int name="endpoint" val="1"/>
<int name="applicationProfileID" val="0x0104"/>
<int name="applicationDeviceID" val="0x0100"/>

<list name="Interfaces">
  <obj>
    <str name="clusterID" val="0x0006"/>
    <enum name="clusterType" val="input"/>

    <list name="attributes">
      <ref name="0x0000"

  href="/<sclBase>/applications/<networkX_nodeY_applicationZ>/containers/0x0006_OnOff"/>
    </list>

    <list name="operations">
      <op name="0x00" href="/<sclBase>/applications/<interworking_proxy_application>/0x0006_off"/>
      <op name="0x01" href="/<sclBase>/applications/<interworking_proxy_application>/0x0006_on"/>
      <op name="0x02"
href="/<sclBase>/applications/<interworking_proxy_application>/0x0006_toggle"/>
    </list>
  </obj>
</list>
...
```

Figure 5.8: Example of ZigBee® Application

5.1.3.1.6 Use of mirroring or retargeting for ZigBee® interfaces (clusters)

The representation of a ZigBee® application includes elements which are dynamic in nature, e.g. the on/off status of a lamp or the execution of commands. The ETSI M2M model offers several alternatives to interface with such elements:

- Retargeting: retargeting enables the issuer application to interact directly with the ZigBee® IPU in order to execute commands or to retrieve parameters. In the example of **Figure 5.8**, the implementation chose to retarget each command. This method presents performance advantages for dynamic or infrequently accessed elements or operations. On the other hand, this method requires the ZigBee® IPU to have server capabilities, and does not leverage the SCL caching and logging capabilities. If many applications are interested in a given attribute, the burden of responding to read commands may be offloaded from the ZigBee® IPU to the SCL by using mirroring.

- Mirroring: The ZigBee® IPU may leverage the mirroring capabilities of the SCL by referring to a container which mirrors the element value in the application representation. In the example of **Figure 5.8**, the implementation chose to mirror each cluster attribute in a separate application container. This method makes it easier for applications to subscribe to individual attribute values. If the attribute values change in a non-correlated way and if the implementation keeps a history of changes, using separate containers for each attribute will reduce the size of history datasets. Mirroring may provide an advantage in terms of delay and M2M area network bandwidth usage for data that is sent upstream. For data sent downstream, the use of the retargeting mechanism may be more optimal.

The content of a container representing a cluster attribute will contain its value formatted as per the selected semantic, and the Access Rights resource associated reflects the access defined in the ZigBee® specification of the cluster (Read only or Read/Write). These constraints cannot be enforced by the ETSI M2M SCL, since it is content agnostic, so the correctness of the content and of the Access Rights is a contract between the IPU creating the M2M resources to represent the ZigBee® PAN entities and the ETSI M2M Application using them.

The content of a container representing a cluster command will contain the payload formatted as per the ZigBee® specification of the cluster (list of parameters and type/format of parameters). These constraints cannot be enforced by the ETSI M2M SCL, since it is content agnostic, so the correctness of the content is a contract between the IPU creating the M2M resources to represent the ZigBee® PAN entities and the ETSI M2M Application using them.

Handling of asynchronous commands will use the general mechanisms defined in TS 102 690 [i.1].

5.1.3.2 UPnP

UPnP is an IP based network technology for pervasive network. It defines an architecture for peer to peer network connectivity of intelligent appliances, wireless devices, and PCs of all form factors. It is designed to bring easy-to-use, flexible, standards-based connectivity to ad-hoc or unmanaged networks whether in the home, in a small business, public spaces, or attached to the Internet.

The UPnP architecture defines the protocols for communication between controllers, control points, and devices.

The UPnP Architecture uses the following protocol stack for discovery, description, control, eventing, and presentation.

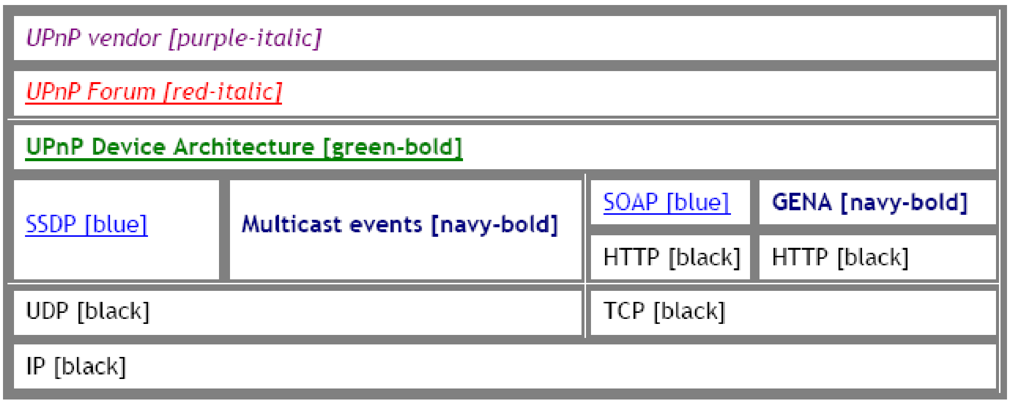

Figure 5.9: UPnP Protocol Stack

There are three major classes of UPnP device:

- **UCP (User/Universal Control Point):** this is a device, such as a PC or PDA, which allows for control of other UPnP devices through the presentation page and rich display.
- **Controlled Device:** any UPnP device that allows control or provides some sort of UPnP service to the rest of the home network (such as IGDs, A/V devices, security cameras, etc.).
- **Bridge:** connects non-UPnP devices to the home network; in essence it speaks UPnP on one end and some proprietary language on the other end (some examples include proprietary lighting control, Bluetooth, HAVi, etc.).

Figure 5.10 shows the UPnP networking.

Figure 5.10: UPnP Networking

- **Addressing**

  IP addressing is the foundation of UPnP networking. Each device supports DHCP mechanism when the device is first connected to the network. If during the DHCP transaction, the device obtains a domain name, for example, through a DNS server or via DNS forwarding, the device should use that name in subsequent network operations; otherwise, the device should use its IP address.

- **Discovery**

  When a device is added to the network, the UPnP discovery protocol allows that device to advertise its services to control points on the network. Similarly, when a control point is added to the network, the UPnP discovery protocol allows that control point to search for devices of interest on the network. The fundamental exchange in both cases is a discovery message containing a few essential specifics about the device or one of its services, e.g. its type, identifier, and a pointer to more detailed information.

- **Description**

  After a control point has discovered a device, the control point still knows very little about the device. For the control point to learn more about the device and its capabilities, or to interact with the device, the control point retrieves the device's description from the URL provided by the device in the discovery message. The UPnP description for a device is expressed in XML and includes vendor-specific, manufacturer information like the model name and number, serial number, manufacturer name, URLs to vendor-specific web sites, etc. The description also includes a list of any embedded devices or services, as well as URLs for control, eventing, and presentation. For each service, the description includes a list of the commands, or actions, to which the service responds, and parameters, or arguments, for each action; the description for a service also includes a list of variables; these variables model the state of the service at run time, and are described in terms of their data type, range, and event characteristics.

- **Control**

  Having retrieved a description of the device, the control point can send actions to a device's service. To do this, a control point sends a suitable control message to the control URL for the service (provided in the device description). Control messages are also expressed in XML using the Simple Object Access Protocol (SOAP). Much like function calls, the service returns any action-specific values in response to the control message. The effects of the action, if any, are modeled by changes in the variables that describe the run-time state of the service.

- **Eventing**

  An additional capability of UPnP networking is event notification, or eventing. The event notification protocol defined in the UPnP Device Architecture is known as General Event Notification Architecture (GENA). A UPnP description for a service includes a list of actions the service responds to and a list of variables that model the state of the service at run time. The service publishes updates when these variables change, and a control point may subscribe to receive this information. The service publishes updates by sending event messages. Event messages contain the names of one or more state variables and the current value of those variables. These messages are also expressed in XML. A special initial event message is sent when a control point first subscribes; this event message contains the names and values for all evented variables and allows the subscriber to initialize its model of the state of the service. To support scenarios with multiple control points, eventing is designed to keep all control points equally informed about the effects of any action. Therefore, all subscribers are sent all event messages, subscribers receive event messages for all "evented" variables that have changed, and event messages are sent no matter why the state variable changed (either in response to a requested action or because the state the service is modeling changed).

- **Presentation**

  The final step in UPnP networking is presentation. If a device has a URL for presentation, then the control point can retrieve a page from this URL, load the page into a web browser, and depending on the capabilities of the page, allow a user to control the device and/or view device status. The degree to which each of these can be accomplished depends on the specific capabilities of the presentation page and device.

5.1.3.2.1 Implementation profile 1 for UPnP interworking with ETSI M2M

This clause specifies a mapping of UPnP entities to the ETSI M2M SCL resource structure. The generic mapping principles are applied to this specific M2M Area Network technology. The following 6LoWPAN PAN entities (with reference to figure 5.1) are considered and mapped to the ETSI M2M resource structure:

- **UPnP interworking proxy application (UPnP IPA):** this is the implementation of an IPU for the UPnP technology.
- **UPnP network:** this is equivalent to M2M Area Network.
- **UPnP node:** this is equivalent to M2M Area Network Device.

The UPnP interworking proxy application uses the following standards.

Area network standard	ANStandard extension
UPnP v1.1	`<obj name="UPnP_1.1"/>`

5.1.3.2.2 UPnP Interworking Proxy Application resource structure

An UPnP InterworkingDescriptor contract is defined for interworking proxies supporting the UPnP standard. This contract overloads the M2M InterworkingDescriptor contract and contains no additional mandatory sub-elements.

```
<obj href="upnp:InterworkingDescriptor" is="m2m:InterworkingDescriptor"
      xmlns="http://obix.org/ns/wsdl/1.1"
      xmlns:m2m="http://uri.etsi.org/m2m/obix"
      xmlns:upnp="http://uri.etsi.org/m2m/upnp/obix">
</obj>
```

5.1.3.2.3 UPnP network resource structure

A UPnP NetworkDescriptor contract is defined for interworking proxies supporting the UPnP standard. This contract overloads the M2M NetworkDescriptor contract and contains the following additional mandatory sub-elements:

- "domainName": The domain name of UPnP network for supporting Dynamic Domain Name System (DDNS).

```
<obj href="upnp:NetworkDescriptor" is="m2m:NetworkDescriptor"
      xmlns="http://obix.org/ns/wsdl/1.1"
      xmlns:m2m="http://uri.etsi.org/m2m/obix"
      xmlns:upnp="http://uri.etsi.org/m2m/upnp/obix">

  <str name=”domainName”/>
</obj>
```

5.1.3.2.4 UPnP node resource structure

An UPnP NodeDescriptor contract is defined for interworking proxies supporting the UPnP standard. This contract overloads the M2M NodeDescriptor contract and contains the following additional mandatory sub-elements:

- "configID": configuration identifier to which the device description.
- "specVersion": defines the architecture version on which the device is implemented.
- "deviceType": UPnP device type.
- "friendlyName": Short description for end user. Specified by UPnP vendor.
- "manufacturer": Manufacturer's name.
- "modelName": Model name.
- "modelnumber": Model number.
- "serialNumber": Serial number.
- "UDN": Unique device name.

```
<obj href="upnp:NodeDescriptor" is="m2m:NodeDescriptor"
      xmlns="http://obix.org/ns/wsdl/1.1"
      xmlns:m2m="http://uri.etsi.org/m2m/obix"
      xmlns:upnp="http://uri.etsi.org/m2m/upnp/obix">

  <str name=”configID”/>
  <str name=”specVersion”/>
  <str name=”devieType”/>
  <str name=”friendlyName”/>
  <str name=”manufacturer”/>
  <str name=”modelName”/>
  <str name=”modelNumber”/>
  <str name=”serialNumber”/>
  <str name=”UDN”/>
</obj>
```

5.1.3.2.5 UPnP service resource structure

A UPnP AppplicationDescriptor contract is defined for interworking proxies supporting the UPnP standard. This contract overloads the M2M ApplicationDescriptor contract and contains the following additional mandatory sub-elements:

- "configID": configuration identifier to which the service description
- "specVersion": defines the architecture version on which the service is implemented
- "action": action defined by a UPnP Forum working committee. It contains the following sub-elements:
  - "name": action name
  - "argument": parameter defined for action
  - "direction": defines whether argument is an input or output

- "serviceState": service state variable defined by a UPnP Forum working committee. It contains the following sub-elements:
    - "name": name of state variable
    - "datatype": datatype for state variable
    - "defaultValue": Expected, initial value. Defined by a UPnP Forum working committee
    - "allowedValuelist": Enumerates legal string values
    - "allowedvalueRange": Defines bounds for legal numeric values

```
<obj href="upnp:ApplicationDescriptor" is="m2m:ApplicationDescriptor"
     xmlns="http://obix.org/ns/wsdl/1.1"
     xmlns:m2m="http://uri.etsi.org/m2m/obix"
     xmlns:upnp="http://uri.etsi.org/m2m/upnp/obix">

  <str name=”configID”/>
  <str name=”specVersion”/>
  <list name=”action”/>
   <obj>
     <str name=”name”/>
     <str name=”argument”/>
     <str name=”direction”/>
   </obj>
  </list>
  <list name=”serviceState”/>
    <obj>
     <str name=”name”/>
     <str name=”datatype”/>
     <str name=”defaultValue”/>
      <str name=”allowedValuelist”/>
      <str name=”allowedValueRange”/>
   </obj>
  </list>
</obj>
```

5.1.3.3 KNX™

The KNX™ architecture is decentralized. KNX™ nodes can interact with other nodes without a need for a central controller. KNX™ nodes are organized in zones (e.g. zone 1) and lines (e.g. line 1.0).

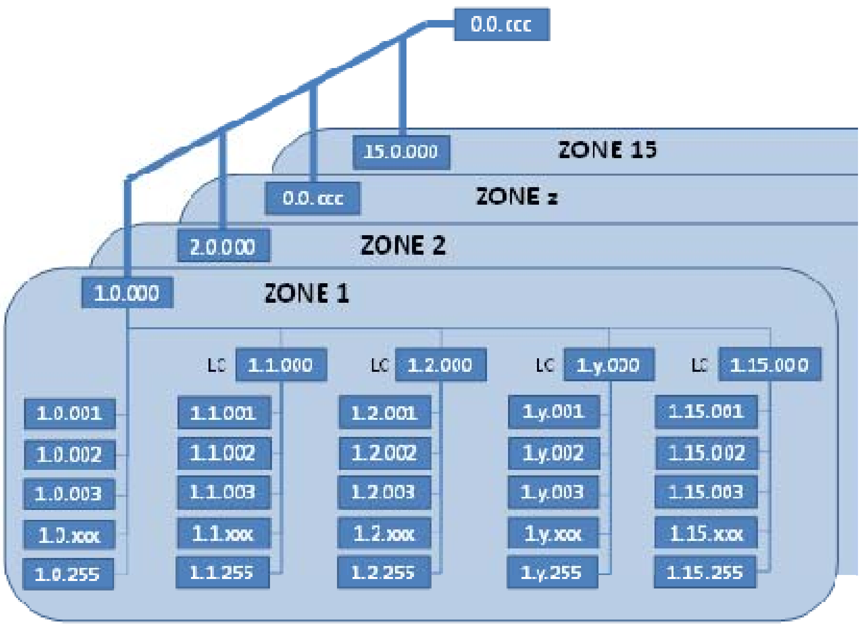

Figure 5.11: KNX™ network topology

According to the KNX™ network topology, each KNX™ node has a physical address (e.g. 1.0.001), used mainly for configuration purposes. This physical address is configured during the installation process.

Each KNX™ node is associated to one or several group objects. A group object may be read or written over the KNX™ bus via an association between the group object and a multicast group address (e.g. 1/0/0). The type of the group object is described by a data point type (DPT). A group object which sends its value may be configured with one and only one destination group address, but a group object may listen to several group addresses. Targeted (write) and monitored (read) group addresses are configured in the KNX™ node for each group object. All group objects linked to the same group address use the same data point type (DPT). Group addresses are configured during the installation process.

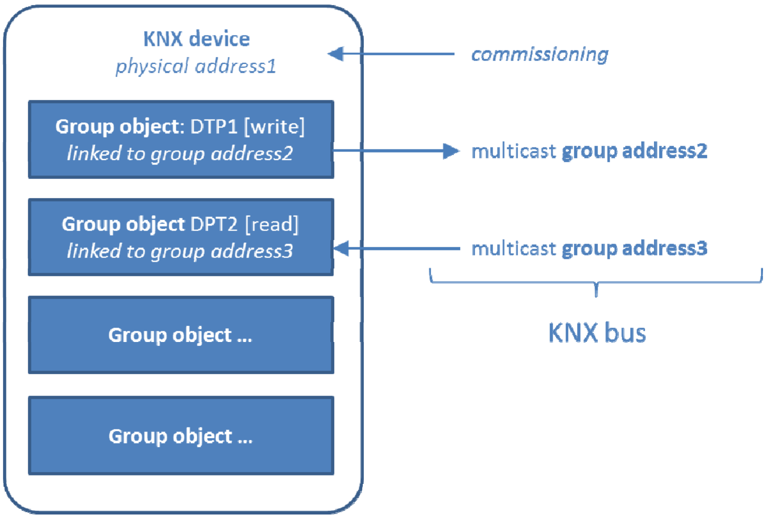

Figure 5.12: KNX™ node

For example, if the On/Off group object (e.g. DPT_Switch) of a Presence sensor (an output) is assigned to the group address 1/0/0, and the On/Off group object (e.g. DPT_Switch) of a Light controller (an input) is also assigned to the group address 1/0/0, then the Presence sensor will control the Light controller.

5.1.3.3.1 Implementation profile 1 for KNX™ PAN interworking with ETSI M2M

This clause specifies a mapping of KNX™ PAN entities to the ETSI M2M SCL resource structure. The generic mapping principles are applied to this specific M2M Area Network technology. The following KNX™ entities are considered and mapped to the ETSI M2M resource structure:

- **KNX™ Interworking Proxy (e.g. KNX™ IP gateway):** this is the implementation of an IPU for the KNX™ technology.
- **KNX™ Network:** this is equivalent to M2M Area Network.
- **KNX™ Node (a physical node address):** this is equivalent to M2M Area Network Device associated with a unique M2M Area Network Application.
- **KNX™ Group (a multicast group address):** this is equivalent to M2M Area Network Device associated with a unique M2M Area Network Application.
- **KNX™ DPT:** this is equivalent to M2M Area Network Interface.

5.1.3.3.2 KNX™ Interworking Proxy resource structure

The <application> resource representing a KNX™ IPU has an ETSI.ObjectType/ETSI.IP tag and contains one <container> sub-resource (the descriptor). The <container> sub-resource has an ETSI.ObjectType/ETSI.IP tag, and has a tag of the ETSI.ObjectSemantic category. The KNX™ IPU may update its representation (e.g. add or remove a network) by creating newer <contentInstance> resources.

Tags are listed in annex A. Published representations are listed in annex B.

5.1.3.3.3 KNX™ Network resource structure

The <application> resource representing a KNX™ network has an ETSI.ObjectType/ETSI.AN_NWK tag and contains one <container> sub-resource (the descriptor). The <container> sub-resource has an ETSI.ObjectType/ETSI.AN_NWK tag, and has a tag of the ETSI.ObjectSemantic category. The network representation may be updated (e.g. add or remove a node or a group) by creating newer <contentInstance> resources.

Tags are listed in annex A. Published representations are listed in annex B.

5.1.3.3.4 KNX™ Group resource structure

The <application> resource representing a KNX™ group has an ETSI.ObjectType/ETSI.AN_GRP tag and contains one <container> sub-resource (the descriptor). The <container> sub-resource has an ETSI.ObjectType/ETSI.AN_GRP tag, and has a tag of the ETSI.ObjectSemantic category. The group representation may be updated by creating newer <contentInstance> resources.

The <application> resource representing the associated M2M Area Network Application has an ETSI.ObjectType/ETSI.AN_APP tag, may have a tags of the KNX.DptID category, and contains one <container> sub-resource (the descriptor). The <container> sub-resource has an ETSI.ObjectType/ETSI.AN_APP tag, and has a tag of the ETSI.ObjectSemantic category. The application representation may be updated by creating newer <contentInstance> resources.

Tags are listed in annex A. Published representations are listed in annex B.

5.1.3.3.5 KNX™ Node resource structure

The <application> resource representing a KNX™ node has an ETSI.ObjectType/ETSI.AN_NODE tag, may have tags of the KNX.AreaID and KNX.LineID categories, and contains one <container> sub-resource (the descriptor). The <container> sub-resource has a ETSI.ObjectType/ETSI.AN_NODE tag, and has a tag of the ETSI.ObjectSemantic category. The node representation may be updated by creating newer <contentInstance> resources.

The <application> resource representing the associated M2M Area Network Application has an ETSI.ObjectType/ETSI.AN_APP tag, may have a tag of the KNX.DptID category, and contains one <container> sub-resource (the descriptor). The <container> sub-resource has an ETSI.ObjectType/ETSI.AN_APP tag, and has a tag of the ETSI.ObjectSemantic category. The application representation may be updated by creating newer <contentInstance> resources.

Tags are listed in annex A. Published representations are listed in annex B.

5.2 Implementation profile 2

Void.

6 Interworking with M2M devices without SCL (D')

6.1 Security considerations

From the point of view of the M2M core, M2M applications residing on D' devices are not necessarily distinguishable from DA or GA. However from the security point of view it makes a difference whether the dIa interface is internal to a D/G or is a potentially exposed link on an M2M Area Network, as in the case of D' devices. Even when appropriate security is implemented in the M2M Area Network to protect dIa, such protection will be out of control of the M2M Service layer, unless specific services are implemented at the Gateway level. Furthermore the experience of WiFi networks shows that users often neglect or are not skilled to setup proper security over their LAN. Therefore it would be valuable for the M2M System to assist applications in addressing MAN security issues, which could be done by coupling the security of the MAN with the Service Layer.

For this purpose, it is useful to consider how an M2M Gateway may interact in terms of security between its upstream mId interface and its downstreams dIa interfaces, and how this impacts the functionalities provided by the M2M Service Layer.

In the analysis below, we assume a general security scenario for M2M area networks, where security is ensured using a group key (referred below as Kg) shared between all devices on the MAN. However the analysis can be easily extended when separate key pairs are established to secure the communication between each individual devices and the Gateway. As an M2M Gateway can be expected to have more computing power than individual D' devices, it is reasonable to assume that it can coordinate the establishment and sharing of Kg between all devices on the MAN (this role is referred as "group leader" in the security literature). This handles the cases of mobile D' devices that may move between different WAN depending on where they are located: old key revocation on the previous network and new key establishment on the next one need to be handled securely. Whatever method is used (there are several in the security literature) for key establishment on the MAN can be left to M2M applications.

Though ETSI M2M release 1 seems to be built on the assumption that Gateways will be deployed and managed by the applications, it is interesting to note that there are scenarios where an M2M service provider would benefit in deploying and managing his own gateways to serve several applications using the same MAN technology: This is especially the case where roaming over capillary networks such as Zigbee® is involved.

6.1.1 D' devices traffic aggregation by Gateway acting as proxy/concentrator

In this basic scenario, only the M2M gateway establishes its own M2M Service Connection with the M2M Core over mId. The security downstream on the M2M Area Network uses a key Kg established locally and is completely independent from the security upstream, using M2M procedures for mId under the control of the Gateway. Therefore the gateway has to decrypt incoming D' device's traffic encoded with Kg, then reencrypt it over the WAN (encoded with the M2M session key as specified in TC M2M) as if it was its own traffic, and do the reverse decryption/reencryption for downstream traffic to the served D' devices.

The effect of this scenario is that individual D' devices are hidden from the M2M core, as only the Gateway is directly visible, and all resulting traffic cannot be identified as related to a specific D' device, since their identities are hidden by the Gateway.

This scenario implies that the gateway is always trusted by all applications using it for protecting their data, as it needs to decode all the information received. However in many M2M verticals there is a dominant MAN technology for which it makes sense to leverage on deployed M2M Gateways by opening them to different customers/applications, and there are also use cases (e.g. in Smart Cities) where leveraging on public gateways has to be considered. An extension of the current M2M security requirements would be needed to properly care for such scenarios, in order to ensure that each traffic stream (i.e. the aggregated private traffic streams and potential public traffic) transiting through shared gateways are properly segregated from each other.

6.1.2 MAN devices generating traffic with their own identity and security via gateway acting as multiplexer/funnel

The M2M gateway could instead act as a funnel to make each device on its MAN visible to the core network with its own identity, and each connected MAN device could be provided with its own M2M Service Layer security key. In meshed networks such as Zigbee®, such scenarios enable confidentiality of individual MAN devices traffic with respect to other MAN devices that may be involved in routing their communication up to the Gateway.

While in the most extreme scenario the "Gateway" may just act as a router extending the mId interface transparently to the M2M service layer, it still makes sense to consider how such a "router" could assist the M2M service layer to discharge the devices behind it from some burdening functionalities.

In all cases the intermediate Gateway makes the MAN devices under it visible as individual D devices to the M2M Core, each with their own M2M Node ID, implying that these devices support at least the M2M node functionalities for secure connection. However the Gateway may still assist the devices behind it in their bootstrapping process, assuming it is fully trusted. Furthermore this scenario does not necessarily imply that the devices behind the gateway support a local SCL, they may instead be served by the SCL in the Gateway (i.e. they all share a common SCL-ID). Such MAN devices seen as D devices could still be considered as D' devices in the sense that they do not have their own SCL (they just share the SCL of the gateway), however they would need to implement some M2M Node functionalities, which is a split of functionalities between devices and Gateway not currently addressed by the specification.

As traffic from each MAN device is secured independently, this scenario enables to leverage on Gateways in an infrastructure to be shared between different M2M applications/customers without any additional requirement.

Such implementations are not explicitly prevented by the current TC M2M Technical Specifications, which state that an M2M Node refers to one SCL without requiring it to be implemented locally. However further study would be necessary to address interoperability.

6.1.3 Gateway acting as security mediator between MAN and M2M Core

In the case of meshed wireless networks where some D' devices cannot directly reach their gateway because of limited range but can still reach one another (e.g. Zigbee® networks), there is generally no interest in securing the communication from each D' devices independently, however a scenario of interest consists in requiring the gateway to communicate its M2M session key previously bootstrapped with the M2M Service Layer, protected by the key Kg independently established over its MAN, to the D' devices as they connect under it. In this way, the D' devices only have to perform a single encryption, using the M2M Service Layer key, to protect the whole communication path, and no decryption/reencryption is required in the Gateway.

This model can be leveraged upon by M2M Service Providers in order to aggregate MANs belonging to distinct customers/applications, enabling D' devices of one owner to leverage on communication capabilities provided by intermediate D' devices or gateways belonging to other applications. This covers the possibility for a D' device to roam over another MAN of the same technology, as well as the possibility to fully merge different MANs of the same technology covering an overlapping geographic area, to multiply the coverage for end devices. For such scenarios, the gateways need to support addition and revocation of D' devices in their secured MAN.

Such services would be especially interesting in the case of mobile power-limited D' devices supporting only wireless technology of limited range. MAN aggregation also assumes that no confidentiality of D' devices communication is necessary, as all devices on the MAN can then access communications from other devices, even when they belong to distinct owner.

Annex A: Example of syntax for searchstring Tags

This example uses the searchString attribute of ETSI M2M resources to implement resource tagging and facilitate discovery and navigation within the resources used for ZigBee® interworking.

The searchString value is defined in [i.2] and is formatted as follows:

searchString= "Tag category/Tag value"

This example defines the following categories and values.

A.1 Category: ETSI.ObjectType

Reserved values:

ETSI.IP: Interworking proxy object

ETSI.AN_NWK: Network object

ETSI.AN_Node: Node object

ETSI.AN_APP: Application object

EXAMPLE: "ETSI.ObjectType/ PROFILE1.AN_NWK".

A.2 Category: ETSI.ObjectSemantic

The syntax of an object representation is usually indicated by its Content-Type, for instance application/xml. However multiple semantic conventions may leverage the same syntactic rules. In the present use case of interworking with control and sensor networks, an example of such semantic convention leveraging application/xml syntax is OASIS oBix (www.oasis-open.org) or ZigBee® Gateway Device REST binding. ASHRAE BACnet (ASHRAE 135 annex am) also leverages XML syntax (RFC 5023 [i.8] ATOM syntax with specific media types application/atomsvc+xml and application/atom+xml) to carry the Control System Modelling Language (CSML) semantic.

This example implements a REST design model which allows multiple representations of the objects manipulated through the ETSI M2M SCL. In order to complement the indication related to syntax carried by the Content type of the representation, it defines the ObjectSemantic tag category.

Reserved values:

OASIS.OBIX_1_1: OASIS oBix semantic conventions, version 1.1 [i.10].

ASHRAE.CSML_1_0: ASHRAE 135 annex am Control System Modelling Language (CSML) semantic conventions [i.11].

A.3 Category: ETSI.ApplicationProfile

Reserved for future use. The intent is to be able to facilitate search of specific devices e.g. "lamps". Nomenclatures have been created by ZigBee®, KNX™ and LONworks, and one is being worked on by BACnet. Future work would lead to a harmonized nomenclature which would use this category.

A.4 Category: ZigBee.ApplicationProfile

This tag facilitates search of devices implemented according to a given ZigBee® application profile. The value is the hexadecimal value of the application profile, represented as a 6 character string "ZigBee.ApplicationProfile/0x0104".

A.5 Category: ZigBee.DeviceIdentifier

This tag facilitates search of devices implemented according to a given ZigBee® device profile. The ZigBee.ApplicationProfile tag is mandatory if the ZigBee.DeviceIdentifier tag is used.

The value is the hexadecimal value of the DeviceIdentifier, represented as a 6 character string "ZigBee.DeviceIdentifier/0x0100".

A.6 Category: KNX.DptID

This tag facilitates search of KNX™ groups or KNX™ nodes matching a particular KNX™ DPT. The value is the numerical identifier of the DPT as specified and formatted by the KNX™ standard (e.g. "KNX.DptID/3.007").

A.7 Category: KNX.AreaID

This tag facilitates search of KNX™ nodes matching a particular area. The value is the numerical identifier of the area as specified and formatted by the KNX™ standard (e.g. "KNX.AreaID/1").

A.8 Category: KNX.LineID

This tag facilitates search of KNX™ nodes matching a particular line. The value is the numerical identifier of the line as specified and formatted by the KNX™ standard (e.g. "KNX.AreaID/1.15").

Annex B: Example of Application/XML syntax, oBix 1.1 semantic conventions

This profile targets the field of automation and sensor networks, for applications that seek to maximize the independence with the underlying area network hardware and technology.

B.1 Generic Area Network object representations

B.1.1 Generic Interworking Proxy Application resource content structure

oBix contracts:

An M2M InterworkingDescriptor contract is defined, which contains following mandatory:

- elements:"interworkingProxyID": An identifier for the interworking proxy
- "supportedTechnologies": A list of supported Access Network technologies defined as a triplet {AN standard, AN profile, AN physical layer}
- "networks": A list of network descriptor references to network objects

The M2M InterworkingDescriptor contract uses the http://uri.etsi.org/m2m/obix namespace.

```
<obj href="m2m:InterworkingDescriptor"
      xmlns="http://obix.org/ns/wsdl/1.1" xmlns:m2m="http://uri.etsi.org/m2m/obix">

  <str name="interworkingProxyID">
  <list name="supportedTechnologies" of="#ANTechnology">
    <obj href="#ANTechnology">
      <enum name="anStandard" range="#ANStandard">
        <list href="#ANStandard" is="obix:Range"/>
      </enum>
      <enum name="anProfile" range="#ANProfile">
        <list href="#ANProfile" is="obix:Range"/>
      </enum>
      <enum name="anPhysical" range="#ANPhysical">
        <list href="#ANPhysical" is="obix:Range"/>
      </enum>
    </obj>
  </list>
  <list name="networks" of="obix:ref m2m:NetworkDescriptor"/>

  standard <op> toward an interworking proxy can be added here
</obj>
```

Area network standards (`#ANStandard`), area network profiles (`#ANProfile`) and area network physical layers (`#ANPhysical`) are defined on a per protocol basis.

B.1.2 Generic Network resource content structure

oBix contracts:

An M2M NetworkDescriptor contract is defined, which contains following mandatory elements:

- "networkID":A network identifier
- "nodes": A list of node descriptor references to node objects on this network

The M2M NetworkDescriptor contract uses the http://uri.etsi.org/m2m/obix namespace.

```
<obj href="m2m:NetworkDescriptor" xmlns="http://obix.org/ns/wsdl/1.1"
xmlns:m2m="http://uri.etsi.org/m2m/obix">

  <str name="networkID"/>
  <list name="nodes" of="obix:ref m2m:NodeDescriptor"/>
  <op name="open" in="m2m:OpenInput" out="obix:nil"/>
  <op name="registerApplication" in="m2m:RegisterApplicationInput"
out="m2m:RegisterApplicationOutput"/>
  <op name="deregisterApplication" in="m2m:DeregisterApplicationInput" out="obix:nil"/>
</obj>
```

The contract defines the following operation:

- "open": When supported by the M2M area network, this operation permits or prohibits association of new devices with the Network. This operation defines the following IN and OUT parameters.

<table>
<tr><td>IN</td><td><pre><?xml version="1.0" encoding="UTF-8"?>
<obix:obj href="m2m:OpenInput"
    xmlns:obix="http://obix.org/ns/schema/1.1"
    xmlns:m2m="http://uri.etsi.org/m2m/obix">

    <obix:reltime name="duration"/>
    <obix:bool name="single"/>
    <obix:str name="expectedID"/>
    <obix:str name="replaceID"/>
</obix:obj></pre>
"duration": When the operation is invoked with a non-null duration, the IPU authorizes the association of new devices. This authorization is granted for the given duration, and is automatically canceled by the IPU when the duration expires.
A granted authorization can also be canceled by invoking the operator with a null duration (PT0S).
When the operation is invoked with a non-null duration, the following attributes can also be specified:
"single": when set to TRUE, the IPU should accept a single device and then immediately close the network. The IPU will close the network anyway if no device has joined before the open primitive timeout.
"expectedID": when associated to a node ID as relevant for the HAN technology, the IPU will only accept a new device that exactly matches the expected node ID.
"replaceID": when associated to a node ID as relevant for the HAN technology, the IPU will only accept a device that matches with identical interfaces as those of replaced node ID. The IPU will update all documents associated to the replaced node ID (e.g. changing the MAC address) to point the accepted node ID, and in general try to preserve the replaced device configuration for seamless replacement.</td></tr>
<tr><td>OUT</td><td>obix:nil</td></tr>
</table>

- "registerAppplication": When supported by the M2M area network, this operation registers an xA (GA, DA or NA) on the M2M area network to allow the xA to be visible on the M2M area network and to interact with other sensors connected to the M2M area network.

  The procedure by which the xA is visible on the M2M area network, and procedures by which native sensors on the M2M area network can interact with the xA, are implemented by the IPU in charge of the M2M are network and are out of scope of the present document.

IN

```
<?xml version="1.0" encoding="UTF-8"?>
<obix:obj href="m2m:RegisterApplicationInput"
    xmlns:obix="http://obix.org/ns/schema/1.1"
    xmlns:m2m="http://uri.etsi.org/m2m/obix">

    <obix:uri name="xA"/>
    <obix:str name="nodeID"/>
    <obix:str name="endpointID"/>
</obix:obj>
```

- "xA": The URI of the xA (GA, DA or NA) that registers the M2M Area Network. A xA that registers an M2M Area Network complies with the Application resource content structure, as described in this clause B, for the relevant HAN technology (e.g. a xA should comply with the ZigBee® Application resource content structure to register a ZigBee® Area Network).

If the xA does not comply with the relevant Application resource content structure, the registration is rejected.
The IPU in charge of the M2M Area Network uses this Application resource content structure to retrieve the description of the xA and to handle the registration on the M2M area network.

- "nodeID": When specified, the node ID as relevant for the HAN technology, on which the xA should be attached (this is typically a "virtual" node created by the IPU). If the xA cannot be attached to this node, the registration is rejected.

When not specified, the nodeID is selected by the IPU and returned in the response.

- "endpointID": When specified, the endpoint ID as relevant for the HAN technology, on which the xA should be attached. If the xA cannot be attached to this endpoint, the registration is rejected.

When not specified, the endpointID is selected by the IPU and returned in the response.

OUT

```
<?xml version="1.0" encoding="UTF-8"?>
<obix:obj href="m2m:RegisterApplicationOutput"
    xmlns:obix="http://obix.org/ns/schema/1.1"
    xmlns:m2m="http://uri.etsi.org/m2m/obix">
    <obix:str name="nodeID"/>
    <obix:str name="endpointID"/>
</obix:obj>
```

- "nodeID": When the registration procedure succeeds, the node ID as relevant for the HAN technology, on which the xA is attached.
- "endpointID": When the registration procedure succeeds, the endpoint ID as relevant for the HAN technology, on which the xA is attached.

- "deregisterAppplication": When supported by the M2M area network, this operation deregisters an xA (GA, DA or NA) on the M2M area network.

IN

```
<?xml version="1.0" encoding="UTF-8"?>
<obix:obj href="m2m:DeregisterApplicationInput"
    xmlns:obix="http://obix.org/ns/schema/1.1"
    xmlns:m2m="http://uri.etsi.org/m2m/obix">

    <obix:uri name="xA"/>
</obix:obj>
```

- "xA": The URI of the xA (GA, DA or NA) that deregisters the M2M Area Network.

OUT

```
obix:nil
```

B.1.3 Generic Device resource content structure

oBix contracts:

An M2M NodeDescriptor contract is defined, which contains the following mandatory elements:

- "nodeID": The node identifier
- "modelID": The model identifier

- "applications": A list of area network application descriptor references to such application objects hosted by this node

The M2M NodeDescriptor contract uses the http://uri.etsi.org/m2m/obix namespace.

```
<obj href="m2m:NodeDescriptor"
     xmlns="http://obix.org/ns/wsdl/1.1" xmlns:m2m="http://uri.etsi.org/m2m/obix">

  <str name="nodeID"/>
  <str name="modelID"/>
  <list name="applications" of="obix:ref m2m:ApplicationDescriptor"/>
  <op name="leave" in="obix:nil" out="obix:nil"/>
</obj>
```

The contract defines the following operation:

- "leave": When supported by the M2M area network, this operation instructs the IPU to dissociate the node for the HAN network. The IPU will remove the M2M representation of the device in the SCL and, according to the HAN technology, will force the device to leave the HAN network.

IN	obix:nil
OUT	obix:nil

B.1.4 Generic Application resource content structure

oBix contracts:

An M2M ApplicationDescriptor contract is defined, which contains following mandatory elements:

- "applicationID": the application identifier
- "applicationTypeID": the application type identifier
- "interfaces": A list of area network interface descriptor (or reference) to such interface objects implemented by this application

The M2M ApplicationDescriptor contract uses the http://uri.etsi.org/m2m/obix namespace.

```
<obj href="m2m:ApplicationDescriptor"
     xmlns="http://obix.org/ns/wsdl/1.1" xmlns:m2m="http://uri.etsi.org/m2m/obix">

  <str name="applicationID"/>
  <str name="applicationTypeID"/>
  <list name="Interfaces" of="obix:ref m2m:InterfaceDescriptor"/>

  Standard <op> toward an application can be added here
</obj>
```

B.1.5 GenericInterface resource content structure

An M2M InterfaceDescriptor contract is defined, which contains following mandatory elements:

- "interfaceID": the interface identifier
- "interfaceTypeID": the interface type identifier
- "points": Zero, one or several area network interface Point object (or reference) published by this interface
- "operations": Zero, one or several area network operation reference implemented by this interface
- "feeds": Zero, one or several area network feed reference of event objects published by this interface

- "sub-interfaces": A list of sub-interfaces, if any

The M2M InterfaceDescriptor contract uses the http://uri.etsi.org/m2m/obix namespace.

```
<obj href="m2m:InterfaceDescriptor"
     xmlns="http://obix.org/ns/wsdl/1.1" xmlns:m2m="http://uri.etsi.org/m2m/obix">

  <str name="InterfaceID"/>
  <str name="InterfaceTypeID"/>

  zero, one or several points (or point references)
  zero, one or several operation references
  zero, one or several feed references
  <list name="sub-interfaces" of="obix:InterfaceDescriptor/>

  Standard <op> toward an interface can be added here
</obj>
```

B.1.6 Generic Point resource content structure

oBix contracts:

A Point is a specific data field which contains a value in one of the primitive types, and optional qualifiers e.g. a measurement unit or status. It is a common concept used in most automation and fieldbus protocols. A generic M2M Point contract is defined, simply as an oBix point; so far it contains no additional mandatory elements.

The M2M Point contract uses the http://uri.etsi.org/m2m/obix namespace.

```
<obj href="m2m:Point" is="obix:Point"
     xmlns="http://obix.org/ns/wsdl/1.1" xmlns:m2m="http://uri.etsi.org/m2m/obix"
</obj>
```

B.2 ZigBee® Area Network object representations

B.2.1 Mapping of native ZigBee® primitive types to oBix types

ZigBee® class/type		oBIX type
Null		For future study
General data		obix:str (hexadecimal encoded)
Logical/Boolean		obix:bool
Bitmap		obix:str (hexadecimal encoded)
Unsigned integer		obix:int NOTE: A value greater than 2^63-1 is represented as a negative value.
Signed integer		obix:int
Enumeration		obix:enum (hexadecimal encoded)
Floating point		obix:real
String	Octet string	obix:str (hexadecimal encoded)
	Character string	obix:str (UTF8-encoding)
	Long octet string	obix:str (hexadecimal encoded)
	Long character string	obix:str (UTF8-encoding)
Ordered sequence		For future study
Collection		For future study
Time	Time of day	obix:reltime
	Date	obix:abstime
	UTC Time	obix:abstime

ZigBee® class/type		oBIX type
Identifier		obix:str (hexadecimal encoded)
Miscellaneous	IEEE address	obix:str (hexadecimal encoded)
	128-bit security -key	obix:str (hexadecimal encoded)

B.2.2 ZigBee® Interworking Proxy Application resource content structure

Constants:

The ZigBee® interworking gateway uses the following AN standard.

Area network standard	ANStandard extension
ZigBee® 1.0	`<obj name="ZigBee_1_0"/>`

The ZigBee® interworking gateway uses the following AN profile.

Area network profile	ANProfile extension
ZigBee® Home automation profile	`<obj name="ZigBee_HA"/>`
ZigBee® Smart Energy profile	`<obj name="ZigBee_SE1"/>`

The ZigBee® interworking gateway uses the following AN physical layer.

Area network profile	ANProfile extension
IEEE 802.15.4-2003 [i.3] (2. 4GHz)	`<obj name="IEEE_802_15_4_2003_2_4GHz”/>`

oBix contracts:

A ZigBee® InterworkingDescriptor contract is defined for interworking proxies supporting the ZigBee® standard. This contract overloads the M2M InterworkingDescriptor contract and contains no additional mandatory sub-elements, but may contain oBix operations referring to ZGD operations.

The ZigBee® InterworkingDescriptor contract uses the http://uri.etsi.org/m2m/zigbee/obix namespace.

```
<obj href="zigbee:InterworkingDescriptor" is="m2m:InterworkingDescriptor"
   xmlns="http://obix.org/ns/wsdl/1.1"
    xmlns:m2m="http://uri.etsi.org/m2m/obix"
    xmlns:zigbee="http://uri.etsi.org/m2m/zigbee/obix">

 <op> toward retargeted ZGD resources (e.g. /version, /ib, /request, /energy, /reset, /startup,
/networks...) can be added here
</obj>
```

NOTE: All XML elements belonging to the generic m2m:InterworkingDescriptor contract are not reproduced in this ZigBee® derived contract.

Representation example

```
GET /gsc/applications/ipu0/containers/descriptor/contentInstances/last/content

 <obj is="zigBee:InterworkingDescriptor">
  <str name=”interworkingProxyID” val="Text for correlation purpose"/>
  <list name="supportedTechnologies">
   <obj>
    <enum name="anPhysical" val="IEEE_802_15_4_2003_2_4GHz"/>
```

```
    <enum name="anStandard" val="ZigBee_1_0"/>
    <enum name="anProfile" val="ZigBee_HA"/>
   </obj>
  </list>

  <list name="networks" of="obix:ref m2m:NetworkDescriptor"/>
    <ref href="/gsc/applications/nwk0/containers/descriptor/contentInstances/last/content/"/>
  </list>
 </obj>
```

B.2.3 ZigBee® Network resource content structure

oBix contracts:

A ZigBee® NetworkDescriptor contract is defined for interworking proxies supporting the ZigBee® standard. This contract overloads the M2M NetworkDescriptor contract and contains the following additional mandatory sub-elements:

- "extendedPanID": The 802.15.4 extended PAN ID of the ZigBee® network represented.

The ZigBee® NetworkDescriptor may also contain oBix operations referring to ZGD operations.

The ZigBee® NetworkDescriptor contract uses the http://uri.etsi.org/m2m/zigbee/obix namespace.

```
<obj href="zigbee:NetworkDescriptor" is="m2m:NetworkDescriptor"
   xmlns="http://obix.org/ns/wsdl/1.1"
   xmlns:m2m="http://uri.etsi.org/m2m/obix"
   xmlns:zigbee="http://uri.etsi.org/m2m/zigbee/obix">

 <str name="extendedPanID"/>

 <op> toward retargeted ZGD resources (e.g. /ib, /callbacks, /aliases, /discovery, /wsnnodes...)
can be added here
 <op> toward retargeted ZDO resources (e.g. /zdoMgntPermitJoin...) can be added here
</obj>
```

Representation example

```
GET /gsc/applications/nwk0/containers/descriptor/contentInstances/last/content

 <obj is="zigbee:NetworkDescriptor">
  <str name="networkID" val="Text for correlation purpose"/>
  <str name="extendedPanID" val"0x685B3C34"/>

  <list name="nodes" of="obix:ref m2m:NodeDescriptor">
   <ref href="/gsc/applications/dev0/containers/descriptor/contentInstances/last/content"/>
   <ref href="/gsc/applications/dev1/containers/descriptor/contentInstances/last/content"/>
  </list>

  <ope name="zdoMgmtPermitJoin" in="zigbee:ZdoMgmtPermitJoinInput" out="obix:nil"
       href="/gsc/applications/nwk0/appCommands/cmd0"/>
 </obj>
```

B.2.4 ZigBee® Device resource content structure

oBix contracts:

A ZigBee® NodeDescriptor contract is defined for interworking proxies supporting the ZigBee® standard. This contract overloads the M2M NodeDescriptor contract and contains the following additional mandatory sub-elements:

- "ieeeAddress": the 802.15.4 64 bit address of the node

- "type": the ZigBee® device type, of values endpoint or router

It may also contain oBix operations referring to ZGD operations.

The ZigBee® NodeDescriptor contract uses the http://uri.etsi.org/m2m/zigbee/obix namespace.

```
<obj href="zigbee:NodeDescriptor" is="m2m:NodeDescriptor"
      xmlns="http://obix.org/ns/wsdl/1.1" xmlns:m2m="http://uri.etsi.org/m2m/obix"
      xmlns:zigbee="http://uri.etsi.org/m2m/zigbee/obix">

  <str name="ieeeAddress"/>
  <enum name="type" range="#NodeType">
    <list href="#NodeType" is="obix:Range">
      <obj name="endDevice"/>
      <obj name="router"/>
      <obj name="coordinator"/>
    </list>
  </enum>

  <op> toward retargeted ZGD resources can be added here
  Specific ZDO resources (e.g. "/zdoMgmtBind" /zdoMgmtLeave"...) can be added here
</obj>
```

Representation example

```
GET /gsc/applications/node0/containers/descriptor/contentInstances/last/content

 <obj is="zigbee:NodeDescriptor">
  <str name="nodeID" val="Text for correlation purpose"/>
  <str name="ieeeAddress" val="0x685B3C8812345678"/>
  <enum name="type" val="endDevice"/>

  <list name="applications" of="obix:ref m2m:ApplicationDescriptor">
   <ref href="/gsc/applications/app0/containers/descriptor/contentInstances/last/content/"/>
  </list>

  <ope name="zdoMgmtBind" in="obix:nil" out="zigbee:ZdoMgmtBindOutput"/
       href="/gsc/applications/node0/appCommands/cmd0"/>
  <ope name="zdoMgmtLeave" in="zigbee:ZdoMgmtLeaveInput" out="obix:nil"
       href="/gsc/applications/node0/appCommands/cmd1"/>
 </obj>
```

B.2.5 ZigBee® Application resource content structure

oBix contracts:

A ZigBee® AppplicationDescriptor contract is defined for interworking proxies supporting the ZigBee® standard. This contract overloads the M2M ApplicationDescriptor contract and contains the following additional mandatory sub-elements:

- "endpoint": the ZigBee® endpoint ID
- "applicationProfileID": the ZigBee® application profile ID
- "applicationDeviceID": the ZigBee® application device ID
- "applicationDeviceVersion": the ZigBee® application device version

It may also contain oBix operations referring to ZGD operations.

The ZigBee® ApplicationDescriptor contract uses the http://uri.etsi.org/m2m/zigbee/obix namespace.

```
<obj href="zigbee:ApplicationDescriptor" is="m2m:ApplicationDescriptor"
      xmlns="http://obix.org/ns/wsdl/1.1" xmlns:m2m="http://uri.etsi.org/m2m/obix"
      xmlns:zigbee="http://uri.etsi.org/m2m/zigbee/obix">

  <int name="endpoint"/>
  <int name="applicationProfileID"/>
  <int name="applicationDeviceID"/>
  <int name="applicationDeviceVersion"/>

  <op> toward retargeted ZGD resources (e.g. "/ SendZDPCommand"...) can be added here
  <op> toward retargeted ZDO resources (e.g. "/zdoBind", "/zdoUnbind"...) can be added here
 </obj>
```

B.2.6 ZigBee® cluster (Interface) resource content structure

A ZigBee® InterfaceDescriptor contract is defined for interworking proxies supporting the ZigBee® standard. This contract overloads the M2M InterfaceDescriptor contract and contains the following additional mandatory sub-elements:

- "clusterID": the ZigBee® cluster identifier, value contains hexadecimal cluster ID represented as string
- "clusterType": server or client cluster, as defined by the ZigBee® cluster library

It may also contain oBix operations referring to ZGD operations.

The ZigBee® InterfaceDescriptor contract uses the http://uri.etsi.org/m2m/zigbee/obix namespace.

```
<obj href="zigbee:InterfaceDescriptor" is="m2m:InterfaceDescriptor"
      xmlns="http://obix.org/ns/wsdl/1.1" xmlns:m2m="http://uri.etsi.org/m2m/obix"
      xmlns:zigbee="http://uri.etsi.org/m2m/zigbee/obix">

  <str name="clusterID"/>
  <enum name="clusterType" range="#ClusterType">
    <list href="#ClusterType" is="obix:Range">
        <obj name="server"/>
        <obj name="client"/>
    </list>
  </enum>

  <op> toward retargeted ZDO resources (e.g. /zdoBind, /zdoUnbind, /zclXXX...) can be added here
</obj>
```

B.2.7 ZigBee® Point resource content structure

A ZigBee® Point contract is defined for interworking proxies supporting the ZigBee® standard. This contract overloads the M2M Point contract and contains the following optional sub-elements:

- "nativeAttributes": the list of zigBee® native attributes (or reference). The rationale for having a list is that in some cases, typically for measurement points, a single "Point", which includes e.g. a unit facet, maps to multiple native ZigBee® attributes, as ZigBee® clusters model values and units as separate attributes.

The ZigBee® Point contract uses the http://uri.etsi.org/m2m/zigbee/obix namespace.

```
<obj href="zigbee:Point" is="m2m:Point"
      xmlns="http://obix.org/ns/wsdl/1.1" xmlns:m2m="http://uri.etsi.org/m2m/obix">
      xmlns:zigbee="http://uri.etsi.org/m2m/zigbee/obix">

  <list name="nativeAttributes" of="obix:ref obix:obj"/>
</obj>
```

B.2.8 ZigBee® Application representation examples

Representation example (an on/off light)

This sample shows an application where mirroring and retargeting have been mixed. In this sample, the GIP has been configured to report the on/off light state of this application in a M2M container.

```
GET /gsc/applications/app0/containers/descriptor/contentInstances/last/content

 <obj is="zigBee:ApplicationDescriptor">
  <str name="applicationID" val="Text for correlation purpose"/>
  <str name="applicationProfileID" val="0x104"/>
  <str name="applicationDeviceID" val="0x0103"/>
  <int name="endpoint" val="1"/>

  <list name="interfaces" of="m2m:InterfaceDescriptor">
   <obj is="zigbee:OnOffLightCluster zigbee:InterfaceDescriptor">
    <str name="interfaceID" val="Text for correlation purpose"/>
    <str name="clusterID" val="0x0006"/>
    <enum name="clusterType" val="server"/>

    <ref name="onOffState"
 href="/gsc/application/app0/containers/point0/contentInstances/last/content is="zigbee:Point"/>

   Other attribute can be added here

    <op name="zclToggle" in="obix:nil" out="obix:nil"
        href="/gsc/applications/app0/appCommands/cmd0"/>
    <op name="zclOn" in="obix:nil" out="obix:nil"
        href="/gsc/applications/app0/appCommands/cmd1"/>
    <op name="zclOff" in="obix:nil" out="obix:nil"
        href="/gsc/applications/app0/appCommands/cmd2"/>
    </obj>
   </list>
 </obj>
```

```
GET /gsc/application/app0/containers/point0/contentInstances/last/content

 <bool name="zclOnOffState" val="off" is="zigbee:Point"/>
```

Representation example (an on/off switch)

This sample shows an application where mirroring and retargeting have been mixed. In this sample, the GIP has been configured to report the on/off switch event of this application in a M2M container.

```
GET /gsc/applications/app0/containers/descriptor/contentInstances/last/content

 <obj is="zigBee:ApplicationDescriptor">
  <str name="applicationID" val="Text for correlation purpose"/>
  <str name="applicationProfileID" val="0x104"/>
  <str name="applicationDeviceID" val="0x0103"/>
  <int name="endpoint" val="1"/>

  <list name="interfaces of="m2m:InterfaceDescriptor">
   <obj is="zigbee:OnOffSwitchCluster zigbee:InterfaceDescriptor">
    <str name="interfaceID" val="Text for correlation purpose"/>
    <str name="clusterID" val="0x0006"/>
    <enum name="clusterType" val="client"/>

    <feed name="onOffEvent" of="m2m:ZigBeeOnOffEvent"
href="/gsc/application/app1/containers/feed0/contentInstances/subscription"/>

    <op name="zdoBind" in="zigbee:ZdoBindInput" out="obix:nil"
        href="/gsc/applications/app1/appCommands/cmd0"/>
    <op name="zdoUnbind" in="zigbee:ZdoUnbindInput" out="obix:nil"
  href="/gsc/applications/app1/appCommands/cmd1"/>
```

```
  </obj>
 </list>
</obj>
```

B.3 wM-Bus Area Network object representations

B.3.1 Mapping of native wM-Bus primitive types and units to oBix types and units

wM-Bus data type	oBIX type
8 Bit Integer/Binary	obix:int
16 Bit Integer/Binary	obix:int
24 Bit Integer/Binary	obix:int
32 Bit Integer/Binary	obix:int
32 Bit Real	obix:real
48 Bit Integer/Binary	obix:int
64 Bit Integer/Binary	obix:int

wM-Bus unit	oBIX unit
Energy (Wh)	obix:units/watt_hour obix:units/kilowatt_hour obix:units/megawatt_hour
Energy (J)	obix:units/joule obix:units/kilojoule obix:units/megajoule
Volume (m3)	obix:units/milliliter obix:units/liter obix:units/cubic_millimeter obix:units/cubic_centimeter obix:units/cubic_meter
Mass (kg)	obix:units/milligram obix:units/gram obix:units/kilogram obix:units/metric_ton
Power (W)	obix:units/milliwatt obix:units/watt obix:units/kilowatt obix:units/megawatt units/gigawatt
Power (J/h)	*not available*
Volume Flow (m3/h, m3/min, m3/s)	obix:units/milliliters_per_second obix:units/liters_per_second obix:units/liters_per_minute obix:units/liters_per_hour obix:units/cubic_meters_per_hour obix:units/cubic_meters_per_minute obix:units/cubic_meters_per_second
Mass flow (kg/h)	obix:units/kilograms_per_second obix:units/kilograms_per_minute obix:units/kilograms_per_hour obix:units/grams_per_second obix:units/grams_per_minute
Temperature (K)	obix:units/kelvin
Temperature (°C)	obix:units/celsius
Pressure (bar)	obix:units/millibar obix:units/bar

B.3.2 wM-Bus Interworking Proxy Application resource content structure

Constants:

The wM-Bus interworking gateway uses the following AN standard.

Area network standard	ANStandard extension
EN 13757 [i.4] 2004	`<obj name="EN_13757_2004"/>`

The wM-Bus interworking gateway uses the following AN physical layer.

Area network profile	ANProfile extension
TBD	`<obj name="TBD"/>`

oBix contracts:

A WM-Bus InterworkingDescriptor contract is defined for interworking proxies supporting the WM-Bus standard. This contract overloads the M2M InterworkingDescriptor contract and contains no additional mandatory sub-elements.

The WM-Bus InterworkingDescriptor contract uses the http://uri.etsi.org/m2m/wmbus/obix namespace.

```
<obj href="wmbus:InterworkingDescriptor" is="m2m:InterworkingDescriptor"
     xmlns="http://obix.org/ns/wsdl/1.1"         xmlns:m2m="http://uri.etsi.org/m2m/obix"
     xmlns:wmbus="http://uri.etsi.org/m2m/wmbus/obix">
</obj>
```

Representation example

```
GET /gsc/applications/ipu0/containers/descriptor/contentInstances/last/content

 <obj is="wmbus:InterworkingDescriptor">
  <str name="interworkingProxyID" val="Text for correlation purpose"/>
  <list name="supportedTechnologies">
   <obj>
    <enum name="anStandard" val="EN_13757_2004"/>
   </obj>
  </list>

  <list name="networks" of="obix:ref m2m:NetworkDescriptor"/>
    <ref href="/gsc/applications/nwk0/containers/descriptor/contentInstances/last/content/"/>
  </list>
 </obj>
```

B.3.3 WM-Bus Network resource content structure

oBix contracts:

A WM-Bus NetworkDescriptor contract is defined for interworking proxies supporting the WM-Bus standard. This contract overloads the M2M NetworkDescriptor contract and contains the following additional mandatory sub-elements:

- "rfcChannel": The RF channel (between 1 and 12) used by the wM-Bus master
- "rfPower": The RF transmission power (-5;0;+5;7;10) used by the wM-Bus master
- "operatingModel": The operating model (R2 ; S1 ; S1-m ; S2 ; T1 ; T2) used by the wM-Bus master
- "manufacturerID": Manufacturer ID of the master (provided as a 3 digits string)

- "identificationNumber": Identification number of the master (provided as a 4 bytes hexadecimal code)
- "version": Version of the master (between 0 and 255)

The WM-Bus NetworkDescriptor contract uses the http://uri.etsi.org/m2m/wmbus/obix namespace.

```
<obj href="wmbus:NetworkDescriptor" is="m2m:NetworkDescriptor"
     xmlns="http://obix.org/ns/wsdl/1.1" xmlns:m2m="http://uri.etsi.org/m2m/obix"
     xmlns:wmbus="http://uri.etsi.org/m2m/wmbus/obix">

  <int name="rfcChannel"/>
  <str name="rfPower"/>
  <str name="operatingModel"/>
  <str name="manufacturerID"/>
  <str name="identificationNumber"/>
  <int name="version"/>

  <op> toward retargeted wM-Bus resources (e.g. /wJoin...) can be added here
</obj>
```

Representation example

```
GET /gsc/applications/nwk0/containers/descriptor/contentInstances/last/content

 <obj is="wmbus:NetworkDescriptor">
  <str name="networkID" val="Text for correlation purpose"/>
  <int name="rfcChannel" val="1"/>
  <str name="rfPower" val="+5"/>
  <str name="operatingModel" val="R2"/>
  <str name="manufacturerID" val="AMB"/>
  <str name="identificationNumber" val="0x12131415"/>
  <int name="version" val="2"/>
  <list name="nodes" of="obix:ref m2m:NodeDescriptor">
   <ref href="/gsc/applications/dev0/containers/descriptor/contentInstances/last/content"/>
  </list>

  <ope name="wmbusJoin" in="wmbus:JoinInput" out="obix:nil"
       href="/gsc/applications/nwk0/appCommands/cmd0"/>
 </obj>
```

B.3.4 WM-Bus Device resource content structure

oBix contracts:

A WM-Bus NodeDescriptor contract is defined for interworking proxies supporting the WM-Bus standard. This contract overloads the M2M NodeDescriptor contract and contains the following additional mandatory sub-elements:

- "type": wM-Bus type (e.g. Oil, Gas, Water, Other, etc.)
- "aesEnable": Enable or disable the AES 128 bits encryption
- "aesKey": Value of AES 128 bits encryption (only mandatory when the AES is enable)
- "manufacturerID": Manufacturer ID of the device
- "identificationNumber": Identification number of the device
- "version": Version of the device
- "type": Type of the device (provided as a 1 byte hexadecimal code)
- "status": Last status byte received from the device. The status byte in a bitmap (see EN 13757-3 [i.5], section 5.9)

The WM-Bus NodeDescriptor contract uses the http://uri.etsi.org/m2m/wmbus/obix namespace.

```
<obj href="wmbus:NodeDescriptor" is="m2m:NodeDescriptor"
      xmlns="http://obix.org/ns/wsdl/1.1" xmlns:m2m="http://uri.etsi.org/m2m/obix">
      xmlns:wmbus="http://uri.etsi.org/m2m/wmbus/obix">

  <int name="type"/>
  <bool name="aesEnable"/>
  <str name="aesKey"/>
  <str name="manufacturerID"/>
  <str name="identificationNumber"/>
  <int name="version"/>
  <str name="type"/>
  <str name="status"/>

  <op> toward retargeted wM-Bus resources (e.g. /wReset...) can be added here
</obj>
```

Representation example

```
GET /gsc/applications/node0/containers/descriptor/contentInstances/last/content

 <obj is="wmbus:NodeDescriptor">
  <str name="nodeID" val="Text for correlation purpose"/>
  <int name="type" value="2"/>
  <bool name="aesEnable" value="true"/>
  <str name="aesKey" value="0xabcdef0123456789"/>
  <str name="manufacturerID" value="AMB"/>
  <str name="identificationNumber" value="0x01020304"/>
  <int name="version" value="1"/>
  <int name="type" value="0x01"/>
  <int name="status" value="0x01"/>

  <list name="applications" of="obix:ref m2m:ApplicationDescriptor">
   <ref href="/gsc/applications/app0/containers/descriptor/contentInstances/last/content/"/>
  </list>

  <ope name="wmReset" in="wmbus:ResetInput" out="obix:nil"/
       href="/gsc/applications/node0/appCommands/cmd0"/>
 </obj>
```

B.3.5 WM-Bus Application resource content structure

oBix contracts:

A WM-Bus AppplicationDescriptor contract is defined for interworking proxies supporting the WM-Bus standard. This contract overloads the M2M ApplicationDescriptor contract and contains no additional mandatory sub-elements.

The WM-Bus ApplicationDescriptor contract uses the http://uri.etsi.org/m2m/wmbus/obix namespace.

```
<obj href="wmbus:AppDescriptor" is="m2m:ApplicationDescriptor"
      xmlns="http://obix.org/ns/wsdl/1.1" xmlns:m2m="http://uri.etsi.org/m2m/obix">
      xmlns:wmbus="http://uri.etsi.org/m2m/wmbus/obix">
 </obj>
```

B.3.6 WM-Bus profile (Interface) resource content structure

A WM-Bus InterfaceDescriptor contract is defined for interworking proxies supporting the WM-Bus standard. This contract overloads the M2M InterfaceDescriptor contract and contains no additional mandatory sub-elements

A WM-Bus application has always only one interface.

The WM-Bus InterfaceDescriptor contract uses the http://uri.etsi.org/m2m/wmbus/obix namespace.

```
<obj href="wmbus:InterfaceDescriptor" is="m2m:InterfaceDescriptor"
      xmlns="http://obix.org/ns/wsdl/1.1" xmlns:m2m="http://uri.etsi.org/m2m/obix">
      xmlns:wM-Bus="http://uri.etsi.org/m2m/wmbus/obix">
</obj>
```

B.3.7 WM-Bus Point resource content structure

A WM-Bus Point contract is defined for interworking proxies supporting the WM-Bus standard. This contract overloads the M2M Point contract and contains the following optional sub-elements:

- "dif": wM-Bus native Data Information Field (DIF). The DIF contents the Function Field and the Data Field. The Function Field gives the type of data (e.g. Instantaneous, Minimum, etc.). The data field shows how the data is interpreted (e.g. 8 Bit Integer/Binary, 16 Bit Integer/Binary, etc.)
- "vif": wM-Bus Value Information Block (VIF). The VIF contents the Unit and multiplier Field. The Unit and multiplier Filed gives the type of the data (e.g. Energy, Power, etc.) and the coding range (e.g. kwh, mwh, etc.)
- "data": wM-Bus native DATA value (hexadecimal encoding)

The WM-Bus Point contract uses the http://uri.etsi.org/m2m/wmbus/obix namespace.

```
<obj href="wmbus:Point" is="m2m:Point"
      xmlns="http://obix.org/ns/wsdl/1.1" xmlns:m2m="http://uri.etsi.org/m2m/obix">
      xmlns:wM-Bus="http://uri.etsi.org/m2m/wmbus/obix">

  <int name="dif"/>
  <int name="vif"/>
  <str name="data"/>
</obj>
```

B.3.8 WM-Bus Application representation examples

Representation example (an electric meter)

This sample shows an application where mirroring and retargeting have been mixed. In this sample, the GIP has been configured to report the current summation (kwh) of this application in a M2M container.

```
GET /gsc/applications/app0/containers/descriptor/contentInstances/last/content

 <obj is="wmbus:ApplicationDescriptor">
  <str name="applicationID" val="Text for correlation purpose"/>

  <list name="interfaces" of="m2m:InterfaceDescriptor">
   <obj is="wmbus:Meter wmbus:InterfaceDescriptor">
    <str name="interfaceID" val="Text for correlation purpose"/>

    <ref name="indexSummation"
href="/gsc/application/app0/containers/point0/contentInstances/last/content is="wmbus:Point"/>
    </obj>
   </list>
 </obj>
```

```
GET /gsc/application/app0/containers/point0/contentInstances/last/content

 <real name="indexSummation" val="7" unit="obix:units/kilowatt_hour" is="wmbus:Point"/>
```

B.4 KNX™ Area Network object representations

B.4.1 Mapping of native KNX™ data point types to oBix types and units

The following table defines the mapping between KNX™ data point types (DPT) and oBIX data types. For each KNX™ DPT, the mapping defines:

- The abstract oBIX data type and the associated oBIX unit (when applicable). When available this mapping ensures that an application with general understanding of oBix, and no specific understanding of KNX™, may interact with KNX™ nodes and understand the information published over the KNX™ network.
- The oBIX data type of all native KNX™ fields. This ensures that no native DPT information is lost in the REST representation.

For instance, KNX™ defines for DPT 2.001 (DPT_Switch_Control) a "c" field (1 bit) and a "v" field (1 bit). According to this definition, DPT 2.001 is mapped to the following oBix contract, where <enum> is the oBix abstraction and where <bool> are associated to native KNX™ fields:

```
<enum range="knx:ranges/dpt2_001" xmlns="http://obix.org/ns/schema/1.1">
 <bool name="v"/>
 <bool name="c"/>
</enum>
```

<table>
<tr><th>KNX™ DPT</th><th>oBIX contract</th></tr>
<tr><td>DPT 1.xxx</td><td><pre><bool>
   <bool name="b"/>
</bool></pre></td></tr>
<tr><td>DPT 2.xxx</td><td><pre><enum range="range">
   <bool name="v"/>
   <bool name="c"/>
</enum></pre>
Ranges:
<pre> <list href="knx:ranges/dpt2_001" is="obix:range">
  <int name="noControl"/>
  <int name="controlOff"/>
  <int name="controlOn"/>
 </list>
 <list href="knx:ranges/dpt2_002" is="obix:range">
  <int name="noControl"/>
  <int name="controlFalse"/>
 <int name="controlTrue"/>
 </list>
 <list href="knx:ranges/dpt2_003" is="obix:range">
  <int name="noControl"/>
  <int name="controlDisable"/>
  <int name="controlEnable"/>
 </list>
 <list href="knx:ranges/dpt2_004" is="obix:range">
  <int name="noControl"/>
  <int name="controlNoRamp"/>
  <int name="controlRamp"/>
 </list>
 <list href="knx:ranges/dpt2_005" is="obix:range">
  <int name="noControl"/>
  <int name="controlNoAlarm"/>
  <int name="controlAlarm"/>
 </list>
 <list href="knx:ranges/dpt2_006" is="obix:range">
  <int name="noControl"/>
  <int name="controlLow"/>
  <int name="controlHigh"/>
 </list>
 <list href="knx:ranges/dpt2_007" is="obix:range">
  <int name="noControl"/></pre></td></tr>
</table>

<table>
<tr><th>KNX™ DPT</th><th>oBIX contract</th></tr>
<tr><td></td><td><pre> <int name="controlDecrease"/>
  <int name="controlIncrease"/>
 </list>
 <list href="knx:ranges/dpt2_008" is="obix:range">
  <int name="noControl"/>
  <int name="controlUp"/>
  <int name="controlDown"/>
 </list>
 <list href="knx:ranges/dpt2_009" is="obix:range">
  <int name="noControl"/>
  <int name="controlOpen"/>
  <int name="controlClose"/>
 </list>
 <list href="knx:ranges/dpt2_010" is="obix:range">
  <int name="noControl"/>
  <int name="controlStop"/>
  <int name="controlStart"/>
 </list>
 <list href="knx:ranges/dpt2_011" is="obix:range">
  <int name="noControl"/>
  <int name="controlInactive"/>
  <int name="controlActive"/>
 </list>
 <list href="knx:ranges/dpt2_012" is="obix:range">
  <int name="noControl"/>
  <int name="controlNotInverted"/>
  <int name="controlInverted"/>
 </list></pre></td></tr>
<tr><td>DPT 3.xxx</td><td><pre><real unit="unit">
  <bool name="b"/>
  <int name="StepCode"/>
</real></pre>
Units:
<table>
<tr><td>3.007</td><td>obix:units/percent</td></tr>
<tr><td>3.007</td><td>obix:units/percent</td></tr>
<tr><td>3.008</td><td>obix:units/percent</td></tr>
</table></td></tr>
<tr><td>DPT 4.xxx</td><td><pre><str/></pre></td></tr>
<tr><td>DPT 5.xxx</td><td><pre><real unit="unit">
  <int name="UnsignedValue"/>
</real></pre>
Units:
<table>
<tr><td>5.001</td><td>obix:units/percent</td></tr>
<tr><td>5.003</td><td>obix:units/degrees_angular</td></tr>
<tr><td>5.004</td><td>obix:units/percent</td></tr>
</table></td></tr>
<tr><td>DPT 6.xxx</td><td><pre><int unit="unit">
  <int name="RelSignedValue" />
</int></pre>
Units:
<table>
<tr><td>6.001</td><td>obix:units/percent</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>KNX™ DPT</th><th>oBIX contract</th></tr>
<tr><td>DPT 6.020</td><td><obj>
  <bool name="a"/>
  <bool name="b"/>
  <bool name="c/>
  <bool name="d"/>
  <bool name="e"/>
  <int name="mode"/>
</obj></td></tr>
<tr><td>DPT 7.xxx</td><td><int unit="unit">
  <int name="UnsignedValue"/>
</int>

Units:
<table><tr><td>7.012</td><td>obix:units/milliampere</td></tr></table></td></tr>
<tr><td>DPT 7. 002,
DPT 7. 003,
DPT 7. 004,
DPT 7. 005,
DPT 7. 006,
DPT 7. 007</td><td><reltime unit="unit">
  <int name="TimePeriod"/>
</reltime>

Units:
<table><tr><td>7.002</td><td>obix:units/second</td></tr><tr><td>7.003</td><td>obix:units/second</td></tr><tr><td>7.004</td><td>obix:units/second</td></tr><tr><td>7.005</td><td>obix:units/second</td></tr><tr><td>7.006</td><td>obix:units/second</td></tr><tr><td>7.007</td><td>obix:units/second</td></tr></table></td></tr>
<tr><td>DPT 8.001,
DPT 8.010,
DPT 8.011</td><td><int unit="unit">
  <int name="SignedValue"/>
</int>

Units:
<table><tr><td>8.001</td><td>obix:units/null</td></tr><tr><td>8.010</td><td>obix:units/percent</td></tr><tr><td>8.011</td><td>obix:units/degrees_angular</td></tr></table></td></tr>
<tr><td>DPT 8.002,
DPT 8.003,
DPT 8.004,
DPT 8.005,
DPT 8.006,
DPT 8.00 7</td><td><reltime unit="unit">
  <int name="DeltaTime"/>
</reltime>

Units:
<table><tr><td>8.002</td><td>obix:units/second</td></tr><tr><td>8.003</td><td>obix:units/second</td></tr><tr><td>8.004</td><td>obix:units/second</td></tr><tr><td>8.005</td><td>obix:units/second</td></tr><tr><td>8.006</td><td>obix:units/second</td></tr><tr><td>8.007</td><td>obix:units/second</td></tr></table></td></tr>
</table>

<table>
<tr><th>KNX™ DPT</th><th>oBIX contract</th></tr>
<tr><td></td><td></td></tr>
<tr><td>DPT 9.xxx</td><td><real unit="unit">
  <real name="FloatValue"/>
</real>

Units:
<table>
<tr><td>9.001</td><td>unit="obix:units/celsius_degrees"</td></tr>
<tr><td>9.002</td><td>unit="obix:units/kelvin_degrees"</td></tr>
<tr><td>9.003</td><td>unit="obix:units/degrees_kelvin_per_hour"</td></tr>
<tr><td>9.004</td><td>unit="obix:units/lux"</td></tr>
<tr><td>9.005</td><td>unit="obix:units/meters_per_second"</td></tr>
<tr><td>9.006</td><td>unit="obix:units/pascal"</td></tr>
<tr><td>9.007</td><td>unit="obix:units/percent_relative_humidity"</td></tr>
<tr><td>9.008</td><td>unit="obix:units/parts_per_million"</td></tr>
<tr><td>9.010</td><td>unit="obix:units/second"</td></tr>
<tr><td>9.011</td><td>unit="obix:units/millisecond"</td></tr>
<tr><td>9.020</td><td>unit="obix:units/millivolt"</td></tr>
<tr><td>9.021</td><td>unit="obix:units/milliampere"</td></tr>
<tr><td>9.022</td><td>unit="obix:units/watts_per_square_meter"</td></tr>
<tr><td>9.023</td><td>unit="obix:units/x-kelvin_per_percent"</td></tr>
<tr><td>9.024</td><td>unit="obix:units/kilowatt"</td></tr>
<tr><td>9.025</td><td>unit="obix:units/liters_per_hour"</td></tr>
</table>
</td></tr>
<tr><td>DPT 10.010</td><td><reltime>
  <int name="Day"/>
  <int name="Hour"/>
  <int name="Minutes"/>
  <int name="Seconds"/>
</reltime></td></tr>
<tr><td>DPT 11.xxx</td><td><date>
  <int name="Year"/>
  <int name="Month"/>
  <int name="Day"/>
</date></td></tr>
<tr><td>DPT 12.xxx</td><td><int>
  <int name="UnsignedValue"/>
</int></td></tr>
<tr><td>DPT 13.001</td><td><int unit="unit">
  <int name="SignedValue"/>
</int>

Units:
<table>
<tr><td>13.100</td><td>obix:units/second</td></tr>
</table>
</td></tr>
<tr><td>DPT 13.100</td><td><reltime>
  <int name="SignedValue"/>
</reltime></td></tr>
</table>

KNX™ DPT	oBIX contract
DPT 14.xxx	(see below)

```
<real unit="unit">
  <real name="FloatValue"/>
</real>
```

Units:

14.000	obix:units/meters_per_second_squared
14.001	obix:units/radians_per_second_squared
14.002	obix:units/x-joule_per_mol
14.003	obix:units/x-per_second
14.004	obix:units/x-mol
14.006	obix:units/radian
14.007	obix:units/degrees_angular
14.008	obix:units/joule_second
14.009	obix:units/x-radians_per_second
14.010	obix:units/square_meter
14.011	obix:units/farad
14.012	obix:units/x-coulomb_square_meter
14.013	obix:units/x-coulomb_cubic_meter
14.014	obix:units/square_meters_per_newton
14.015	obix:units/x-per_ohm
14.016	obix:units/siemens_per_meter
14.017	obix:units/kilograms_per_cubic_meter
14.018	obix:units/coulomb
14.019	obix:units/amperes
14.020	obix:units/amperes_per_square_meter
14.021	obix:units/x-coulomb_meter
14.022	obix:units/x-coulomb_per_square_meter
14.023	obix:units/volts_per_meter
14.024	obix:units/x-c
14.025	obix:units/x-coulomb_per_square_meter
14.026	obix:units/x-coulomb_per_square_meter
14.027	obix:units/volt
14.028	obix:units/volt
14.029	obix:units/x-joule_per_tesla
14.030	obix:units/volt
14.031	obix:units/joule
14.032	obix:units/newton
14.033	obix:units/hertz
14.034	obix:units/x-radians_per_second

KNX™ DPT	oBIX contract	
	14.035	obix:units/x-joule_per_kelvin
	14.036	obix:units/watt
	14.037	obix:units/joule
	14.038	obix:units/ohm
	14.039	obix:units/meter
	14.040	obix:units/joule
	14.041	obix:units/candelas_per_square_meter
	14.042	obix:units/lumen
	14.043	obix:units/illuminance
	14.044	obix:units/amperes_per_meter
	14.045	obix:units/weber
	14.046	obix:units/tesla
	14.047	obix:units/ampere_square_meter
	14.048	obix:units/tesla
	14.049	obix:units/amperes_per_meter
	14.050	obix:units/ampere
	14.051	obix:units/kilogram
	14.052	obix:units/kilograms_per_second
	14.053	obix:units/x-newton_per_second
	14.054	obix:units/radian
	14.055	obix:units/degrees_angular
	14.056	obix:units/watt
	14.057	obix:units/power_factor
	14.058	obix:units/pascal
	14.059	obix:units/ohm
	14.060	obix:units/ohm
	14.061	obix:units/ohm_meter
	14.062	obix:units/henry
	14.063	obix:units/steradian
	14.064	obix:units/watts_per_square_meter
	14.065	obix:units/meters_per_second
	14.066	obix:units/pascal
	14.067	obix:units/newtons_per_meter
	14.068	obix:units/celsius_degrees
	14.069	obix:units/kelvin_degrees
	14.070	obix:units/temperature_differential
	14.071	obix:units/x-joule_per_kelvin

<table>
<tr><th>KNX™ DPT</th><th>oBIX contract</th></tr>
<tr><td></td><td><table>
<tr><td>14.072</td><td>obix:units/watts_per_meter_degree_kelvin</td></tr>
<tr><td>14.073</td><td>obix:units/volts_per_degree_kelvin</td></tr>
<tr><td>14.074</td><td>obix:units/second</td></tr>
<tr><td>14.075</td><td>obix:units/newton_meter</td></tr>
<tr><td>14.076</td><td>obix:units/cubic_meter</td></tr>
<tr><td>14.077</td><td>obix:units/cubic_meters_per_second</td></tr>
<tr><td>14.078</td><td>obix:units/newton</td></tr>
<tr><td>14.079</td><td>obix:units/joule</td></tr>
</table></td></tr>
<tr><td>DPT 15.xxx</td><td><obj>
  <bool name="c"/>
  <bool name="d" />
  <bool name="p" />
  <bool name="e" />
  <int name="d1" />
  <int name="d2" />
  <int name="d3"/>
  <int name="d4" />
  <int name="d5" />
  <int name="d6" />
  <int name="index" />
</obj></td></tr>
<tr><td>DPT 16.xxx</td><td><str/></td></tr>
<tr><td>DPT 17.xxx</td><td><int>
  <int name="SceneNumber"/>
</int></td></tr>
<tr><td>DPT 18.xxx</td><td><obj>
  <bool name="c"/>
  <int name="SceneNumber"/>
</obj></td></tr>
<tr><td>DPT 19.xxx</td><td><abstime>
  <int name="Year"/>
  <int name="Month"/>
  <int name="DayOfMonth"/>
  <int name="DayOfWeek"/>
  <int name="HourOfDay"/>
  <int name="Minutes"/>
  <int name="Seconds"/>
  <bool name="F"/>
  <bool name="WD"/>
  <bool name="NWD"/>
  <bool name="NY"/>
  <bool name="ND"/>
  <bool name="NDOW"/>
  <bool name="NT"/>
  <bool name="SUTI"/>
  <bool name="CLQ"/>
</abstime></td></tr>
<tr><td>DPT 20.xxx</td><td><enum range="range">
  <int name="field1"/>
</enum>

Ranges:
<list href="knx:ranges/dpt20_001" is="obix:range">
  <int name="autonomous"/>
  <int name="slave"/>
  <int name="master"/>
</list>
<list href="knx:ranges/dpt20_002" is="obix:range">
  <int name="buildingInUse"/>
  <int name="buildingNotUsed"/>
  <int name="buildingProtection"/>
</list>
<list href="knx:ranges/dpt20_003" is="obix:range">
  <int name="occupied"/>
  <int name="standby"/></td></tr>
</table>

KNX™ DPT	oBIX contract
	`` <int name="notOccupied"/> </list> <list href="knx:ranges/dpt20_004" is="obix:range"> <int name="high"/> <int name="medium"/> <int name="low"/> <int name="void"/> </list> <list href="knx:ranges/dpt20_005" is="obix:range"> <int name="normal"/> <int name="presenceSimulation"/> <int name="nightRound"/> </list> <list href="knx:ranges/dpt20_006" is="obix:range"> <int name="noFault"/> <int name="systemAndFunctionsOfCommonInterest"/> <int name="HVAC_GeneralFBs"/> <int name="HVAC_HotWaterHeating"/> <int name="HVAC_DirectElectricalHeating"/> <int name="HVAC_TerminalUnits"/> <int name="HVAC_VAC"/> <int name="lighting"/> <int name="security"/> <int name="loadManagement"/> <int name="shuttersAndBlinds"/> </list> <list href="knx:ranges/dpt20_007" is="obix:range"> <int name="basicAlarm"/> <int name="extendedAlarm"/> <int name="void"/> </list> <list href="knx:ranges/dpt20_008" is="obix:range"> <int name="disabled_PSU_DPSU_fixedOff"/> <int name="enabled_PSU_DPSU_fixedOn"/> <int name="auto_PSU_DPSU_automaticOnOff"/> </list> <list href="knx:ranges/dpt20_011" is="obix:range"> <int name="noFault"/> <int name="generalDeviceFault"/> <int name="communicationFault"/> <int name="configurationFault"/> <int name="hardwareFault"/> <int name="softwareFault"/> <int name="insufficientNonVolatileMemory"/> <int name="insufficientVolatileMemory"/> <int name="memoryAllocationCommandWithSize_0_Received"/> <int name="CRC_error"/> <int name="watchdogResetdetected"/> <int name="invalidOpcodeDetected"/> <int name="generalProtectionFault"/> <int name="maximalTableLength exceeded"/> <int name="undefinedLoadCommandReceived"/> <int name="groupAddressTable is not sorted"/> <int name="invalidConnectionNumber_TSAP"/> <int name="invalidGroupObjectNumber_ASAP"/> <int name="groupObjectTypeExceeds"/> </list> <list href="knx:ranges/dpt20_012" is="obix:range"> <int name="noFault"/> <int name="sensorFault"/> <int name="processFault_controllerFault"/> <int name="actuatorFault"/> <int name="otherFault"/> </list> <list href="knx:ranges/dpt20_013" is="obix:range"> <int name="notActive"/> <int name="1s"/> <int name="2s"/> <int name="3s"/> <int name="5s"/> <int name="10s"/> <int name="15s"/> <int name="20s"/> <int name="30s"/> <int name="45s"/> <int name="1min"/> <int name="1,25min"/> ```

KNX™ DPT	oBIX contract
	<int name="1,5min"/> <int name="2min"/> <int name="2,5min"/> <int name="3min"/> <int name="5min"/> <int name="15min"/> <int name="20min"/> <int name="30min"/> <int name="1h"/> <int name="2h"/> <int name="3h"/> <int name="5h"/> <int name="12h"/> <int name="24h"/> </list> <list href="knx:ranges/dpt20_017" is="obix:range"> <int name="inactive"/> <int name="digitalInputNotInverted"/> <int name="digitalInputInverted"/> <int name="analogInput_0_to_100_percent"/> <int name="temperatureSensorInput"/> </list>
DPT 21.001	<bool> <bool name="OutOfService"/> <bool name="Fault"/> <bool name="Overridden"/> <bool name="InAlarm"/> <bool name="AlarmUnAck"/> </bool>
DPT 21.002	<bool> <bool name="UserStopped" /> <bool name="OwnIA"> <bool name="VerifyMode"/> </bool>
DPT 23.xxx	<enum> <int name="s"/> </enum> **Ranges:** <list href="knx:ranges/dpt23_001" is="obix:range"> <int name="off"/> <int name="on"/> <int name="offOn"/> <int name="onOff"/> </list> <list href="knx:ranges/dpt23_002" is="obix:range"> <int name="noAlarmIsUsed"/> <int name="alarmPositionIsUp"/> <int name="alarmPositionIsDown"/> </list> <list is="obix:range" href="knx:ranges/dpt23_003"> <int name="Up"/> <int name="Down"/> <int name="UpDown"/> <int name="DownUp"/> </list>
DPT 24.xxx	<str/>
DPT 26.xxx	<obj> <bool name="B" /> <int name="Scene-Number"/> </obj>
DPT 27.xxx	<obj> <bool name="s0"/> <bool name="m0" /> <bool name="s1"/> <bool name="m1"/> <bool name="s2"/> <bool name="m2"/> <bool name="s3"/> <bool name="m3"/> <bool name="s4"/> <bool name="m4"/> <bool name="s5"/> <bool name="m5"/> <bool name="s6"/> <bool name="m6"/> <bool name="s7"/> <bool name="m7"/> <bool name="s8"/> <bool name="m8"/> <bool name="s9"/> <bool name="m9"/> <bool name="s10"/> <bool name="m10"/> <bool name="s11"/> <bool name="m11"/> <bool name="s12"/> <bool name="m12"/>

KNX™ DPT	oBIX contract
	<bool name="s13"/> <bool name="m13"/> <bool name="s14"/> <bool name="m14"/> <bool name="s15"> <bool name="m15"/> </obj>
DPT 28.xxx	<str/>
DPT 202.xxx	DPT 202.001 combines DPT 5.004 and DPT 21.001 DPT 202.002 combines DPT 5.010 and DPT 21.001
DPT 203.xxx	DPT 203.002 combines DPT 7.002 and DPT 21.001 DPT 203.003 combines DPT 7.003 and DPT 21.001 DPT 203.004 combines DPT 7.004 and DPT 21.001 DPT 203.005 combines DPT 7.005 and DPT 21.001 DPT 203.006 combines DPT 7.006 and DPT 21.001 DPT 203.007 combines DPT 7.007 and DPT 21.001 DPT 203.011 combines DPT 14.077 and DPT 21.001 DPT 203.012 combines DPT 7.001 and DPT 21.001 DPT 203.013 combines DPT 14.019 and DPT 21.001 DPT 203.014 combines DPT 9.024 and DPT 21.001 DPT 203.015 combines DPT 9.006 and DPT 21.001 DPT 203.017 combines DPT 5.001 and DPT 21.001
DPT 204.xxx	DPT 204.001 combines DPT 6.001 and DPT 21.001
DPT 205.xxx	DPT 205.002 combines DPT 8.002 and DPT 21.001 DPT 205.003 combines DPT 8.003 and DPT 21.001 DPT 205.004 combines DPT 8.004 and DPT 21.001 DPT 205.005 combines DPT 8.005 and DPT 21.001 DPT 205.006 combines DPT 8.006 and DPT 21.001 DPT 205.007 combines DPT 8.007 and DPT 21.001
DPT 217.xxx	<str> <int name="magicNumber"/> <int name="versionNumber"/> <int name="revisionNumber"/> </str>
DPT 218.xxx	DPT 218.001 combines DPT 14.076 and DPT 21.001
DPT 219.xxx	<obj> <bool name="Ack_Sup"/> <bool name="TS_Sup"/> <bool name="AlarmText_Sup"/> <bool name="ErrorCode_Sup"/> <bool name="InAlarm_Sup"/> <bool name="AlarmUnAck_Sup"/> <bool name="Locked_Sup"/> <int name="LogNumber"/> <int name="AlarmPriority"/> <int name="ApplicationArea"/> <int name="ErrorClass"/> </obj>
DPT 221.xxx	<str> <int name="manufacturerCode"/> <int name="incrementedCode"/> </str>
DPT 222.100	<obj unit="obix:units/celsius_degrees"> <real name="Comf"/> <real name="Stdby"/> <real name="Eco"/> </obj>
DPT 222.101	<obj unit="obix:units/kelvin_degrees"> <real name="ShiftComf"/> <real name="ShiftStdby"/> <real name="ShiftEco"/> </obj>

B.4.2 KNX™ Interworking Proxy resource content structure

Constants:

The KNX™ interworking proxy uses the following AN standard.

Area network standard	ANStandard extension
ISO_14543_3 [i.9]	`<obj name=" ISO_14543_3"/>`

oBix contracts

A KNX™ InterworkingDescriptor contract is defined for interworking proxies supporting the KNX™ standard. This contract overloads the M2M InterworkingDescriptor contract, and contains the following additional sub-elements:

<table>
<tr><th>oBIX Type</th><th>Name</th><th>Value</th></tr>
<tr><td><ope></td><td>knxCreateNetwork</td><td>This operation creates a KNX™ network.
IN: knx:CreateNetworkInput
<table>
<tr><th>oBIX Type</th><th>Name</th><th>Value</th></tr>
<tr><td><str></td><td>knxNetworkID</td><td>KNX™ network ID.</td></tr>
<tr><td><uri></td><td>knxNetworkTopology</td><td>A M2M content instance URI containing the KNX™ network topology document. The KNX™ network topology document is a document describing the network topology (e.g. generated from an ETS4 export) and its definition is out of scope of the present document.</td></tr>
</table>
OUT: empty</td></tr>
</table>

The KNX™ InterworkingDescriptor contract uses the http://uri.etsi.org/m2m/knx/obix namespace:

```
<obj href="knx:InterworkingDescriptor" is="m2m:InterworkingDescriptor"
   xmlns:knx="http://uri.etsi.org/m2m/knx/obix" xmlns:m2m="http://uri.etsi.org/m2m/obix">
   xmlns="http://obix.org/ns/wsdl/1.1">

 <ope name="knxCreateNetwork" in="knx:CreateNetworkInput"/>
</obj>
```

Representation example

```
<obj xmlns="http://obix.org/ns/schema/1.1">
 <str name="interworkingProxyID" val="IPU_KNX"/>
 <list name="supportedTechnologies">
  <obj>
   <str name="anStandard" val="ISO_14543_3"/>
  </obj>
 </list>
 <ope name="knxCreateNetwork" href="/m2m/applications/IPU_KNX/retargeting1/knxCreateNetwork"/>
 <ref name="knxTopology" href="/m2m/applications/IPU_KNX/containers/topology"/>
 <list name="networks">
  <ref href="/m2m/applications/NW_nw1/containers/DESCRIPTOR/contentInstances/latest/content/"/>
 </list>
</obj>
```

B.4.3 KNX™ Network resource content structure

oBix contracts

A KNX™ NetworkDescriptor contract is defined for interworking proxies supporting the KNX™ standard. This contract overloads the M2M NetworkDescriptor contract, and contains the following additional sub-elements:

<table>
<tr><th>oBIX Type</th><th>Name</th><th>Value</th></tr>
<tr><td><str></td><td>knxNetworkName</td><td>KNX™ network name as defined when the KNX network was created (e.g. nw1)</td></tr>
<tr><td><ope></td><td>knxUpdateNetwork</td><td>This operation updates the KNX™ network.
IN: knx:UpdateNetworkInput
<table>
<tr><th>oBIX Type</th><th>Name</th><th>Value</th></tr>
<tr><td><uri></td><td>knxNetworkTopology</td><td>A M2M content instance URI containing the KNX™ network topology document. The KNX™ network topology document is a document describing the network topology (e.g. generated from an ETS4 export) and its definition is out of scope of the present document.</td></tr>
</table>
OUT: empty</td></tr>
<tr><td><ope></td><td>knxDeleteNetwork</td><td>This operation deletes the KNX™ network.
IN: empty
OUT: empty</td></tr>
</table>

The KNX™ NetworkDescriptor contract uses the http://uri.etsi.org/m2m/ knx/obix namespace:

```
<obj href="knx:NetworkDescriptor" is="m2m:NetworkDescriptor"
   xmlns:knx="http://uri.etsi.org/m2m/knx/obix" xmlns:m2m="http://uri.etsi.org/m2m/obix">
   xmlns="http://obix.org/ns/wsdl/1.1">

  <str name="knxNetworkName"/>
  <ope name="knxUpdateNetwork" in="knx:CreateNetworkInput"/>
  <ope name="knxDeleteNetwork"/>
</obj>
```

Representation example

```
<obj xmlns="http://obix.org/ns/schema/1.1">
 <str name="networkID" val="NW_nw1"/>
 <str name="knxNetworkName" val="nw1"/>
 <ope name="knxUpdateNetwork" href="/m2m/applications/NW_nw1/retargeting1/knxUpdateNetwork"/>
 <ope name="knxDeleteNetwork" href="/m2m/applications/NW_nw1/retargeting1/knxDeleteNetwork"/>
 <list name="nodes">
 <ref href="/m2m/applications/DEV_nw1_grp_0_0_1/containers/DESCRIPTOR/contentInstances/latest/
content"/>
  <ref href="/m2m/applications/DEV_nw1_dev_1_1_10/containers/DESCRIPTOR/contentInstances/latest/
content"/>
 </list>
</obj>
```

B.4.4 KNX™ Group resource content structure (Device)

oBix contracts

A KNX™ GroupDescriptor contract is defined for interworking proxies supporting the KNX™ standard. This contract overloads the M2M NodeDescriptor contract, and contains the following additional sub-elements:

oBIX Type	Name	Value
<str>	knxGroupAddress	KNX™ Group address (e.g. 0/0/1)

The KNX™ GroupDescriptor contract uses the http://uri.etsi.org/m2m/knx/obix namespace:

```
<obj href="knx:GroupDescriptor" is="m2m:NodeDescriptor"
   xmlns:knx="http://uri.etsi.org/m2m/knx/obix" xmlns:m2m="http://uri.etsi.org/m2m/obix">
   xmlns="http://obix.org/ns/wsdl/1.1">

 <str name="knxGroupAddress"/>
</obj>
```

Representation example

```
<obj xmlns="http://obix.org/ns/schema/1.1">
 <str name="nodeID" val="DEV_nw1_grp_0_0_1"/>
 <str name="knxGroupAddress" val="0/0/1"/>
 <list name="applications">
  <ref
href="/m2m/applications/DEV_nw1_grp_0_0_1.1/containers/DESCRIPTOR/contentInstances/latest/
content"/>
 </list>
</obj>
```

B.4.5 KNX™ Group resource content structure (Application)

The M2M ApplicationDescriptor contract is not overloaded.

B.4.6 KNX™ Group resource content structure (Interface)

oBix contracts

A KNX™ GroupInterfaceDescriptor contract is defined for interworking proxies supporting the KNX™ standard. This contract overloads the M2M InterfaceDescriptor contract, and contains the following additional sub-elements:

oBIX Type	Name	Value
<str>	knxDptID	The KNX™ DPT identifier (e.g. 2.001)
<str>	knxDptName	The KNX™ DPT name (e.g. DPT_SwitchControl)
<obj>	knxDpt	The value of the KNX™ group. The representation of the DPT is inherited from an oBIX <obj> as defined in clause B.4.1. e.g. <enum name="knxDpt" href="/m2m/applications/DEV_nw1_grp_0_0_1.1 /containers/val1/contentInstances/latest/content"> <bool name="v"/> <bool name="c"/> <str name="raw"/> </enum>
<ope>	knxDptSet	Update the value of the KNX™ group. The representation of the DPT is inherited from an oBIX <obj> as defined in clause B.4.1. e.g. <enum name="knxDpt" val="controlOn "/> **IN:** *<obj>* **OUT:** empty

The KNX™ GroupInterfaceDescriptor contract uses the http://uri.etsi.org/m2m/knx/obix namespace:

```
<obj href="knx:GroupInterfaceDescriptor" is="m2m:InterfaceDescriptor"
   xmlns:knx="http://uri.etsi.org/m2m/knx/obix" xmlns:m2m="http://uri.etsi.org/m2m/obix">
   xmlns="http://obix.org/ns/wsdl/1.1">

  <str name="knxDptID"/>
  <str name="knxDptName"/>
  <obj name="knxDpt"/>
  <ope name="knxDptSet" in="obix:obj">
</obj>
```

Representation example

```
<obj xmlns="http://obix.org/ns/schema/1.1">
 <str name="knxDptID" val="2.001"/>
 <str name="knxDptName" val="DPT_SwitchControl"/>
 <enum name="knxDpt" href="/m2m/applications/DEV_nw1_grp_0_0_1.1/containers/val1/
contentInstances/latest/content"/>
 <ope name="knxDptSet" href="/m2m/applications/DEV_nw1_grp_0_0_1.1/retargeting2/ope1">
</obj>
```

B.4.7 KNX™ Node resource content structure (Device)

oBix contracts

A KNX™ NodeDescriptor contract is defined for interworking proxies supporting the KNX™ standard. This contract overloads the M2M NodeDescriptor contract, and contains the following additional sub-elements:

oBIX Type	Name	Value
<str>	knxDeviceAddress	KNX™ physical address of the node (e.g. 1.1.10)
<str>	knxManufacturerID	KNX™ manufactured ID (e.g. M-0001)
<str>	knxManufacturerName	KNX™ manufacturer name (e.g. Manufacturer1)
<str>	knxFirmwareID	KNX™ firmare ID (e.g. 981D-01-54E8)
<str>	knxAreaName	KNX™ area name (e.g. "Area1")
<str>	knxLineName	KNX™ kine name (e.g. "Line1")
<str>	knxDeviceName	KNX™ device name (e.g. "Load switch N 511/02 (8-fold), blink before off")
<str>	knxDeviceDescription	KNX™ device description (e.g. "8x16 A")

The KNX™ NodeDescriptor contract uses the http://uri.etsi.org/m2m/knx/obix namespace:

```
<obj href="knx:NodeDescriptor" is="m2m:NodeDescriptor"
   xmlns:knx="http://uri.etsi.org/m2m/knx/obix" xmlns:m2m="http://uri.etsi.org/m2m/obix">
   xmlns="http://obix.org/ns/wsdl/1.1">

 <str name="knxDeviceAddress"/>
 <str name="knxManufacturerID"/>
 <str name="knxManufacturerName"/>
 <str name="knxFirmwareID"/>
 <str name="knxAreaName"/>
 <str name="knxLineName"/>
 <str name="knxDeviceName"/>
 <str name="knxDeviceDescription"/>
</obj>
```

Representation example

```
<obj xmlns="http://obix.org/ns/schema/1.1">
 <str name="nodeID" val="DEV_nw1_dev_1_1_10"/>
 <str name="knxDeviceAddress" val="1.1.10"/>
 <str name="knxManufacturerID" val="M-0001"/>
 <str name="knxManufacturerName" val="Manufacturer1"/>
```

```
 <str name="knxFirmwareID" val="981D-01-54E8"/>
 <str name="knxAreaName" val="Area1"/>
 <str name="knxLineName" val="Line1"/>
 <str name="knxDeviceName" val="Load switch N 511/02 (8-fold), blink before off"/>
 <str name="knxDeviceDescription" val="8x16 A"/>
 <list name="applications">
  <ref href="/m2m/applications/DEV_nw1_dev_1_1_10.1/containers/DESCRIPTOR/contentInstances/
latest/content"/>
 </list>
</obj>
```

B.4.8 KNX™ Node resource content structure (Application)

The M2M ApplicationDescriptor contract is not overloaded.

B.4.9 KNX™ Node resource content structure (Interface)

oBix contracts

A KNX™ NodeInterfaceDescriptor contract is defined for interworking proxies supporting the KNX™ standard. This contract overloads the M2M InterfaceDescriptor contract, and contains the following additional sub-elements:

oBIX Type	Name	Value	
<str>	knxObjectID	The object ID of the KNX™ object (e.g. M-0001_A-981D-01-54E8_O-7)	
<str>	knxObjectName	The object name of the KNX™ object (e.g. Switch, Channel B)	
<str>	knxObjectDescription	The object ID of the KNX™ object (e.g. On / Off)	
<str>	knxDptID	KNX™ DPT ID (e.g. 2.001)	
<str>	knxDptName	KNX™ DPT name (e.g. DPT_SwitchControl)	
<obj> (list of)	knxGroups	List of KNX™ group linked to the interface.	
	<str>	knxGroupAddress	The linked KNX™ groups (e.g. 0/0/1)
	<obj>	knxDpt	A reference to the KNX™ group value. The representation of the DPT is inherited from an oBIX <obj> as defined in clause B.4.1.
	<ope>	knxDptSet	A reference to the KNX™ group set operation The representation of the DPT is inherited from an oBIX <obj> as defined in clause B.4.1.
	<bool>	knxReadFlag	Read flag (e.g. true)
	<bool>	knxWriteFlag	Write flag (e.g. false)

The KNX™ NodeInterfaceDescriptor contract uses the http://uri.etsi.org/m2m/knx/obix namespace::

```
<obj href="knx:NodeInterfaceDescriptor" is="m2m:InterfaceDescriptor"
   xmlns:knx="http://uri.etsi.org/m2m/knx/obix" xmlns:m2m="http://uri.etsi.org/m2m/obix">
   xmlns="http://obix.org/ns/wsdl/1.1">

   <str name="knxObjectID"/>
   <str name="knxObjectName"/>
   <str name="knxObjectDescription"/>
   <str name="knxDptID"/>
   <str name="knxDptName"/>
   <list name="knxGroups">
    <obj>
      <str name="knxGroupAddress"/>
      <obj name="knxDpt"/>
      <ope name="knxDptSet">
      <bool name="knxReadFlag"/>
      <bool name="knxWriteFlag"/>
    </obj>
   </list>
</obj>
```

Representation example

```
<obj xmlns="http://obix.org/ns/schema/1.1">
  <obj>
   <str name="knxObjectID" val="M-0001_A-981D-01-54E8_O-7"/>
   <str name="knxObjectName" val="Switch, Channel B"/>
   <str name="knxObjectDescription" val="On / Off"/>
   <str name="knxDptID" val="2.001"/>
   <str name="knxDptName" val="DPT_SwitchControl"/>
   <list name="knxGroups">
    <obj>
      <str name="knxGroupAddress" val="0/0/1"/>
      <obj name="knxDpt" href="/m2m/applications/DEV_nw1_grp_0_0_1.1/containers/val1/
contentInstances/latest/content"/>
      <ope name="knxDptSet" href="/m2m/applications/DEV_nw1_grp_0_0_1.1/retargeting2/ope1">
      <bool name="knxReadFlag" val="false"/>
      <bool name="knxWriteFlag" val="true"/>
    </obj>
   </list>
  </obj>
 </list>
</obj>
```

Annex C:
Example of Interworking Using Containers and Subscriptions

The following call flows illustrate interworking a SEP2 device application and SEP2 network application to one another via ETSI M2M NSCL, DSCL and the interworking proxy (xIP).

In this example, the SEP2 device application hosts its own SEP2 compliant resource structure. The SEP2 device application is interworked with an ETSI M2M DSCL using an ETSI M2M device interworking proxy (DIP). The high level procedures/steps involved in interworking with a SEP2 Device and Application are outlined below:

1) The DIP first discovers the SEP2 device application and its resources and then performs an ETSI M2M registration to the DSCL on behalf of the SEP2 device application.
2) The DSCL in turn performs a SCL registration to the NSCL.
3) The SEP2 NA supports ETSI M2M functionality that it uses to perform ETSI M2M registration to the NSCL.
4) The SEP2 NA creates a M2M container resource in the NSCL which is then announced to the DSCL. This container is used by the NA to post SEP2 commands for the SEP2 device application.
5) The DIP discovers the announced container in the DSCL and creates a subscription to the corresponding container residing in the NSCL.
6) The DIP configures the "contact" attribute of this subscription with a URI of a resource residing in the interworking function (which the interworking function maps to a corresponding SEP2 device application resource).
7) Whenever the NA posts a new SEP2 command to the container, the NSCL generates a notification directly to the interworking function.
8) The interworking function forwards the SEP2 command to the SEP2 device application resource.
9) The SEP2 device application processes the SEP2 command.

The following ETSI M2M services are captured in the call flows below:

1) DIP registers to DSCL on behalf of SEP2 application
2) M2M/SEP2 NA registers to NSCL
3) NSCL announces NA to DSCL
4) DIP discovers announced NA via DSCL
5) NA creates container in NSCL
6) NSCL announces container to DSCL
7) DIP discovers announced container in DSCL
8) DIP subscribes to remote container in NSCL on behalf of SEP2 device application and configures contact attribute with address of a DIP resource (which it maps internally to a SEP2 resource)
9) NA updates NSCL container with SEP2 command
10) NSCL sends notification to DIP resource
11) DIP extracts SEP2 command from M2M notification
12) DIP forwards SEP2 command to corresponding SE2 resource
13) SEP2 device application receives SEP2 command and processes it

C.1 NA/DA Registration to NSCL/DSCL

The following is summary of the steps illustrated in figure C.1:

- DSCL registers with NSCL
- DIP registers to DSCL on behalf of SEP2 application
- M2M/SEP2 NA registers to NSCL and NA is announced to DSCL

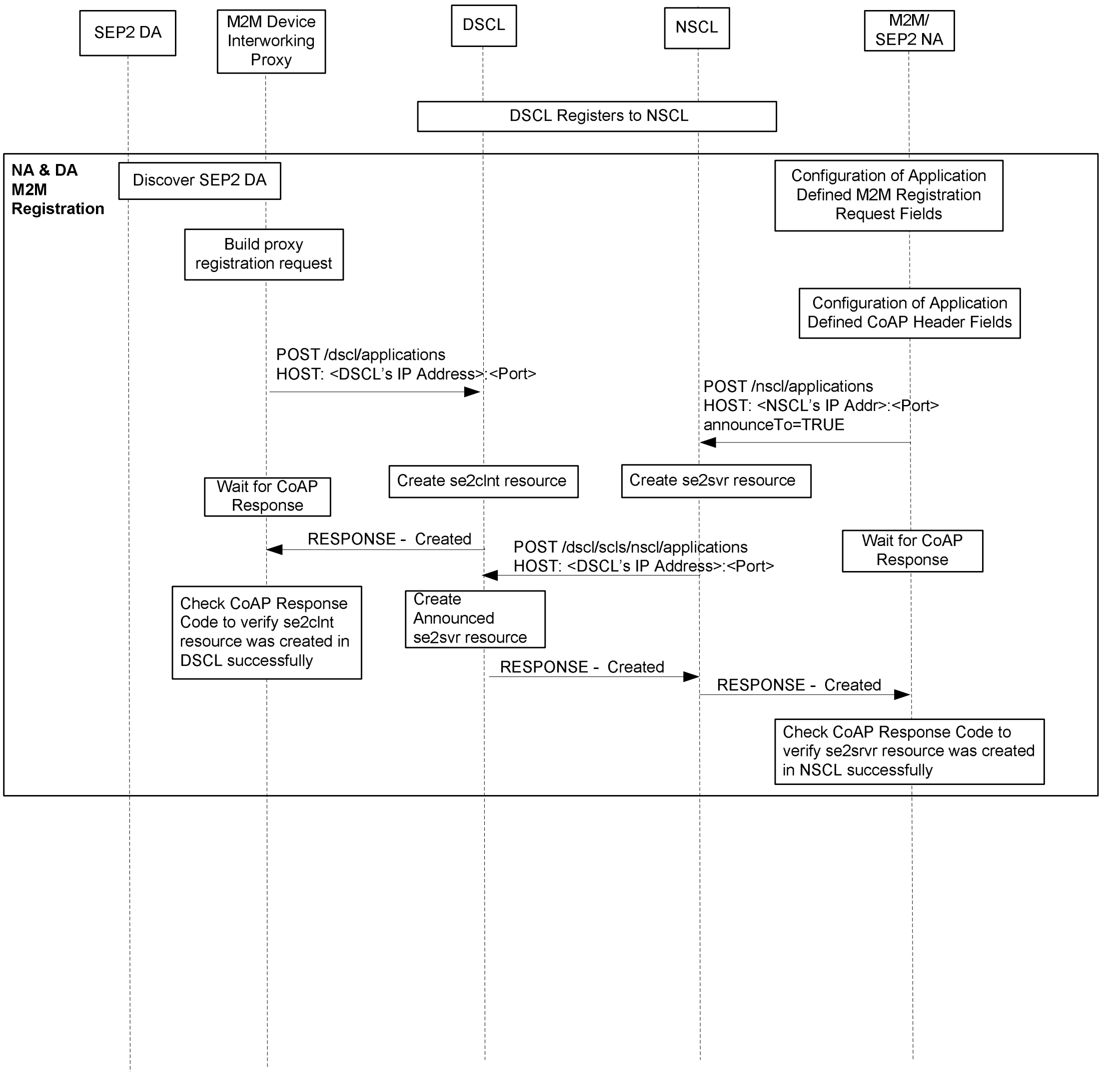

Figure C.1: NA and DA Application Registration to NSCL and DSCL

C.2 Discovery of Announced NA Resource

The following is summary of the steps illustrated in figure C.2:

- DIP does a retrieve on the DSCL's scls collection to discover the NSCL
- DIP does a retrieve on the DSCL's scls/nscl/applications collection to discover the announced NA (se2svr) application

Note that the DIP does not use the DSCL's discovery resource since it was not yet supported in this version of the DSCL.

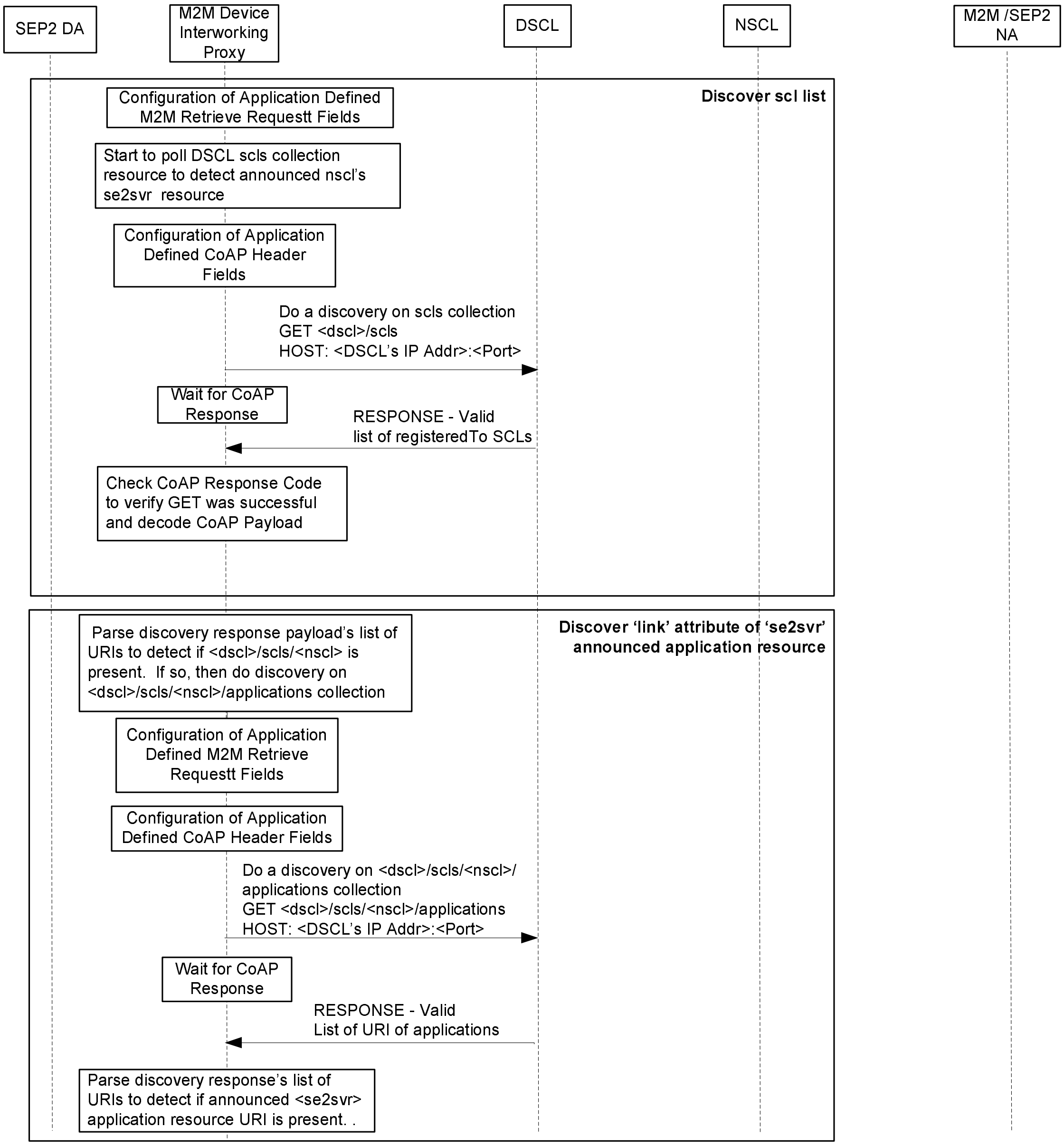

Figure C.2: DIP Discovers Announced NA (se2svr) DSCL Application Resource

C.3 NA Creates M2M Container Resource & Announces it to DSCL

The following is summary of the steps illustrated in figure C.3:

- NA creates a container resource in the NSCL named "socket1"
- NA requests that the NSCL announce the "socket1" container resource to the DSCL

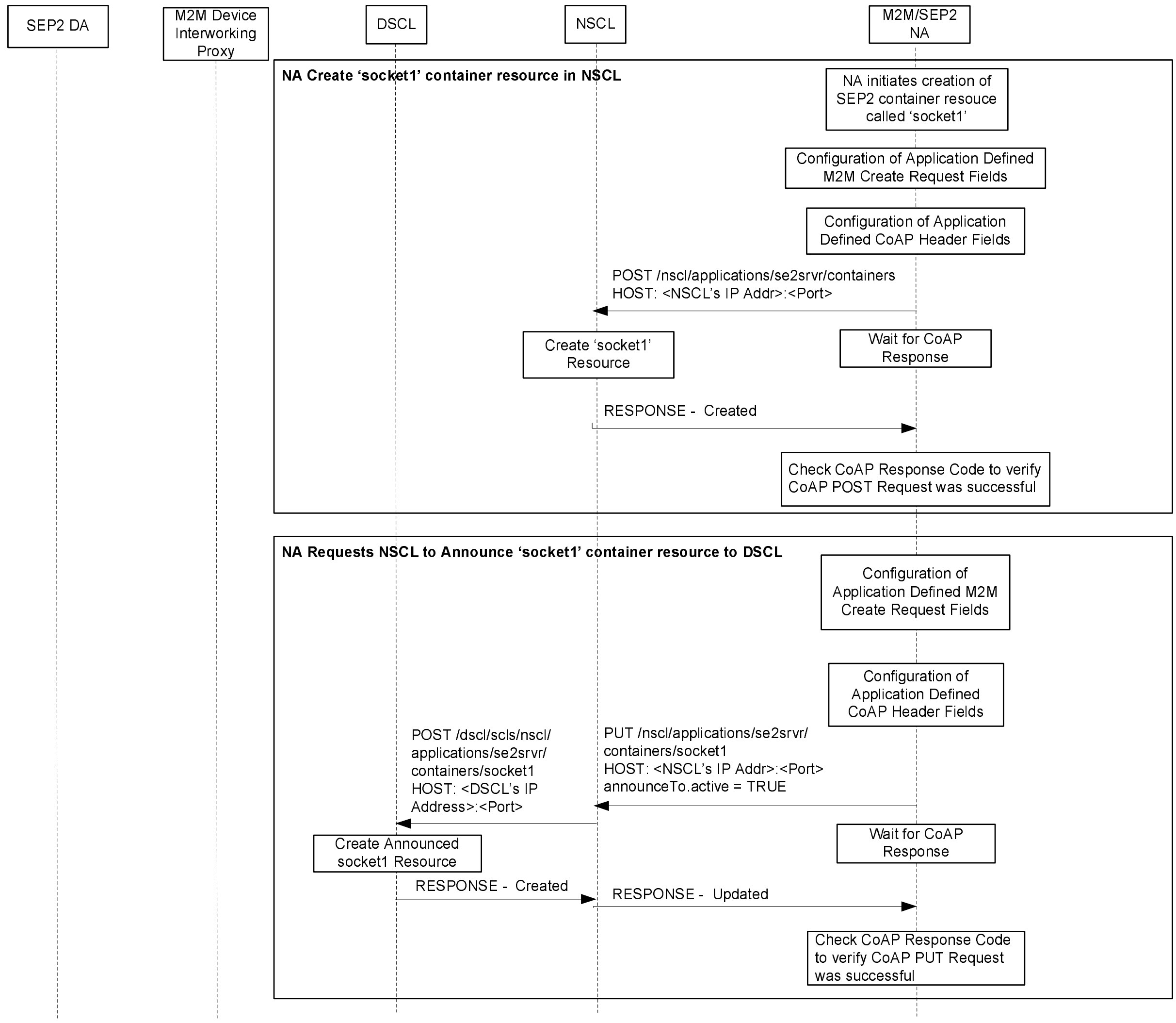

Figure C.3: NA Creates M2M Container Resource & Announces it to DSCL

C.4 Subscription to "socket1" NSCL Container Resource

The following is summary of the steps illustrated in figure C.4:

- DIP subscribes to the "socket1" container resource hosted on the NSCL
    - The DIP configures the contact attribute of the subscription to an absolute URI of a resource hosted in the interworking function (which it internally maps to a SEP2 resource).

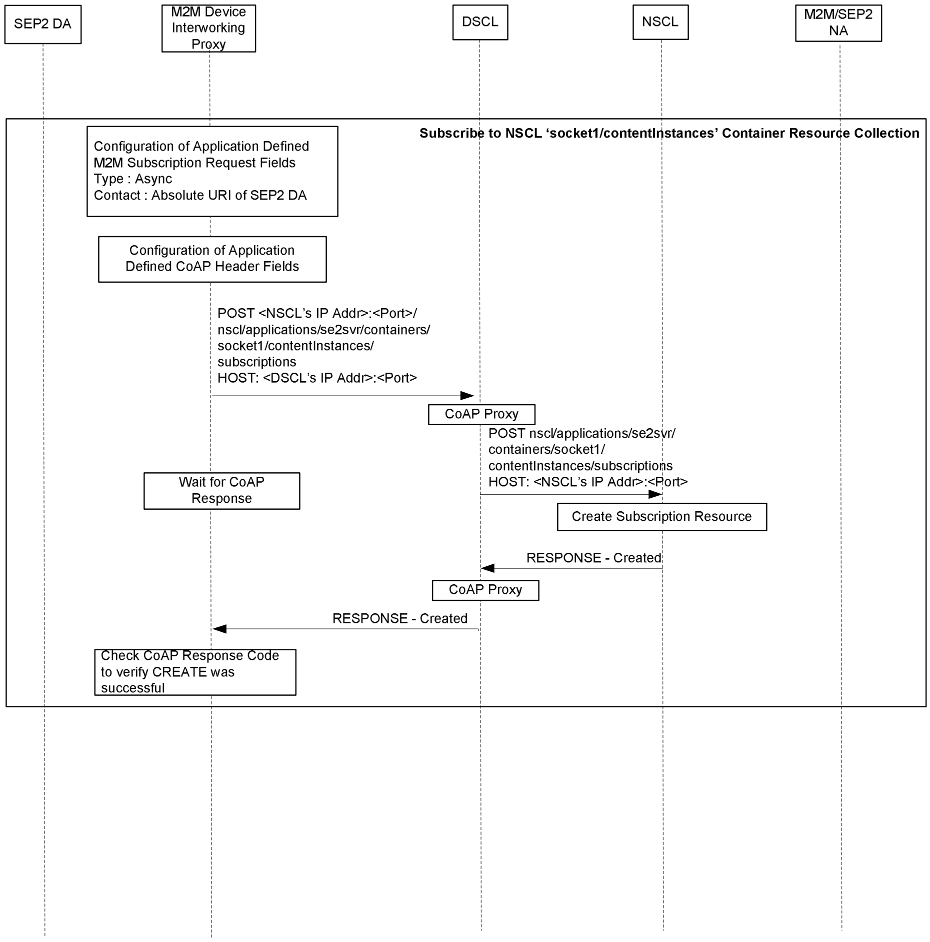

Figure C.4: DIP Subscribes to "socket1" NSCL Container Resource

C.5 NSCL Sends Notification to SEP2 DA

The following is summary of the steps illustrated in figure C.5:

- NA creates the SEP2 command
- NA encapsulates the SEP2 command in a ETSI M2M contentInstance and posts it to the "socket1" container resource
- NSCL generates a notification to the subscribed DIP URI
- The notification is coap proxied by DSCL to the DIP
- The DIP extracts the SEP2 command from the M2M notification
- The DIP forwards the SEP2 command to the SEP2 device application
- SEP2 device application processes SEP2 command

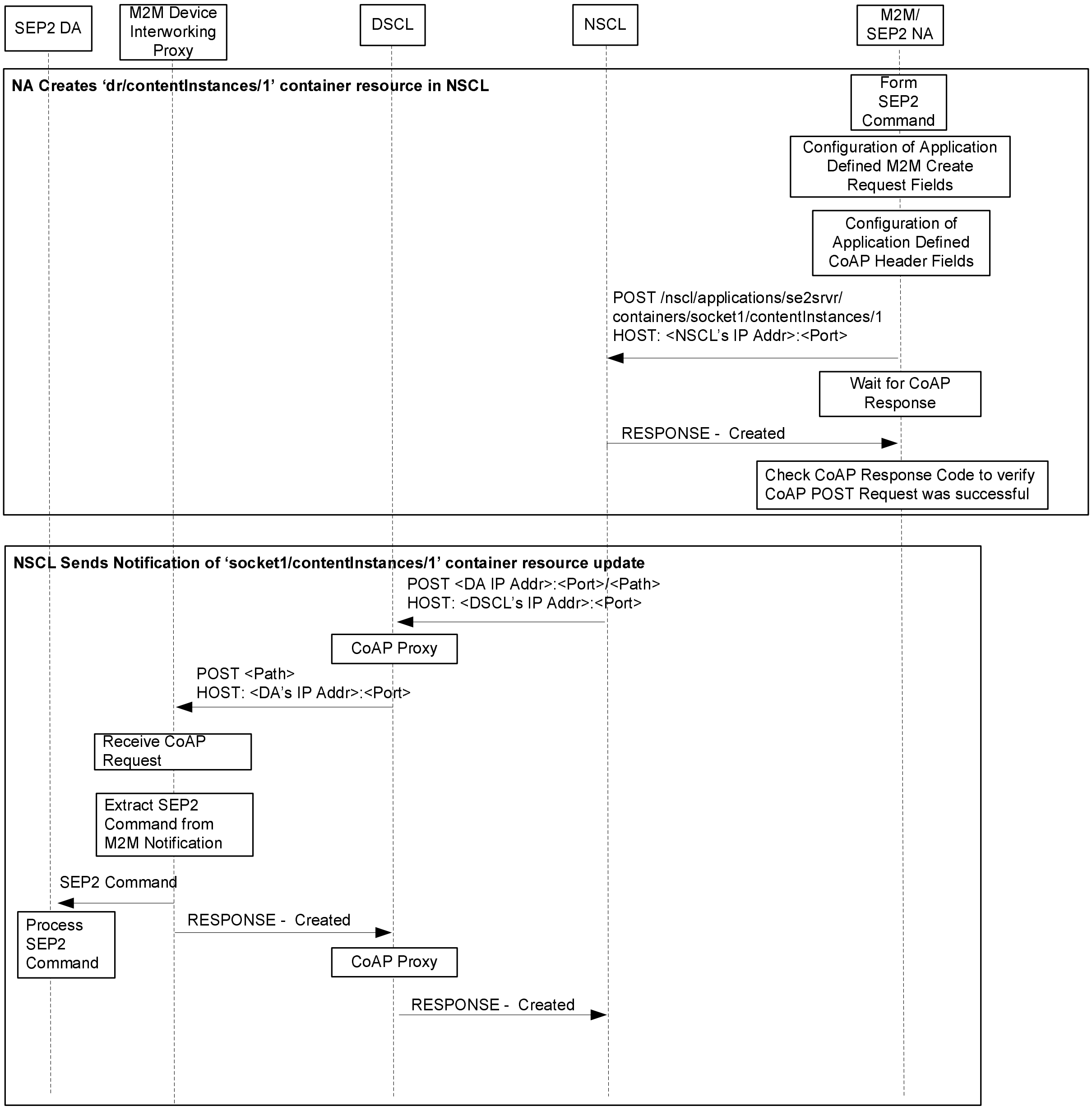

Figure C.5: NSCL Sends Notification to SEP2 DA

Annex D: Example of Interworking using aPoC

The following call flows illustrate interworking SEP2 area network devices with an ETSI M2M GSCL via aPoC functionality.

In this example, each ZigBee® SEP2 device hosts its own SEP2 application having its own locally hosted SEP2 resource structure. In addition, a M2M DA is used to discover and register each SEP2 application to the GSCL. Note that this functionality could reside within the ZigBee® Coordinator or the ETSI Gateway as GIP. The high level procedures/steps involved in interworking with a SEP2 area network are outlined below:

1) During registration with the GSCL, the M2M DA configures the ETSI M2M aPoC attribute with the base address of the SEP2 resource structure.

   a) By doing this, the M2M DA makes the SEP2 resource structure discoverable and accessible (by GA and NA) via the ETSI M2M GSCL.

2) To assist with discovery, the GSCL announces the registered SEP2 application to the NSCL so that it can be discovered by the remote NA.

3) In this example, the NA and GA are SEP2 aware applications having intelligence of the structure of the SEP2 resource tree hosted on the devices.

   a) In addition, the NA and GA are ETSI M2M aware applications that can register and communicate with the NSCL and GSCL, respectively.

4) Using the aPoC functionality of the GSCL, the NA and GA are able to access the SEP2 resource structure of the ZigBee® SEP2 applications.

In this example, the following use-cases are shown:

1) GA communicates locally with DA via GSCL aPoC

2) NA communicates remotely with DA via NSCL CoAP Proxying and GSCL aPoC

Through the above use-cases, the following ETSI M2M services were demonstrated:

- DA registration to GSCL
- GSCL aPoC functionality
- GSCL announcement of DA to NSCL
- GA discovery of DA via GSCL
- NA discovery of DA via NSCL
- GA communicating with DA via GSCL aPoC
- NA communicating with DA via GSCL aPoC

D.1 GA and DA Registration and Discovery

The following is summary of the steps illustrated in figure D.1:

- M2M DA discover SEP2 DA
- M2M DA registers to the GSCL on behalf of SEP2 DA
- GA registers to the GSCL

- GA discovers DA by doing a retrieve on the gscl/applications collection

Note that GA does not use the GSCL discovery resource to perform discovery since it was not supported in this version of the GSCL

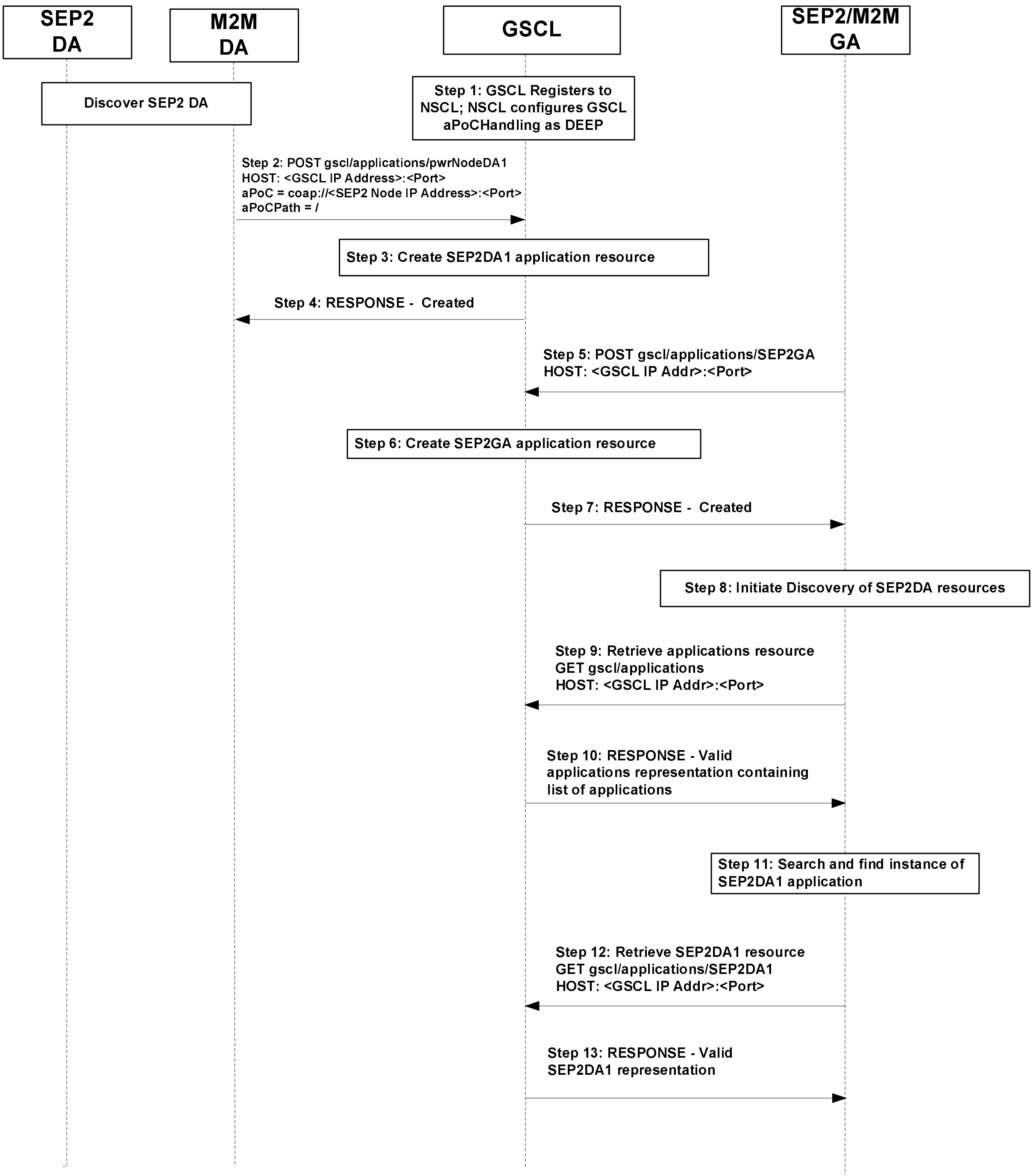

Figure D.1: GA and DA Register to GSCL, GA Discovers DA

D.2 GA and DA Communication via aPoC

The following is summary of the steps illustrated in figure D.2:

- GA reads SEP2 DA resource named "ubiami" by doing a GET using the GSCL aPoC functionality
- GA writes to the SEP2 DA resource named "a/relay" by doing a POST using the GSCL aPoC functionality

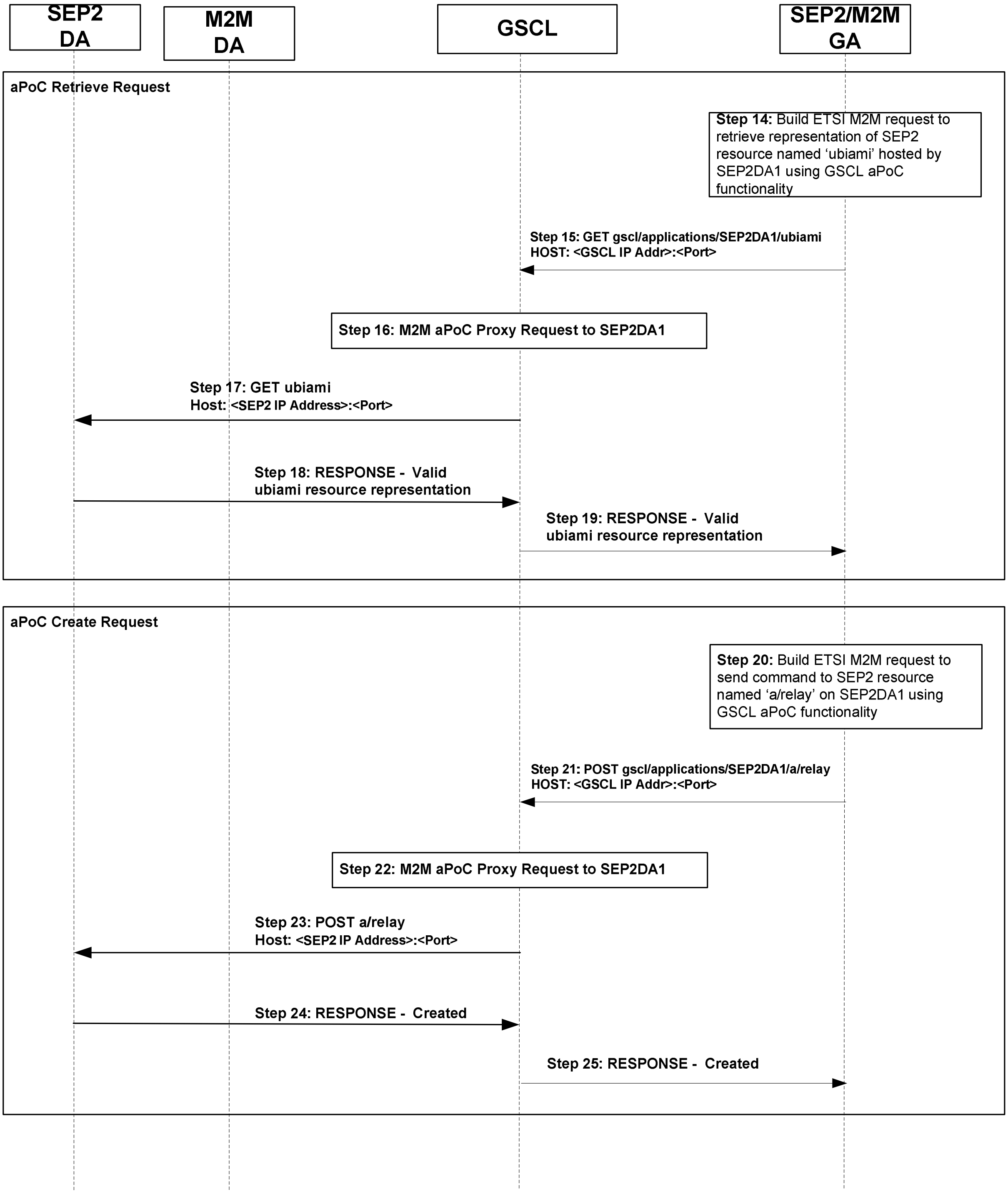

Figure D.2: GA communicates with SEP2 Resources via aPoC

D.3 NA and DA Registration and Discovery

The following is summary of the steps illustrated in figure D.3:

- GSCL registers with the NSCL
- M2M DA discovers SEP2 DA
- M2M DA registers to the GSCL on behalf of SEP2 DA
    - aPoC is configured with absolute URI of topmost resource in SEP2 DA's resource structure

- GSCL is instructed to announce DA to NSCL

- NA registers to NSCL
- NA discovers announced DA by first doing a retrieve on nscl/scls collection and then another retrieve on nscl/scls/gscl/applications collection

Note that NA does not use the NSCL discovery resource to perform discovery since it was not supported in this version of the NSCL.

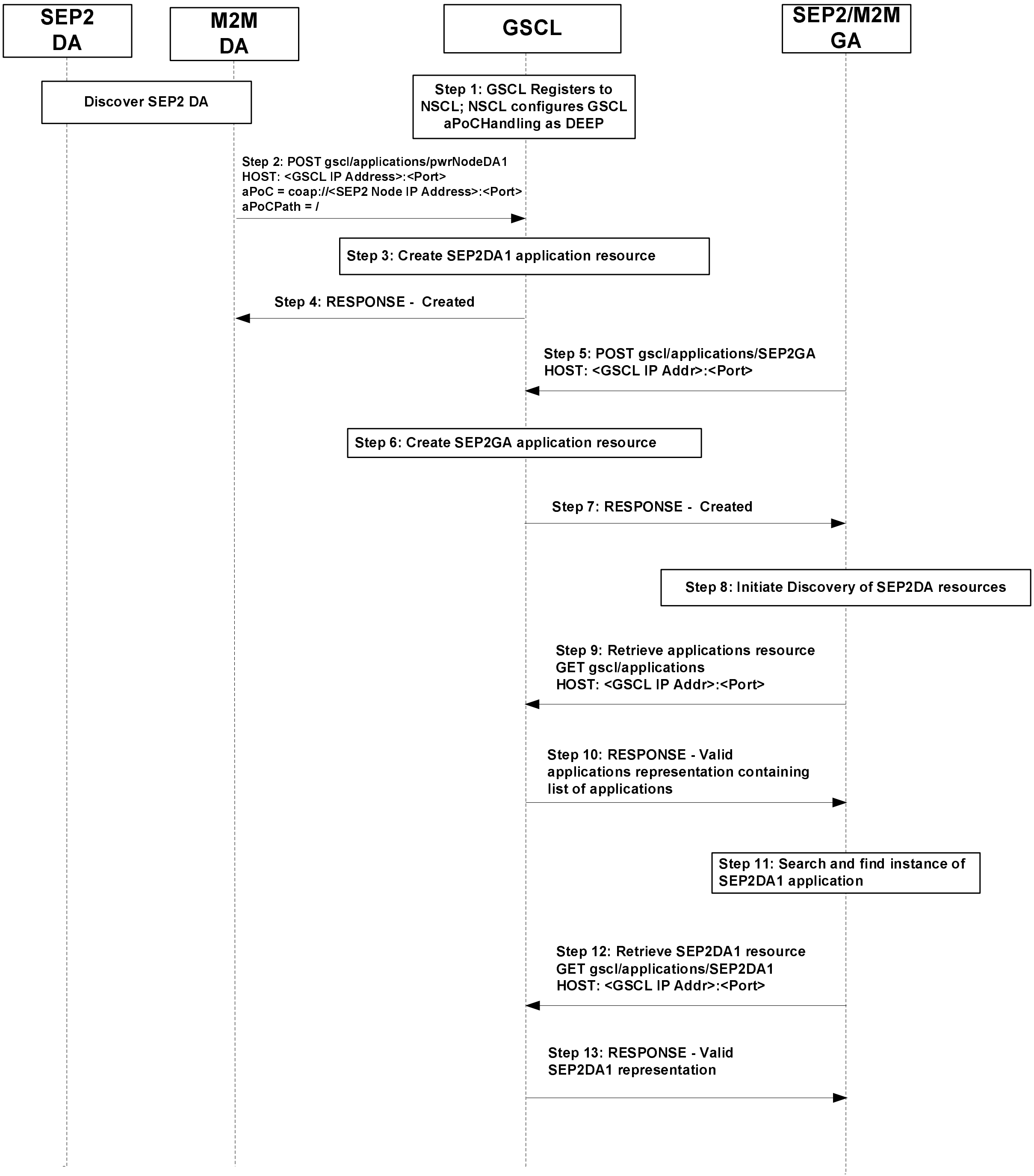

Figure D.3: NA/DA Registration and Discovery

D.4 NA to DA Communication via aPoC

The following is summary of the steps illustrated in figure D.4:

- NA reads SEP2 DA resource named "ubiami" by doing a GET which is CoAP proxied by NSCL and aPoC proxied by GSCL to DA
- NA writes to the SEP2 DA resource named "a/relay" by doing a POST which is CoAP proxied by NSCL and aPoC proxied by GSCL to SEP2 DA

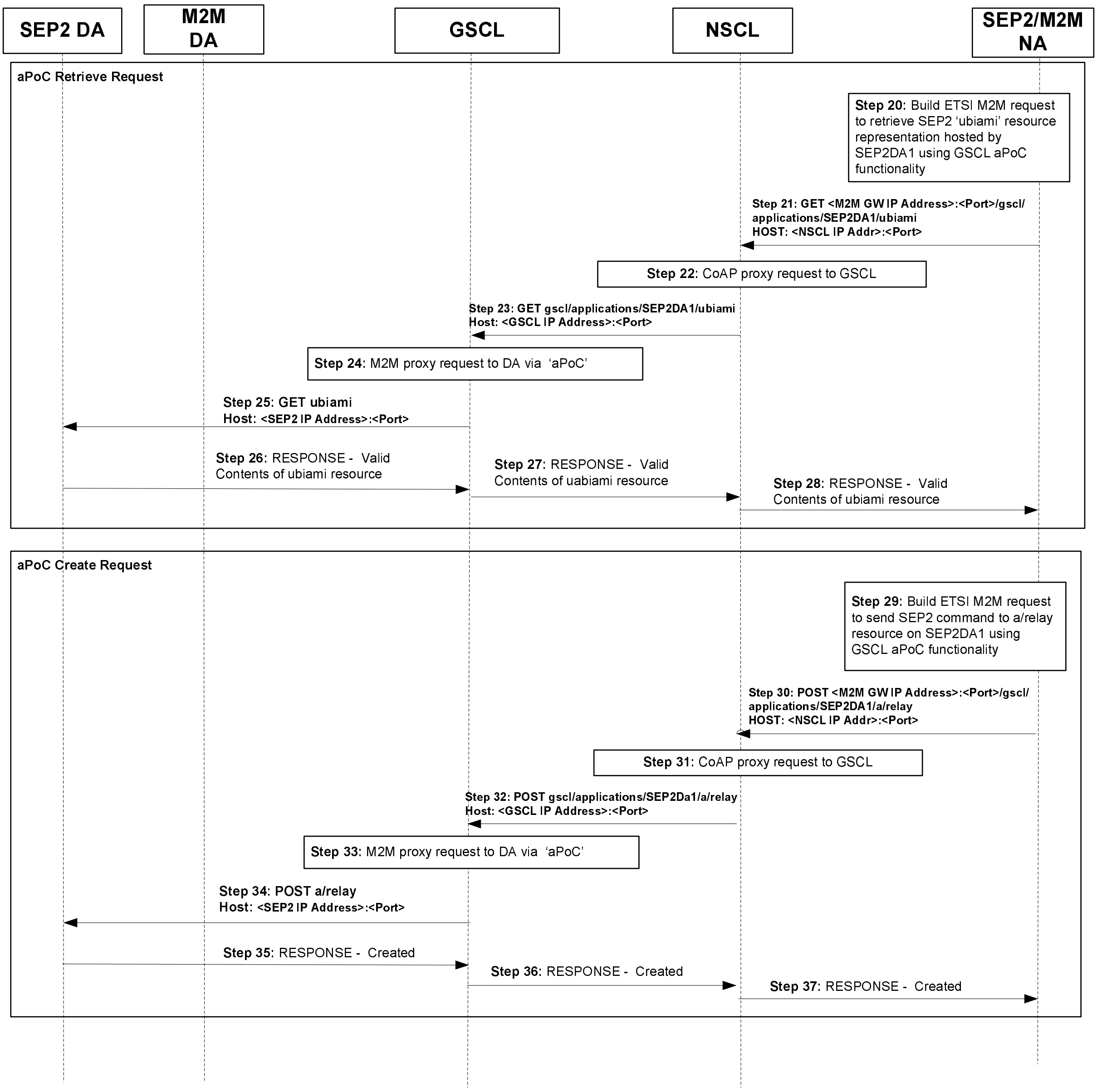

Figure D.4: NA to DA Communication via aPoC

Annex E: dId interface for limited resource devices

E.1 Scope

This annex defines a dId interface for external devices with limited resources. The dId interface applies between a micro GIP (or a micro DA hosted on an external device with limited resources), and an assisting GIP, hosted on a M2M Gateway.

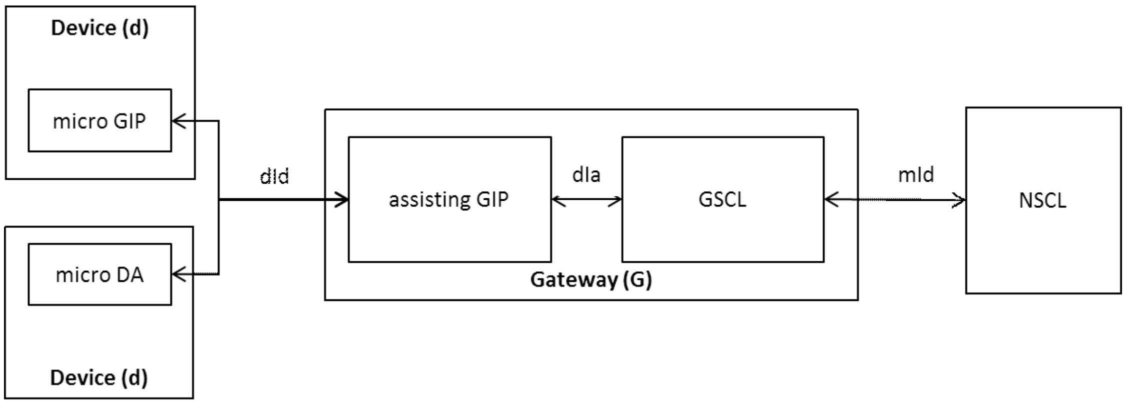

Figure E.1: dId binding for limited resource devices

The dId interface uses the M2M Area Network modeling defined in clause 5.1, with the OASIS.OBIX_1_1 ObjectSemantic defined in clause A.2 and detailed in annex B for various HAN technologies.

E.2 dId interface

Query parameters

The dId interface defines a set of query parameters used when a M2M <application> resource is addressed over the assisting GIP. These parameters are defined in clauses E.2.1 to E.2.4. Through these query parameters, the micro GIP or micro DA addresses M2M Area Network objects defined in clause 5.1, through the assisting GIP. The micro GIP/micro DA:

- does not need to know the exact location of the associated M2M resources (which are under control of the M2M Operator);
- does not need to define the exact configuration of the associated M2M resources (this information is provided to the M2M Operator by means of description templates that are out of scope of the present document);
- does not need to fully understand the ETSI M2M resource tree, as the assisting GIP will assist in building the resource tree as defined in the present document.

The intended effect is to obtain a **decoupling** between the software running on the micro GIP/micro DA (typically a dongle manufactured and sold independently of any specific M2M Operator), and the REST resources that an M2M Operator expects to be published on SCLs of his network. The adaptation is performed by the assisting GIP, under control of the M2M Operator.

Contracts

The dId interface uses the notion of "contract" defined in oBIX. A contract is a template model for an M2M Area Network object. A contract is defined through a <contract-uuid> which is a globally unique identifier, and is referenced in oBIX document through the XML "is" attribute. For instance:

```
  <obj is="nw.manufacturer1.com"/>
```

M2M Area Network objects (IPU, Network, Device, etc.) can be associated with a contract, by the micro GIP, to instruct the assisting GIP to apply a particular template of ETSI M2M resource tree.

The dId interface does not define the mechanism used by the assisting GIP to apply a particular template associated to a contract on a M2M Area Network object and does not define the format of the template itself. The dId binding only defines the existence of such contracts exchanged between the micro GIP and the assisting GIP and the way of representing contracts in M2M Area Network objects.

E.2.1 Interworking Proxy Unit

An "ipu" query parameter is defined to address an Interworking Proxy Unit: ipu=<ipu-uuid>. A <ipu-uuid> is a globally unique identifier allocated to the IPU by the micro GIP (e.g. zigbee.manufacture1.com).

E.2.1.1 CREATE

An Interworking Proxy Unit is created by triggering a M2M CREATE request to the assisting GIP. The Request URI contains the <ipu-uuid> of the Interworking Proxy Unit and, the request body contains the **descriptor of the IPU**. The descriptor document is associated with an oBIX contract that identifies the type of IPU (e.g. ipu.manufacturer1.com). According to the level of definition of the contract, the oBIX attributes associated to the descriptor document provided in the request can be partially provided in the request or completely defined by the contract.

The <contact-uuid> provided in the request allows the assisting GIP to derive the ETSI M2M resource tree associated to the IPU in the SCL.

```
CREATE /applications?ipu=<ipu-uuid>

  <obj is="<contract-uuid>">
    Interworking Proxy Unit descriptor as defined in annex B
  </obj>
```

EXAMPLE:

```
CREATE /applications?ipu=zigbee.manufacturer1.com

  <obj is="ipu.manufacturer1.com"/>
```

Upon receiving this request, the assisting GIP derives the ETSI M2M resource tree, by fetching resource description templates associated to the ipu.manufacturer1.com contract. This leads to the creation of the following ETSI M2M resource tree:

- /applications/zigbee.manufacturer1.com
- /applications/zigbee.manufacturer1.com/containers/descriptor
- /applications/zigbee.manufacturer1.com/containers/descriptor/contentInstances/latest

E.2.1.2 RETRIEVE

An Interworking Proxy Unit is retrieved by triggering a M2M RETRIEVE request to the assisting GIP. The Request URI contains the <ipu-uuid>. The request returns the descriptor associated to the IPU as stored in the SCL.

```
RETRIEVE /applications?ipu=<ipu-uuid>
```

E.2.1.3 UPDATE

Not applicable.

E.2.1.4 DELETE

An Interworking Proxy Unit is deleted by triggering a M2M DELETE request to the assisting GIP. The Request URI contains the <ipu-uuid>. The request deletes the entire Interworking Proxy Unit representation in the SCL.

```
DELETE /applications?ipu=<ipu-uuid>
```

E.2.2 Network

An "nw" query parameter is defined to address an Network: nw=<network-id>. A <network-id> is a unique identifier allocated to a Network by the micro GIP (e.g. nw1). A <network-id> is relative to an IPU.

E.2.2.1 CREATE

A Network is created by triggering a M2M CREATE request to the assisting GIP. The Request URI contains the <network-id> of the Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request body contains the **descriptor of the Network**. The descriptor document is associated with an oBIX contract that identifies the type of Network (e.g. nw.manufacturer1.com). According to the level of definition of the contract, the oBIX attributes associated to the descriptor document can be partially provided in the request or completed defined by the contract.

The <contact-uuid> provided in the request allows the assisting GIP to derive the ETSI M2M resource tree associated to the Network in the SCL.

```
CREATE /applications?ipu=<ipu-uuid>&nw=<network-id>

  <obj is="<contract-uuid>">
    Network descriptor as defined in annex B
  </obj>
```

As defined in annex B, the Interworking Proxy Unit descriptor includes the list of networks. If the list of networks ("networks" attribute) is provided by the micro GIP, when the Interworking Proxy Unit is created, the list of networks is updated by the micro GIP. If the list of networks is not provided, the list of networks is updated by the assisting GIP.

EXAMPLE:

```
CREATE /applications?ipu=zigbee.manufacturer1.com&nw=nw1

  <obj is="nw.manufacturer1.com"/>
```

Upon receiving this request, the assisting GIP derives the ETSI M2M resource tree, by fetching resource description templates associated to the nw.manufacturer1.com contract. This leads to the creation of the following ETSI M2M resource tree:

- /applications/nw1.zigbee.manufacturer1.com
- /applications/nw1.zigbee.manufacturer1.com/containers/descriptor
- /applications/nw1.zigbee.manufacturer1.com/containers/descriptor/contentInstances/latest

Additionally, if the list of networks is maintained by the assisting GIP, this also leads to the update of the following ETSI M2M resource:

- /applications/zigbee.manufacturer1.com/containers/DESCRIPTOR/contentInstances/latest

E.2.2.2 RETRIEVE

A Network is retrieved by triggering a M2M RETRIEVE request to the assisting GIP. The Request URI contains the <network-id> of the Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request returns the descriptor associated to the Network as stored in the SCL.

```
RETRIEVE /applications?ipu=<ipu-uuid>&nw=<network-id>
```

E.2.2.3 UPDATE

Not applicable.

E.2.2.4 DELETE

An Interworking Proxy Unit is deleted by triggering a M2M DELETE request to the assisting GIP. The Request URI contains the <network-id> of the Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request deletes the entire Network representation in the SCL.

```
DELETE /applications?ipu=<ipu-uuid>&nw=<network-id>
```

E.2.3 Device, Application and Interface

A "node" query parameter is defined to address a Device: node=<node-id>. A <node-id> is a unique identifier allocated to a Device by the micro GIP (e.g. mac1). A <node-id> is relative to a Network. An "app" query parameter is defined to address an Application: app=<application-id>. A <application-id> is a unique identifier allocated to an Application by the micro GIP (e.g. mainspoweroutlet). A <application-id> is relative to a node. An "itf" query parameter is defined to address an Interface: itf=<interface-id>. A <interface-id> is a unique identifier allocated to an Interface by the micro GIP (e.g. simplemetering). A <interface-id> is relative to an Application.

E.2.3.1 CREATE

E.2.3.1.1 Case 1: Define an area network device through a well-known device profile

When the micro GIP is able to associate to the Device a triplet { manufacturer, product, version }, the Device is created by triggering a M2M CREATE request to the assisting GIP. The Request URI contains the <node-id> of the Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request body contains the **descriptor of the Device**. The descriptor document is associated with an oBIX contract that identifies the exact type of the Device (e.g. partnum1.manufacturer1.com). According to the level of definition of the contract, the oBIX attributes associated to the description document can be partially provided in the request or completed defined by the contract.

The <contract-uuid> provided in the request allows the assisting GIP to derive the ETSI M2M resource tree associated to the Device and associated to each Application of the Device in the SCL.

```
CREATE /applications?ipu=<ipu-uuid>&nw=<network-id>&node=<node-id>

<obj is="<contract-uuid>">
  Device descriptor as defined in annex B
</obj>
```

As defined in annex B:

- The Network descriptor includes the list of nodes. If the list of nodes ("nodes" attribute) is provided by the micro GIP, when the Network is created, the list of nodes is updated by the micro GIP. If the list of nodes is not provided, the list of nodes is updated by the assisting GIP.
- The same consideration applies for the list of applications in the Device descriptor, and the list of interfaces in the Application descriptor. Typically in this case 1 the list of applications and the list of interface are not provided by the micro GIP.

EXAMPLE:

```
CREATE /applications?ipu=zigbee.manufacturer1.com&nw=nw1&node=mac1

  <obj is="partnum1.manufacturer1.com"/>
```

Upon receiving this request, the assisting GIP derives the ETSI M2M resource tree of the Device, and the ETSI M2M resource tree of each Application hosted by the Device, by fetching resource description templates associated to the partnum1.manufacturer1.com contract. This leads to the creation of the following ETSI M2M resource tree (in this example we assume that Interface descriptors are embedded in the Application descriptor):

- /applications/mac1.nw1.zigbee.manufacturer1.com
- /applications/mac1.nw1.zigbee.manufacturer1.com/containers/descriptor
- /applications/mac1.nw1.zigbee.manufacturer1.com/containers/descriptor/contentInstances/latest
- /applications/mainspoweroutlet.mac1.nw1.zigbee.manufacturer1.com
- /applications/mainspoweroutlet.mac1.nw1.zigbee.manufacturer1.com/containers/descriptor
- /applications/mainspoweroutlet.mac1.nw1.zigbee.manufacturer1.com/containers/descriptor/contentInstances/latest

Additionally, if the list of nodes is maintained by the assisting GIP, this also leads to the update of the following ETSI M2M resource:

- /applications/nw1.zigbee.manufacturer1.com/containers/DESCRIPTOR/contentInstances/latest

E.2.3.1.2 Case 2: Define an area network device through a set of well-known device application profiles

When the micro GIP is not able to determine the exact type of the Device, but it able to associate to each Application hosted by the Device an exact Application type, the Device is created by triggering a M2M CREATE request to the assisting GIP. The Request URI contains the <node-id> of the Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request body contains the **descriptor of the Device**. The descriptor document is associated with an oBIX contract that identifies the generic type of the Device (e.g. dev.manufacturer1.com). The descriptor document also contains the list of Applications hosted by the Device, where each Application is associated with an oBIX contract that identifies the exact type of the Application (e.g. mainspoweroutlet.manufacturer1.com). According to the level of definition of the contract, another oBIX attributes associated to the description document can be partially provided in the request or completely defined by the contract.

The Device Type <contract-uuid> provided in the request allows the assisting GIP to derive the ETSI M2M resource trees associated to the Device, and the Application Type <contract-uuid> provided in the request allows the assisting GIP to derive the ETSI M2M resource trees associated to each Application hosted by the Device.

```
CREATE /applications?ipu=<ipu-uuid>&nw=<network-id>&node=<node-id>

<obj is="<contract-uuid>">
  Device descriptor as defined in annex B (where the "applications" attribute is always provided)
  <list name="applications">
    <ref href="<application-id>" is="<contract-uuid>"/>
  </list>
</obj>
```

As defined in annex B:

- The Network descriptor includes the list of nodes. If the list of nodes ("nodes" attribute) is provided by the micro GIP, when the Network is created, the list of nodes is updated by the micro GIP. If the list of nodes is not provided, the list of nodes is updated by the assisting GIP.
- The same consideration applies for the list of applications in the Device descriptor, and the list of interfaces in the Application descriptor. Typically in this case 2 the list of applications is provided by the micro GIP and the list of interfaces is not provided by the micro GIP.

EXAMPLE:

```
CREATE /applications?ipu=zigbee.manufacturer1.com&nw=nw1&node=mac1

  <obj is="dev.manufacturer1.com">
    <list name="applications">
      <ref href="mainspoweroutlet" is="mainspoweroutlet.manufacturer1.com"/>
    </list>
  </obj>
```

Upon receiving this request, the assisting GIP derives the ETSI M2M resource tree of the Device, and the ETSI M2M resource tree of each Application hosted by the Device, by fetching resource description templates associated to the dev.manufacturer1.com contract and the mainspoweroutlet.manufacturer1.com contract. This leads to the creation of the following ETSI M2M resource tree (in this example we assume that Interface descriptors are embedded in the Application descriptor):

- /applications/mac1.nw1.zigbee.manufacturer1.com
- /applications/mac1.nw1.zigbee.manufacturer1.com/containers/descriptor
- /applications/mac1.nw1.zigbee.manufacturer1.com/containers/descriptor/contentInstances/latest
- /applications/mainspoweroutlet.mac1.nw1.zigbee.manufacturer1.com
- /applications/mainspoweroutlet.mac1.nw1.zigbee.manufacturer1.com/containers/descriptor
- /applications/mainspoweroutlet.mac1.nw1.zigbee.manufacturer1.com/containers/descriptor/contentInstances/latest

Additionally, if the list of nodes is maintained by the assisting GIP, this also leads to the update of the following ETSI M2M resource:

- /applications/nw1.zigbee.manufacturer1.com/containers/DESCRIPTOR/contentInstances/latest

E.2.3.1.3 Case 2 (variant)

As described in the previous clause, the case 2 allows the micro GIP to provide the descriptor of the Device but does not allow providing the descriptor of each Application. If the micro GIP needs to provide such descriptors, the following variation can be used.

When the Device is created, the request body still contains the **descriptor of the Device** but the list of Applications is not provided and therefore will be maintained by the assisting GIP. Each Application hosted by the Device is created by triggering a M2M CREATE request to the assisting GIP. The Request URI contains the <application-id> of the Application, the <node-id> of the associated Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request body contains the **descriptor of the Application**. The descriptor document is associated with an oBIX contract that identifies the exact type of the Application (e.g. mainspoweroutlet.manufacturer1.com).

```
CREATE /applications?ipu=<ipu-uuid>&nw=<network-id>&node=<node-id>&app=<application-id>

  <obj is="<contract-uuid>">
    Application descriptor as defined in annex B
  </obj>
```

E.2.3.1.4 Case 3: Define an area network device through a set of well-known interface profiles

When the micro GIP is not able to determine the exact type of the Device and the exact type of Applications, but it able to associate to each Interface of each Application hosted by the Device an exact Interface type, the Device is created by triggering a M2M CREATE request to the assisting GIP. The Request URI contains the <node-id> of the Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request body contains the **descriptor of the Device**. The descriptor document is associated with an oBIX contract that identifies the generic type of the Device (e.g. dev.manufacturer1.com). The descriptor document does not contain the list of Applications hosted by the Device. According to the level of definition of the contract, another oBIX attributes associated to the description document can be partially provided in the request or completely defined by the contract.

The Device Type <contract-uuid> provided in the request allows the assisting GIP to derive the ETSI M2M resource trees associated to the Device.

```
CREATE /applications?ipu=<ipu-uuid>&nw=<network-id>&node=<node-id>

<obj is="<contract-uuid>">
  Device descriptor as defined in annex B (where the "applications" attribute is not provided)
</obj>
```

Each Application hosted by the Device is created by triggering a M2M CREATE request to the assisting GIP. The Request URI contains the <application-id> of the Application, the <node-id> of the associated Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request body contains the **descriptor of the Application**. The descriptor document is associated with an oBIX contract that identifies the generic type of the Application (e.g. app.manufacturer1.com). The descriptor document also contains the list of Interfaces hosted by the Application, where each Interface is associated with an oBIX contract that identifies the exact type of the Interface (e.g. simplemetering.manufacturer1.com). According to the level of definition of the contract, another oBIX attributes associated to the description document can be partially provided in the request or completely defined by the contract.

The Application Type <contract-uuid> provided in the request allows the assisting GIP to derive the ETSI M2M resource trees associated to the Application, and the Interface Type <contract-uuid> provided in the request allows the assisting GIP to derive the ETSI M2M resource trees associated to each Interface hosted by the Application.

```
CREATE /applications?ipu=<ipu-uuid>&nw=<network-id>&node=<node-id>&app=<application-id>

  <obj is="<contract-uuid>">
    Application descriptor as defined in annex B (where the "interfaces" attribute is always
provided)
    <list name="interfaces">
      <ref href="<interface-id>" is="<contract-uuid>"/>
    </list>
  </obj>
```

As defined in annex B:

- The Network descriptor includes the list of nodes. If the list of nodes ("nodes" attribute) is provided by the micro GIP, when the Network is created, the list of nodes is updated by the micro GIP. If the list of nodes is not provided, the list of nodes is updated by the assisting GIP.
- The same consideration applies for the list of applications in the Device descriptor, and the list of interfaces in the Application descriptor. Typically in this case 4 the list of applications is not provided by the micro GIP and the list of interfaces is provided by the micro GIP.

EXAMPLE:

```
CREATE /applications?ipu=zigbee.manufacturer1.com&nw=nw1&node=mac1

  <obj is="dev.manufacturer1.com"/>
```

Upon receiving this request, the assisting GIP derives the ETSI M2M resource tree of the Device, by fetching resource description templates associated to the dev.manufacturer1.com contract. This leads to the creation of the following ETSI M2M resource tree:

- /applications/mac1.nw1.zigbee.manufacturer1.com
- /applications/mac1.nw1.zigbee.manufacturer1.com/containers/descriptor
- /applications/mac1.nw1.zigbee.manufacturer1.com/containers/descriptor/contentInstances/latest

Additionally, if the list of nodes is maintained by the assisting GIP, this also leads to the update of the following ETSI M2M resource:

- /applications/nw1.zigbee.manufacturer1.com/containers/DESCRIPTOR/contentInstances/latest

```
CREATE /applications?ipu=zigbee.manufacturer1.com&nw=nw1&node=mac1&app=app1

  <obj is="app.manufacturer1.com">
    <list name="interfaces">
      <ref href="simplemetering" is="simplemetering.manufacturer1.com"/>
    </list>
  </obj>
```

Upon receiving this request, the assisting GIP derives the ETSI M2M resource tree of the Application, and the ETSI M2M resource tree of each Interface hosted by the Application, by fetching resource description templates associated to the app.manufacturer1.com contract and the simplemetering.manufacturer1.com contract. This leads to the creation of the following ETSI M2M resource tree (in this example we assume that Interface descriptors are embedded in the Application descriptor):

- /applications/app1.mac1.nw1.zigbee.manufacturer1.com
- /applications/app1.mac1.nw1.zigbee.manufacturer1.com/containers/descriptor
- /applications/app1.mac1.nw1.zigbee.manufacturer1.com/containers/descriptor/contentInstances/latest

Additionally, because the list of applications is maintained by the assisting GIP, this also leads to the update of the following ETSI M2M resource:

- /applications/mac1.nw1.zigbee.manufacturer1.com/containers/DESCRIPTOR/contentInstances/latest

E.2.3.1.5 Case 3 (variant)

As described in the previous clauses, the case 3 allows the micro GIP to provide the descriptor of the Device, the descriptor of each Application but does not allow providing the descriptor of each Interface. If the micro GIP needs to provide such descriptors, the following variation can be used.

When the Application is created, the request body still contains the **descriptor of the Application** but the list of Interfaces is not provided and therefore will be maintained by the assisting GIP. Each Interface hosted by the Application is created by triggering a M2M CREATE request to the assisting GIP. The Request URI contains the <interface-id> of the Interface, the <application-id> of the associated Application, the <node-id> of the associated Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request body contains the **descriptor of the Interface**. The descriptor document is associated with an oBIX contract that identifies the exact type of the Interface (e.g. simplemetering.manufacturer1.com).

```
CREATE /applications?ipu=<ipu-uuid>&nw=<network-id>&node=<node-id>&app=<application-
id>&itf=<interface-id>

  <obj is="<contract-uuid>">
    Interface descriptor as defined in annex B
  </obj>
```

E.2.3.2 RETRIEVE

A Device is retrieved by triggering a M2M RETRIEVE request to the assisting GIP. The Request URI contains the <node-id> of the Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request returns the descriptor associated to the Device as stored in the SCL.

```
RETRIEVE /applications?ipu=<ipu-uuid>&nw=<network-id>&node=<noted-id>
```

An Application is retrieved by triggering a M2M RETRIEVE request to the assisting GIP. The Request URI contains the <application-id> of the Application, the <node-id> of the associated Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request returns the descriptor associated to the Application as stored in the SCL.

```
RETRIEVE /applications?ipu=<ipu-uuid>&nw=<network-id>&node=<noted-id>&app=<application-id>
```

An Interface is retrieved by triggering a M2M RETRIEVE request to the assisting GIP. The Request URI contains the <interface-id> of the Interface, the <application-id> of the associated Application, the <node-id> of the associated Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request returns the descriptor associated to the Interface as stored in the SCL.

```
RETRIEVE /applications?ipu=<ipu-uuid>&nw=<network-id>&node=<noted-id>&app=<application-id>&itf=<interface-id>
```

E.2.3.3 UPDATE

Not applicable.

E.2.3.4 DELETE

A Device is deleted by triggering a M2M DELETE request to the assisting GIP. The Request URI contains the <node-id> of the Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request deletes the entire Device representation in the SCL.

```
DELETE /applications?ipu=<ipu-uuid>&nw=<network-id>&node=<noted-id>
```

An Application is deleted by triggering a M2M DELETE request to the assisting GIP. The Request URI contains the <application-id> of the Application, the <node-id> of the associated Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy. The request deletes the entire Application representation in the SCL.

```
DELETE /applications?ipu=<ipu-uuid>&nw=<network-id>&node=<noted-id>&app=<application-id>
```

An Interface is deleted by triggering a M2M DELETE request to the assisting GIP. The Request URI contains the <interface-id> of the Interface, the <application-id> of the associated Application, the <node-id> of the associated Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request deletes the entire Interface representation in the SCL.

```
DELETE /applications?ipu=<ipu-uuid>&nw=<network-id>&node=<noted-id>&app=<application-id>&itf=<interface-id>
```

E.2.4 Data Field reporting

A "data" query parameter is defined to address a Data Field: data=<data-id>. A <data-id> is a unique identifier allocated to a Data Field by the micro GIP (e.g. data1). A <data-id> is relative to an Interface.

E.2.4.1 CREATE

A change of value (CoV) associated to a Data Field is reported by triggering a M2M CREATE request to the assisting GIP. The Request URI contains the <data-id> of the Data Field, the <interface-id> of the associated Interface, the <application-id> of the associated Application, the <node-id> of the associated Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request body contains the **value associated to the Data Field**. Data field value documents are not changed by the assisting GIP, leaving all flexibility to the interworking device designer to represent any protocol specificity or innovation.

```
CREATE /applications?ipu=<ipu-uuid>&nw=<network-id>&node=<noted-id>&app=<application-id>&itf=<interface-id>&data=<date-id>

  Data field value
```

EXAMPLE:

```
CREATE
/applications?ipu=zip.manufacturer1.com&nw=nw1&node=mac1;app=app1;itf=simplemetering;data=data1

  <real val="12"/>
```

Upon receiving this request, the assisting GIP selects an ETSI M2M <container> resource where the Data Field value will be saved. This is typically done through templates of ETSI M2M resource tree associated to oBIX contract of related M2M Area Network objects (Device, Application, Interface, etc.). The way used by the assisting GIP to create and to configure the ETSI M2M <container> resource when the related M2M Area Network object is created, and the way used by the assisting GIP to select the ETSI M2M <container> resource when the Data Field value is reported is out of scope of the present document.

For instance (in this example we assume that Interface descriptors are embedded in the Application descriptor), the selected ETSI M2M <container> resource can be a dedicated container:

- /applications/app1.mac1.nw1.zigbee.manufacturer1.com/containers/data1.simplemetering /contentInstances/latest

Or can be the Interface descriptor of the related M2M Area Network object:

- /applications/app1.mac1.nw1.zigbee.manufacturer1.com/containers/descriptor/contentInstances/latest

E.2.4.2 RETRIEVE

Not applicable.

E.2.4.3 UPDATE

Not applicable.

E.2.4.4 DELETE

Not applicable.

E.2.5 Method retargeting

Methods use the applicative retargeting mechanism defined in TS 102 690 [i.1]. Therefore the micro GIP exposes a resource, accessible to assisting GIP, to enable Methods retargeting from the assisting GIP to the micro GIP. When the micro GIP supports Methods processing, the micro GIP exposes an "apoc" resource: /apoc.

In addition, a "meth" query parameter is defined to address a Method: meth=<meth-id>. A <meth-id> is a unique identifier allocated to a Method by the micro GIP (e.g. toggle). A <meth-id> is relative to an Interface.

E.2.5.1 CREATE

A Method is invoked by triggering a M2M CREATE request to the micro GIP. The Request URI contains the <meth-id> of the Method, the <interface-id> of the associated Interface, the <application-id> of the associated Application, the <node-id> of the associated Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request body contains the **value of IN parameters**. The response body contains the value of **OUT parameters**. Method IN/OUT parameter documents are not changed by the assisting GIP, leaving all flexibility to the interworking device designer to represent any protocol specificity or innovation.

The processing of the Method is assumed synchronous and therefore associated to a single transaction of the underlying bearer (for instance a single CoAP request/response transaction).

```
CREATE /apoc?ipu=<ipu-uuid>&nw=<network-id>&node=<node-id>&app=<application-id>&itf=<interface-
id>&meth=<method-id>

Optional IN parameter

Response

Optional OUT parameter
```

EXAMPLE:

```
CREATE /apoc?ipu=zip.manufacturer1.com&nw=nw1&node=mac1;app=app1;itf=onff;meth=toggle
```

Upon receiving this request, the micro GIP processes the Method. The way used by the micro GIP to correlate the Method with a M2M Area Network object is out of scope of the present document.

E.2.5.1.1 IPU, Network, Device and Application level Methods

In some cases, the micro GIP can also support Methods addressing the IPU, the Network, a Device or an Application. In this case, the Method processing described in the previous clause is re-used. However some query parameters are not provided according to the targeted scope.

Application Methods: The Request URI contains the <meth-id> of the Method, the <application-id> of the associated Application, the <node-id> of the associated Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit.

Device Methods: The Request URI contains the <meth-id> of the Method, the <node-id> of the associated Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit.

Network Methods: The Request URI contains the <meth-id> of the Method, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit.

IPU Methods: The Request URI contains the <meth-id> of the Method and the <ipu-uuid> of the associated Interworking Proxy Unit.

E.2.5.2 RETRIEVE

Not applicable.

E.2.5.3 UPDATE

Not applicable.

E.2.5.4 DELETE

Not applicable.

E.2.6 Data Field retargeting

When supported by the micro GIP, the Data Field retargeting uses the same mechanism as defined for Methods retargeting.

E.2.6.1 CREATE

Not applicable.

E.2.6.2 RETRIEVE

A Data Field retrieval is invoked by triggering a M2M RETRIEVE request to the micro GIP. The Request URI contains the <data-id> of the Data Field, the <interface-id> of the associated Interface, the <application-id> of the associated Application, the <node-id> of the associated Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request body is empty. The response body contains the **Data field value**. Data field value documents are not changed by the assisting GIP, leaving all flexibility to the interworking device designer to represent any protocol specificity or innovation.

```
RETRIEVE /apoc?ipu=<ipu-uuid>&nw=<network-id>&node=<node-id>&app=<application-id>&itf=<interface-
id>&data=<data-id>

Response

  Data field value
```

E.2.6.3 UPDATE

A Data Field update is invoked by triggering a M2M UPDATE request to the micro GIP. The Request URI contains the <data-id> of the Data Field, the <interface-id> of the associated Interface, the <application-id> of the associated Application, the <node-id> of the associated Device, the <network-id> of the associated Network and the <ipu-uuid> of the associated Interworking Proxy Unit. The request body contains the **Data field value** (the content specification is out of scope of the present document). The response body may also contain a body.

```
UPDATE /apoc?ipu=<ipu-uuid>&nw=<network-id>&node=<node-id>&app=<application-id>&itf=<interface-
id>&data=<data-id>
  Data field value

Response

  Optional body
```

E.2.6.4 DELETE

Not applicable.

E.3 CoV configuration

M2M Area Network appliances need to be configured to properly set the change of value reporting settings. This is especially required to minimize battery consumption and network traffic for wireless sensors by indirectly setting the wakeup frequency of the wireless sensor. This is also required to save disk space on the SCL. When the micro GIP supports configuration of change of value reporting, the micro GIP exposes a "conf" resource: /conf.

This resource, exposed by the micro GIP, is used by the assisting GIP to configure the micro GIP, by sending a XML configuration document: <conf>.

The assisting GIP will attempt to UPDATE the <conf> document to this URI. Due to the limited resources, not all micro GIP are expected to support such reporting configuration file. In this case the micro GIP rejects the attempt of the assisting GIP to UPDATE the reporting configuration document with error code 404 (Not Found). Such simple micro GIP may have their own internal logic to configure reporting. The assisting GIP has to be prepared to receive unsolicited reports from the micro GIP.

E.3.1 XML <conf> element

The <conf> element is the root element, sent by the assisting GIP, to configure the micro GIP. The element is associated to a <ipu-uuid> IPU (i.e. a given micro GIP can handle several IPU):

```
<conf ipu="<ipu-uuid>">
  ...
</conf>
```

E.3.2 XML <itf> element

The <conf> element contains a list of Interfaces for which a CoV reporting is configured. Each Interface is associated with a <itf> element and is identified through a <filter>. A <filter> allows identifying a particular type of Interface. The type of Interface expressed through a <filter>, can apply to the entire M2M Area Network (e.g. */itf/simplemetering), can apply to a given type of Application on the M2M Area Network (e.g. */app/mainspoweroutlet/itf/simplemetering), can apply to a given type of Device on the M2M Area Network (e.g. dev/partnum1/*/itf/simplemetering), or can apply to any other combination supported by the micro GIP. The exact syntax used to express a <filter> is not defined by the present document and is implementer dependent:

```
<itf filter="<filter>">
    ...
</itf>
```

E.3.3 XML <data> element

The <itf> element contains a list of Data Fields for which a CoV reporting is configured. Each Data Field is associated with a <data> element and is identified through a <data-id>:

```
<data id="<data-id>" .../>
```

E.3.3.1 CoV configuration

When the Data Field is associated with a static value (e.g. an identifier, a serial number, a software version, etc.), the <data> element does not define additional attribute, and the Data Field value is reported only once by the micro GIP:

```
<data id="<data-id>"/>
```

When the Data Field is associated with a dynamic value (e.g. a sensor binary output, a sensor analog output, etc.), the <data> element is associated with the following attributes: minInt (optional), maxInt (mandatory), minCOV (mandatory for analog/continuous data type):

```
<data id="<data-id>" [minInt="<duration>"] maxInt="<duration>" [minCOV="<integer or real>"]/>
```

The time interval between two reports needs to be always greater than minInt, and smaller than maxInt. <duration> are typically expressed as ISO 8601 [i.6] durations. When minInt is not specified, no constraint applies to the minimal interval. A minimal COV is mandatory for attribute with an analog/continuous data type (e.g. an analog output). The exact format to express a minCoV is out of scope of the present document.

E.3.4 CoV configuration XML schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://uri.etsi.org/m2m"
xmlns:tns="http://uri.etsi.org/m2m">
    <element name="conf" type="tns:ConfigurationType" />
    <complexType name="ConfigurationType">
        <sequence>
            <element name="itf" type="tns:InterfaceType" minOccurs="1" maxOccurs="unbounded"/>
        </sequence>
        <attribute name="ipu" type="string" use="required"/>
    </complexType>

    <complexType name="InterfaceType">
        <sequence>
            <element name="data" type="tns:DataType" minOccurs="1" maxOccurs="unbounded"/>
        </sequence>
        <attribute name="filter" type="string" use="required"/>
    </complexType>

    <complexType name="DataType">
        <attribute name="id" type="string" use="required"/>
        <attribute name="minInt" type="duration" use="optional"/>
        <attribute name="maxInt" type="duration" use="optional"/>
        <attribute name="minCOV" type="integer" use="optional"/>
    </complexType>
</schema>
```

E.3.5 Example

Example of configuration, sent to a micro GIP, for a given type of Interface (simplemetering.manufacturer1.com) and a given IPU (zigbee.manufacturer1.com) without constraint on the type of the Application or the type of the Device:

```
UPDATE /conf

<?xml version="1.0" encoding="UTF-8"?>
<m:conf ipu="zigbee.manufacturer1.com" xmlns:m="http://uri.etsi.org/m2m">
    <itf filter="itf/simplemetering">
        <data id="0x0000" maxInt="PT5M"/>
        <data id="0x0400" maxInt="PT5M"/>
    </itf>
</m:conf>
```

Example of configuration for a given type of Interface (simplemetering.manufacturer1.com) and a given IPU (zigbee.manufacturer1.com) limited to a given type of Device (partnum1.manufacturer1.com):

```
UPDATE /conf

<?xml version="1.0" encoding="UTF-8"?>
<m:conf ipu="zigbee.manufacturer1.com" xmlns:m="http://uri.etsi.org/m2m">
    <itf filter="dev/partnum1/*/itf/simplemetering">
        <data id="0x0000" maxInt="PT10S"/>
        <data id="0x0400" maxInt="PT10S"/>
    </itf>
</m:conf>
```

Example of configuration that defines a default configuration for a given type of Interface (simplemetering.manufacturer1.com), except for a given type of Device (partnum1.manufacturer1.com) which has a specific configuration:

```
UPDATE /conf

<?xml version="1.0" encoding="UTF-8"?>
<m:conf ipu="zigbee.manufacturer1.com" xmlns:m="http://uri.etsi.org/m2m">
    <itf filter="itf/simplemetering">
        <data id="0x0000" maxInt="PT5M"/>
        <data id="0x0400" maxInt="PT5M"/>
    </itf>
    <itf filter="dev/partnum1/*/itf/simplemetering">
        <data id="0x0000" maxInt="PT10S"/>
        <data id="0x0400" maxInt="PT10S"/>
    </itf>
</m:conf>
```

E.4 dId over USB

The dId over USB uses mechanisms defined in annex D (CoAP Binding for M2M REST Resources). However, as defined in the following clauses, the USB binding is constrained to take into account specificities introduced by the USB bearer.

E.4.1 Base URI

The USB interface implies a point to point interface between the micro GIP and the assisting GIP (no DNS resolution or IP routing is requested). Therefore URI authorities, representing the micro GIP and the assisting GIP, can be associated to well-known aliases.

Two URI authority aliases are defined, to be used in CoAP requests:

- Micro GIP authority: gip.lan
- Assisting GIP authority: proxy.lan

Sample

Sample of CoAP request sent by the micro GIP to the assisting GIP:

```
POST coap://proxy.lan/applications?ipu=zigbee.manufacturer1.com

  <obj is="ipu.manufacturer1.com"/>
```

Sample of CoAP request sent by the assisting GIP to the micro GIP:

```
POST coap://gip.lan/apoc?ipu=zip.manufacturer1.com&nw=nw1&node=mac1;app=app1;itf=onff;meth=toggle
```

E.4.2 Transport over serial link

The serial link is typically provided by USB dongles, which presents at least an interface descriptor advertising one IN data bulk endpoint and one OUT data bulk endpoint.

Sample of USB Descriptor

The following set of descriptors is advertised by FTDI chips commonly used by dongle manufacturers.

Table E.1: Sample of USB descriptor

Device Descriptor	bLength	18		
	bDescriptorType	1		
	bcdUSB	2.00		
	bDeviceClass	0 (DefinedatInterfacelevel)		
	bDeviceSubClass	0		
	bDeviceProtocol	0		
	bMaxPacketSize0	8		
	idVendor	0x0403(FutureTechnologyDevicesInternational,Ltd)		
	idProduct	0x6001 (FT232USB-Serial(UART)IC)		
	bcdDevice	6.00		
	iManufacturer	1		
	iProduct	2		
	iSerial	3		
	bNumConfigurations	1		
Configuration Descriptor	bLength	9		
	bDescriptorType	2		
	wTotalLength	32		
	bNumInterfaces	1		
	bConfigurationValue	1		
	iConfiguration	0		
	bmAttributes	0xa0 (BusPowered)		
	RemoteWakeup			
	MaxPower	90mA		
Interface Descriptor	bLength	9		
	bDescriptorType	4		
	bInterfaceNumber	0		
	bAlternateSetting	0		
	bNumEndpoints	2		
	bInterfaceClass	255 (VendorSpecificClass)		
	bInterfaceSubClass	255 (VendorSpecificSubclass)		
	bInterfaceProtocol	255 (VendorSpecificProtocol)		
	iInterface	2		
Endpoint Descriptor	bLength	7		
	bDescriptorType	5		
	bEndpointAddress	0x81	**Endpoint Number**	**EP1**
			Direction	**IN**
	bmAttributes	2	**TransferType**	**Bulk**
			SynchType	**None**
			UsageType	**Data**
	wMaxPacketSize	0x0040 (1x 64 bytes)		
	bInterval	0		
Endpoint Descriptor	bLength	7		
	bDescriptorType	5		
	bEndpointAddress	0x02	**Endpoint Number**	**EP2**
			Direction	**OUT**
	bmAttributes	2	**TransferType**	**Bulk**
			SynchType	**None**
			UsageType	**Data**
	wMaxPacketSize	0x0040 (1x 64 bytes)		
	bInterval	0		

Serial data transported over USB bulk transfer endpoints are already protected by a CRC16, therefore only a framing and a multiplexing mechanisms are required to transport CoAP data units:

- The framing mechanism allows detecting CoAP data units on the asynchronous link.
- The multiplexing mechanism allows transporting data units associated to other protocols over the asynchronous link (these protocols are out of scope of the present document).

Framing and multiplexing:

- SLIP (RFC 1006 [i.7]) is used to frame CoAP data units over USB (UDP/IP headers are not included). A 1-byte header is added in front of the CoAP data unit for multiplexing. The 0x00 header value is defined for CoAP data units. Any other values are reserved for future use.

The protocol stack is illustrated on figure E.2.

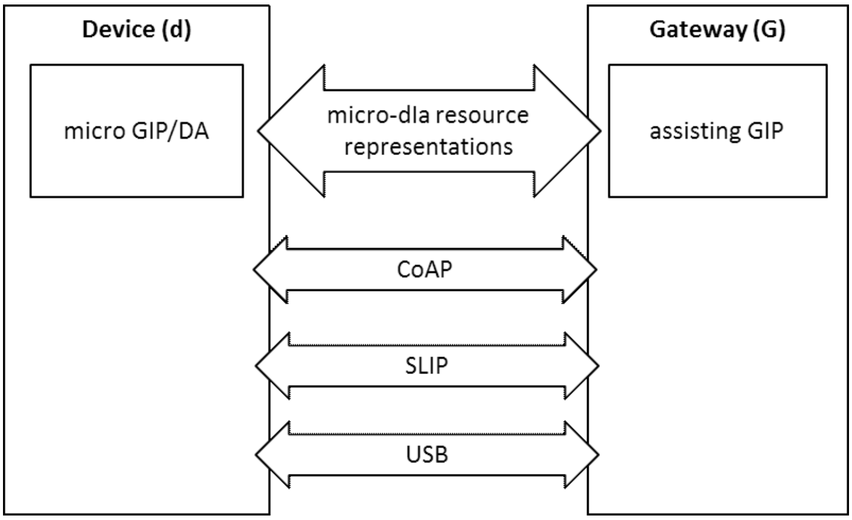

Figure E.2: Protocol stack for dId over USB

E.5 dId over IP

The dId binding over IP uses mechanism defined in TS 102 921 [i.2], annex D (CoAP Binding for M2M REST Resources).

Annex F: Bibliography

ZigBee® document 053474r17: "ZigBee specification release 17, ZigBee Technical Steering Committee".

ZigBee® document 053820r18: "ZigBee Bridge Device Specification, ZigBee Gateway Working Group".

ZigBee® document 064699r12: "ZigBee Commissioning Cluster, Application Framework Group".

ZigBee® document 075123r02: "ZigBee Cluster Library Specification, Application Framework Group".

History

Document history		
V1.1.1	February 2014	Publication