INTERNATIONAL STANDARD

ISO/IEC 17811-2

First edition
2015-02-15

Information technology — Device control and management —

Part 2: Specification of Device Control and Management Protocol

Technologies de l'information — Commande et gestion de périphériques —

Partie 2: Spécifications du protocole de commande et gestion de périphériques

Reference number
ISO/IEC 17811-2:2015(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT), see the following URL: Foreword — Supplementary information.

ISO/IEC 17811-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology* SC 6, *Telecommunications and information exchange between systems*.

Introduction

This International Standard provides the architecture for device control and management (DCM). DCM can support the various control and management services, regardless of the network protocols or interfaces. DCM is composed of two protocols; device control and management protocol (DCMP) and reliable message delivery protocol (RMDP).

This International Standard, ISO/IEC 17811, consists of the following parts:

— Part 1: Architecture

— Part 2: Specification of Device Control and Management Protocol

— Part 3: Specification of Reliable Message Delivery Protocol

ISO/IEC 17811-1 describes the architecture of DCM, which includes definition, general concept, requirements, design principles, service scenarios for device management control, and management.

ISO/IEC 17811-2 specifies the Device Control and Management Protocol (DCMP), which includes the functional entities, protocol operations, message structure, and detailed parameter format associated with DCMP.

ISO/IEC 17811-3 specifies the Reliable Message Delivery Protocol (RMDP), which includes the interworking with DCMP, protocol operations, and message structure associated with RMDP.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this International Standard may involve the use of a patent concerning the message structure of DCMP given in Clause 7.

ISO and IEC take no position concerning the evidence, validity, and scope of this patent right.

The holder of this patent right has assured the ISO and IEC that he is willing to negotiate licences either free of charge or under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with ISO and IEC. Information may be obtained from:

Patent Holder: Electronics and Telecommunications Research Institute (ETRI)

Address: 138 Gajeongno, Yuseong-gu, Daejeon, 305-700, Korea

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO (www.iso.org/patents) and IEC (http://patents.iec.ch) maintain on-line databases of patents relevant to their standards. Users are encouraged to consult the databases for the most up to date information concerning patents.

Information technology — Device control and management —

Part 2: Specification of Device Control and Management Protocol

1 Scope

This part of ISO/IEC 17811 provides the specification of Device Control and Management Protocol (DCMP), which is an application-layer protocol used to control and manage the various devices. DCMP supports the device and network status information retrieval, device initialization, firmware and software update, file transmission, and so on. This part of ISO/IEC 17811 specifies the protocol operations and message structure of DCMP.

The network security is out of scope in this part of ISO/IEC 17811. However, the security services can be necessary according to applications of DCMP. DCMP can suffer from many network specific threats. To countermeasure those threats, some security mechanism can be deployed.

2 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

2.1
device control and management
DCM
purposed to control and manage the various smart devices

Note 1 to entry: For this purpose, DCM is composed of the two protocols; Device Control and Management Protocol (DCMP) and Reliable Message Delivery Protocol (RMDP).

[SOURCE: ISO/IEC 17811-1]

2.2
device control and management protocol
DCMP
used to perform various management operations which are categorized into information retrieval, control, diagnostic, and debugging

[SOURCE: ISO/IEC 17811-1]

2.3
reliable message delivery protocol
RMDP
used to provide uniform and reliable message delivery among devices regardless of the underlying network protocols or interfaces

[SOURCE: ISO/IEC 17811-1]

2.4
administrative domain
represents a network area where a single administrator can configure and manage a network with the same policy

[SOURCE: ISO/IEC 17811-1]

2.5
device management server
DMS
used to keep track of the various device information and also to manage the devices in an administration domain

Note 1 to entry: There may be one DMS in an administrative domain, if needed.

[SOURCE: ISO/IEC 17811-1]

3 Abbreviation

The following abbreviations are used in this document.

DCMP Device Control and Management Protocol

DCM Device Management Architecture and Protocol

DMS Device Management Server

RMDP Reliable Message Delivery Protocol

NTP Network Time Protocol

UUID Universally Unique Identifier

UPnP Universal Plug and Play

4 Overview

The DCMP is a protocol used to control and manage a variety of smart devices in the network. The DCMP messages are exchanged between different devices or between device and DMS.

The DCMP operates over RMDP for reliable message delivery. In the networking perspective, RMDP provides one or more devices with an interface to the network. That is, a group of devices are connected to an RMDP node via an internal API interface or a network, and the RMDP performs the reliable delivery of DCMP messages to the different RMDP nodes which are also connected to other devices. For this purpose, RMDP maintains the mapping information between DCM device identifier and physical network identifier such as IP address and port number. After RMDP retrieves the target node information, DCMP messages can be exchanged over RMDP.

Figure 1 gives a general example of DCM operations between a device and DMS. Those operations are divided into RMDP initialization, DCMP initialization, and management service.

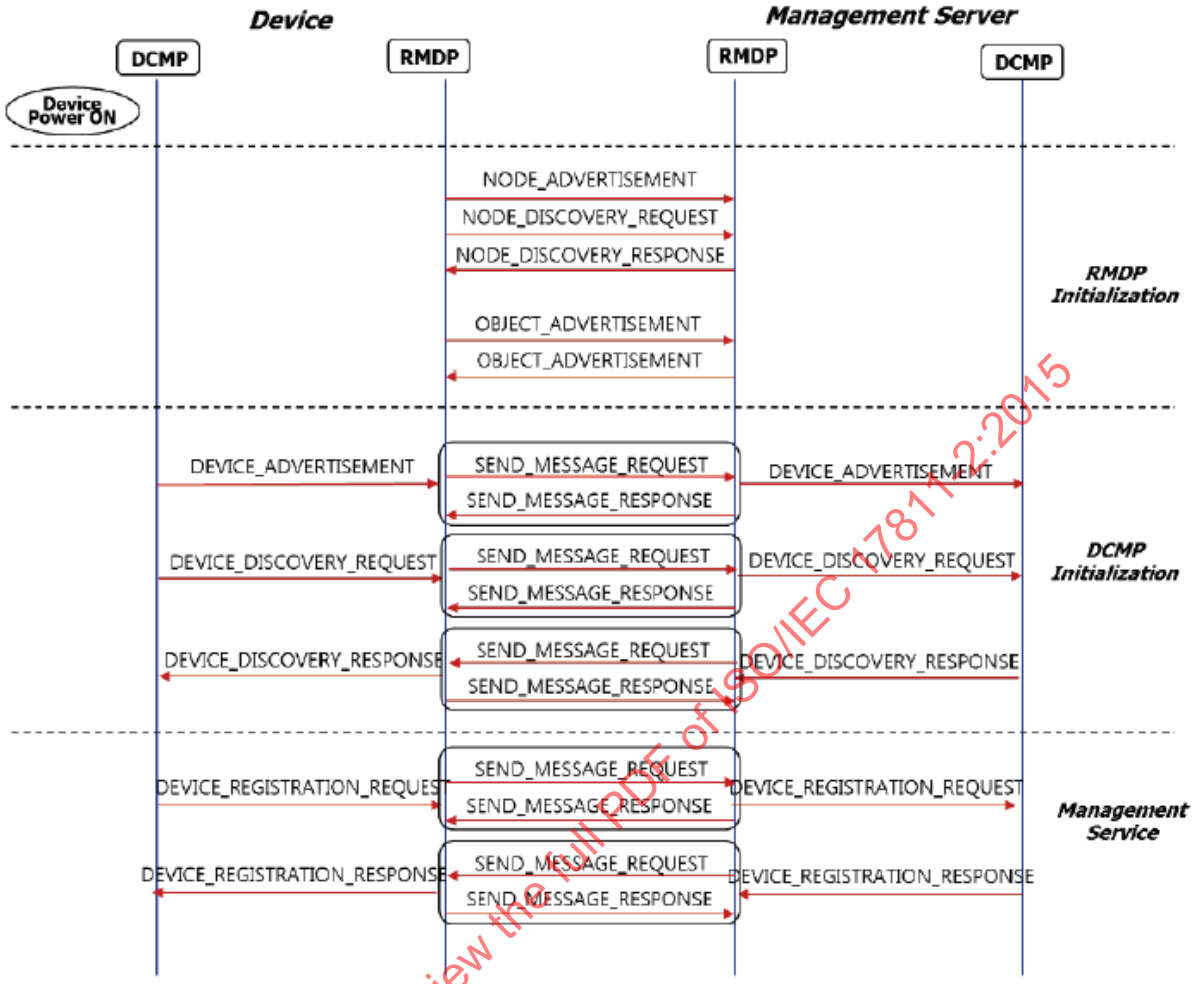


Figure 1 — DCM Operations

The RMDP manages the mapping information between the device identifier and physical network identifier. For an advertisement of the physical network identifier information in the RMDP initialization phase, a "NODE_ADVERTISEMENT" and "NODE_DISCOVERY_REQUEST/RESPONSE" messages are broadcast by the RMDP. To advertise the device identifier information, an OBJECT_ADVERTISEMENT message is then broadcast by the RMDP. The physical address information of the DMS can be added in the RMDP module of a device using the API of RMDP. The RMDP module of the device sends a NODE_ADVERTISEMENT message using a broadcast to the other devices connected in the local network, but the RMDP module of the device uses a unicast when sending the NODE_ADVERTISEMENT message to the DMS. The RMDP module of the DMS can manage the physical address information of the device using the socket information, that is, the IP address and port information of the access point (AP) or router. In the DCMP initialization phase, the DCMP modules of DMS and device can retrieve the basic information of the concerned devices (e.g. device name, device ID and device type) by using the "DEVICE_ADVERTISEMENT" and "DEVICE_DISCOVERY_REQUEST/RESPONSE" messages. In these processes, the DCMP module asks its RMDP module with the device ID of the target (corresponding) device by using a DCMP message, and then the RMDP module will transmit the received DCMP message to the target RMDP module of the target device. In the management service phase, a device can register its basic information (e.g., model name, model number and serial number) on the DMS server by using the "DEVICE_REGISTRATION_REQUEST/RESPONSE" message of DCMP.

The protocol operations of DCMP are classified as follows:

— Device Discovery;

— Device Advertisement;

— Device Information Retrieval;

— Device Control;

— Event Notification;

— Event Subscription;

— Get File Information;

— File Download;

— File Upload;

— Apply;

— Device Registration; and

— Service Registration.

5 Protocol Operation

5.1 Device Discovery

For device discovery, a DEVICE_DISCOVERY_REQUEST and DEVICE_DISCOVERY_RESPONSE messages are exchanged between devices, as shown in Figure 2. A source device broadcasts a DEVICE_DISCOVERY_REQUEST message to the target devices. In response to the DEVICE_DISCOVERY_REQUEST message, all devices which fit into the requested information shall respond with a DEVICE_DISCOVERY_RESPONSE message.

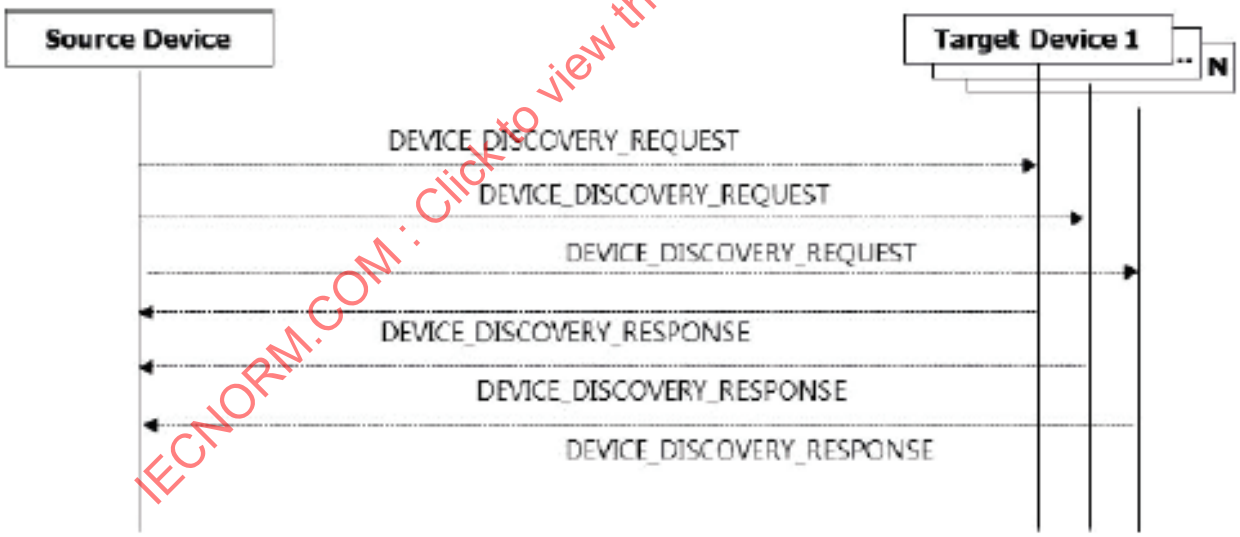


Figure 2 — Device Discovery Operation

The device discovery operation is performed with a two-message transaction. This operation requires a request message at source and a response message at the destination. When a response message is not received within a specific time interval, the source may cancel the transaction or re-issue the transaction.

5.2 Device Advertisement

The device advertisement operation can be used to inform device's plug-in or plug-out, as shown in Figure 3. The associated device advertisement transaction is one-way transaction. This means that only one message is required to finish a transaction, and any response message is not required.

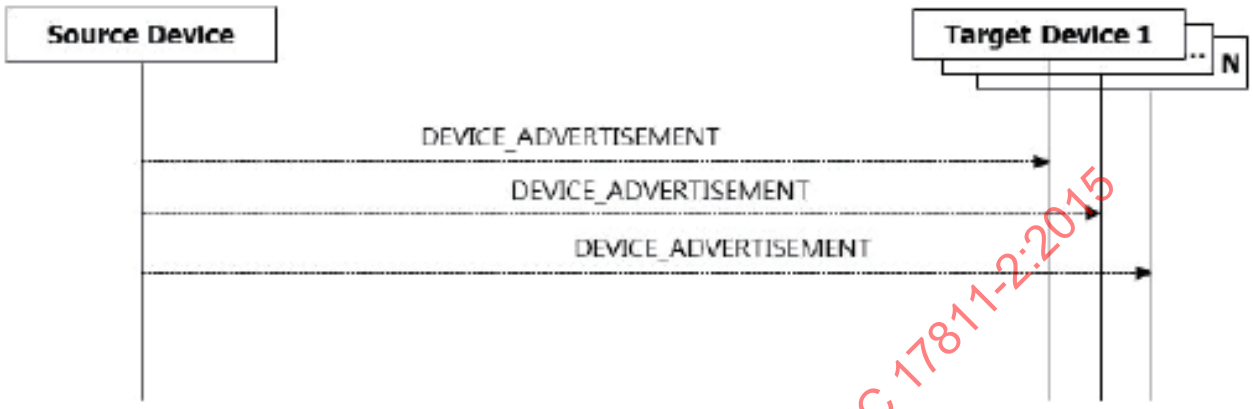


Figure 3 — Device Advertisement Operation

5.3 Device Information Retrieval

The device information retrieval operation can be used when a device needs to know the various device information such as device ID, device name, device property, device status and so on. The volume of the device information is various and depends on the type of device and its capability. To accommodate different level of devices, we may categorize the relevant information into device basic information, device configuration, and system and network information. The basic information represents the fixed information for all devices, whereas the other information is variable for each device.

The operation of device information retrieval is composed of two-message transaction, as shown in Figure 4. This operation requires a request message at source and a response message at destination. When a response message is not received within a specific time interval, the source may cancel the transaction or re-issue the transaction.

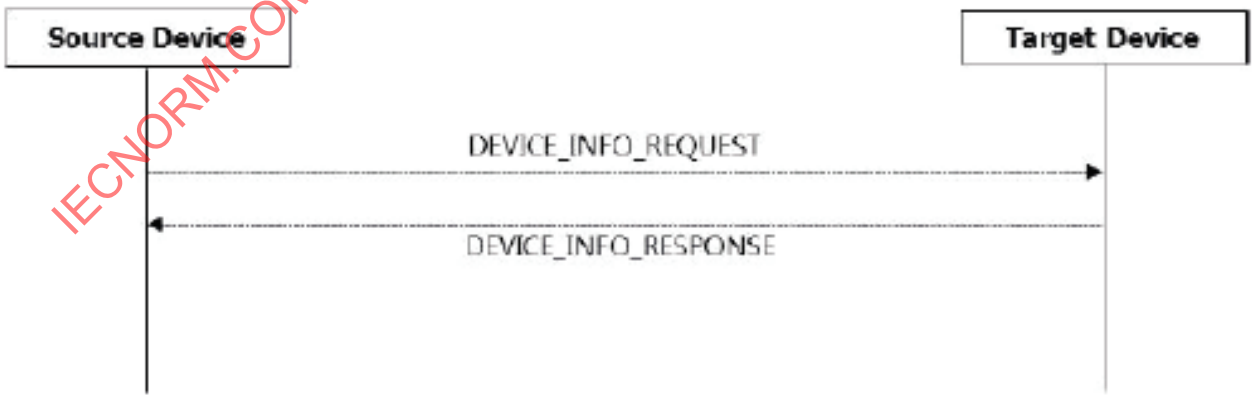


Figure 4 — Device Information Retrieval Operation

5.4 Device Control

Device control is used to perform device specific operations. For device control, a source device shall provide the control code and parameters associated with the device control operation. When a

target device receives a DEVICE_CONTROL_REQUEST message, it checks if it the message contains a valid control code for the device and if the control code can be supported. If so, it executes the control operation requested and returns the result to the source device with a DEVICE_CONTROL_RESPONSE message, as shown in Figure 5.

The operation of device control is performed with a two-message transaction. This operation requires a request message at source and a response message at the destination. When a response message is not received within a specific time interval, the source device may cancel the transaction or re-issue the transaction.

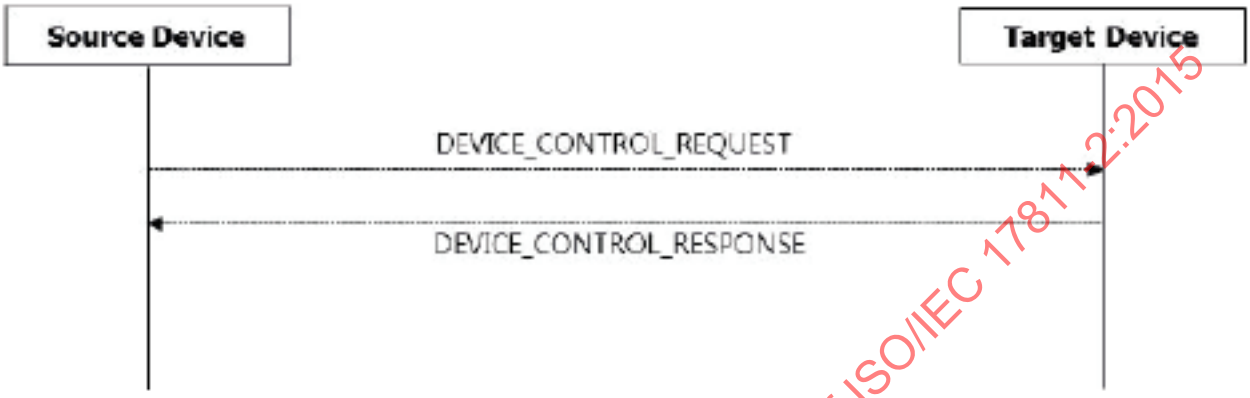


Figure 5 — Device Control Operation

5.5 Event Notification

When an event occurs in a device, the event can be reported to the interested devices by EVENT_NOTIFICATION message, as shown in Figure 6. Event notification transaction is one-way transaction. This means that only one message is required to finish a transaction, and any response message is not required.

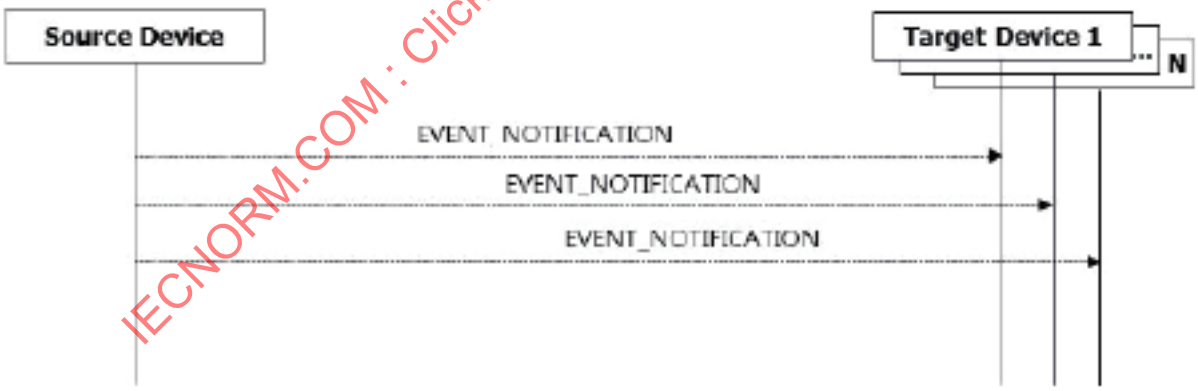


Figure 6 — Event Notification Operation

5.6 Event Subscription

When an event occurs in a device, the event can be reported to all devices in a network. However, whenever an event is broadcast to all devices in network, the event messages are overwhelmed in the network. So, the event information shall be reported to only the interested devices. For this purpose, the event handling related operations include event notification as well as the associated event subscription/un-subscription operations. This is helpful to reduce traffic overhead within a local network.

The operation of event subscription is a two-message transaction, as shown in Figure 7. This operation requires a request message at source and a response message at destination. When a response message is not received within a specific time interval, the source may cancel the transaction or re-issue the transaction.

Figure 7 — Event Subscription Operation

5.7 Get File Information

The file upload and download capability is the essential functions for device management, since the software or firmware update requires the updated file to be transferred to the target device. Sometimes it is useful to send quite a large size of message instead of using a simple message transfer. So, the file upload/download operation is defined as a part of common device operations. The processing of the files is various in a large system. So, we classify each file by a file type and provide this information with the file.

The operation of 'get file information' is a two-message transaction, as shown in Figure 8. This operation requires a request message at source and a response message at destination. When a response message is not received within a specific time interval, the source device may cancel the transaction or re-issue the transaction.

Figure 8 — Get File Information Operation

5.8 File Download

The file download capability is an essential function for device management since the software or firmware update requires the updated file to be transferred to the target device. Sometimes it is useful to send quite a large size of message instead of using a simple message transfer. So, file upload/download operation is defined as a part of common device operations.

The file download transaction can be used to get some file in a remote device and totally the three messages are required, as shown in Figure 9. First, a source device requests the file download by using a GET_FILE_REQUEST message. Then, a destination device decides if the request is accepted or rejected. When the request is accepted, the destination device shall respond with a GET_FILE_RESPONSE message containing a SUCCESS indication code. However, if the destination device rejects the file transfer, the 'get file' transaction is terminated.

After finishing the file transfer, the destination device sends a GET_FILE_RESULT message with a SUCCESS indication code. When a file transfer is aborted by some reasons, the destination device sends the GET_FILE_RESULT message with an appropriate error code.

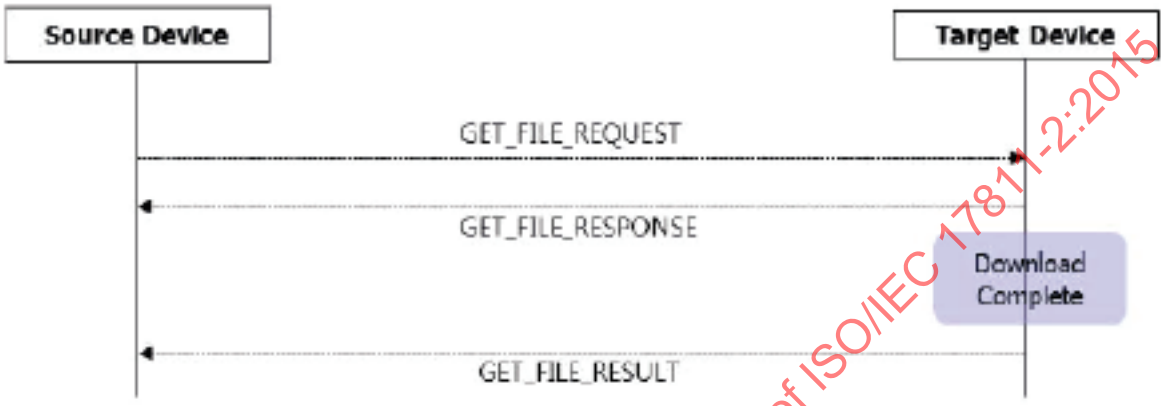


Figure 9 — File Download Operation

5.9 File Upload

The file upload capability is an essential function for device management since the software or firmware update requires the updated file to be transferred to the target device. Sometimes it is useful to send quite a large size of message instead of using a simple message transfer. So, file upload/download operation is defined as a part of common device operations.

The file upload transaction can be used to send some file from the source device to the target device and totally the three messages are required, as shown in Figure 10. First, a source device requests the file upload by using a PUT_FILE_REQUEST message. Then, a destination device decides if the request is accepted or rejected. When the request is accepted, the destination device shall respond with a PUT_FILE_RESPONSE message containing a SUCCESS indication code. However, if the destination device rejects the file transfer, a 'put file' transaction is terminated.

After finishing the file transfer, the source device sends a PUT_FILE_RESULT message with a SUCCESS indication code. When a file transfer is aborted by some reasons, the source device sends the PUT_FILE_RESULT message with an associated error code.

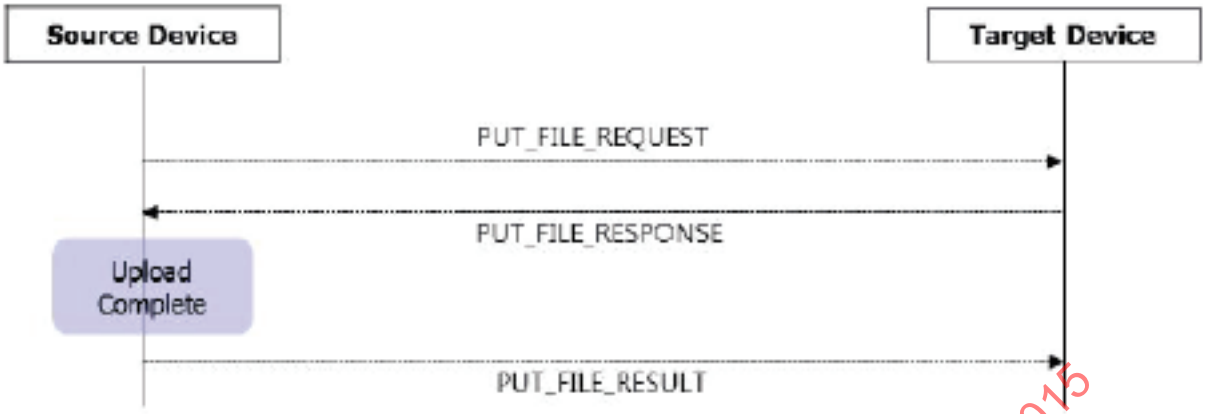


Figure 10 — File Upload Operation

5.10 Apply

Apply transaction can be used to request a critical control action to a device and get the result, as shown in Figure 11. A source device requests an APPLY_REQUEST message specifying the requested operation such as firmware update, reboot, configuration, etc. Then a destination device decides if the request will be accepted or rejected. When the requested operation is accepted, then the device immediately replies with an APPLY_RESPONSE message and performs the requested action. When the requested action is finished, then an apply result message is sent back to the source device.

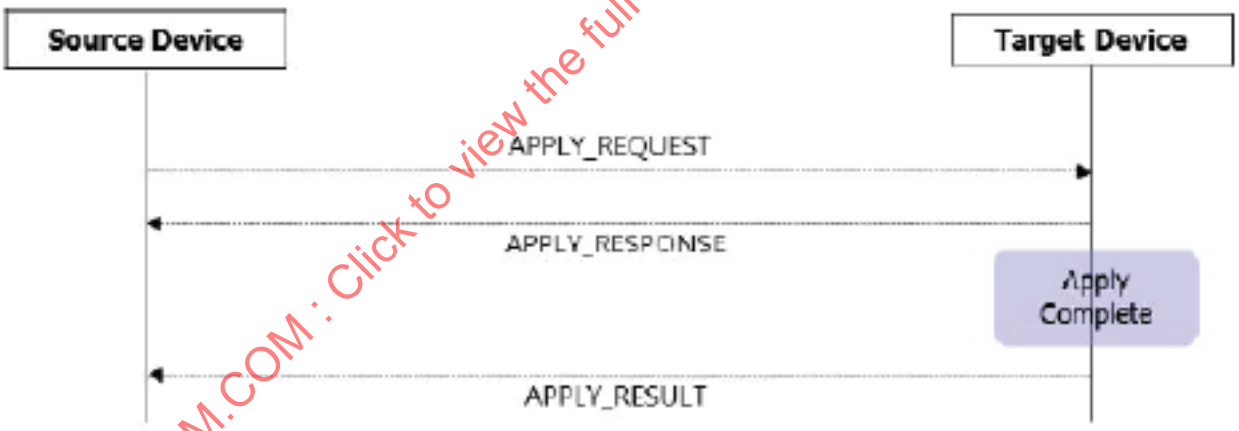


Figure 11 — Apply Operation

5.11 Device Registration

Device registration message can be used when a device wants to register its own information to the DMS. The operation of device registration is a two-message transaction, as shown in Figure 12. This operation requires a request message at source and a response message at destination. When a response message is not received within a specific time interval, the source device may cancel the transaction or re-issue the transaction.

Figure 12 — Device Registration Operation

5.12 Service Registration

Service registration message can be used when a source device wants to register the service-specific information with the DMS. The operation of service registration is a two-message transaction, as shown in Figure 13. This operation requires a request message at source and a response message at destination. When a response message is not received within a specific time interval, the source may cancel the transaction or re-issue the transaction.

Figure 13 — Service Registration Operation

6 Messages

6.1 DCMP Message Structure

Basically DCMP message is composed of header and payload. Message header includes the information about 'Start signal', 'Source device ID', 'Target device ID', 'Message Type' and so on. Especially 'Message Type' defines the various functions, which can be provided by DCMP and the payload message is determined according to the 'Message Type'. Figure 14 and Table 1 shows the more details about header message structure of DCMP.

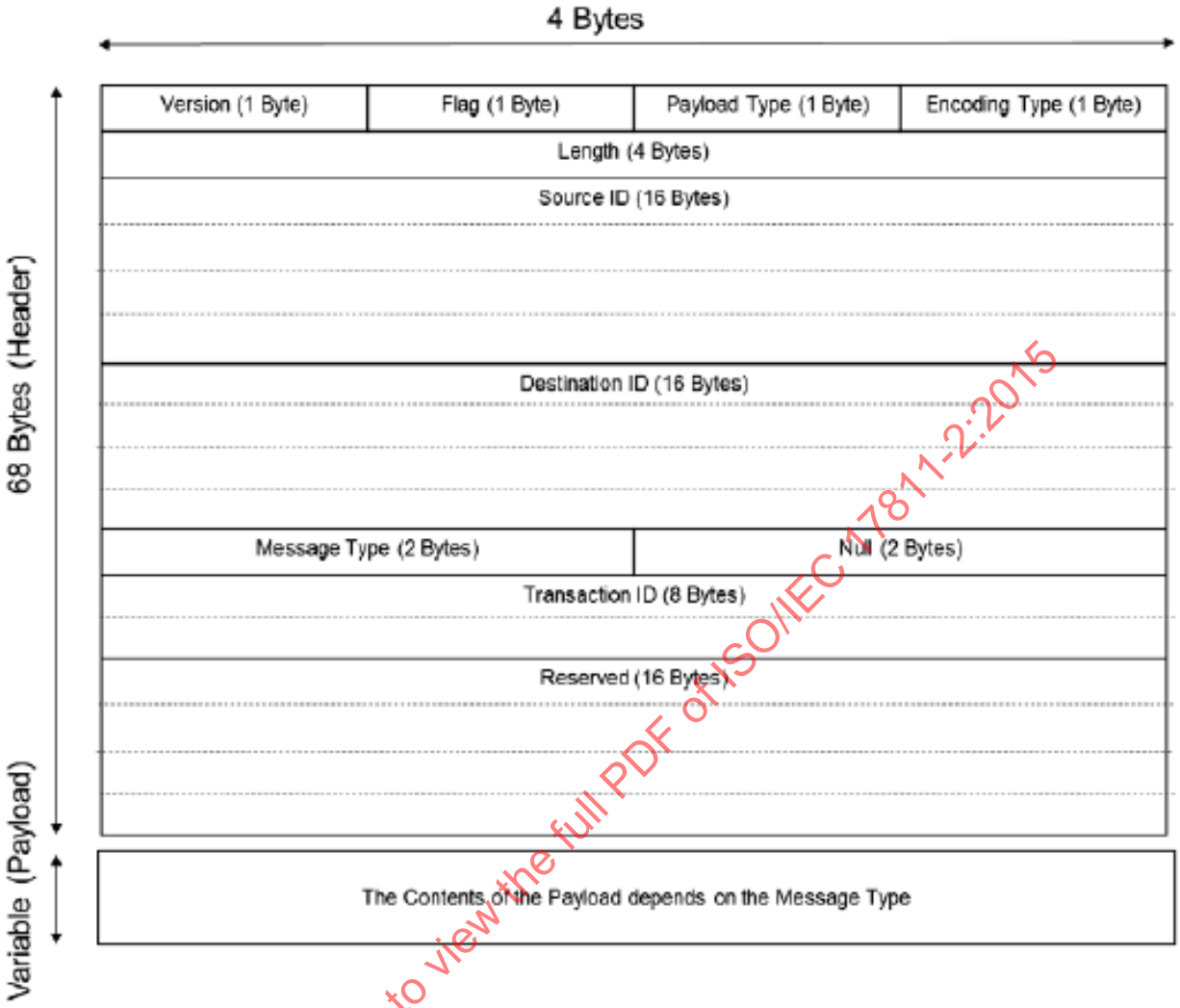


Figure 14 — DCMP Message Header Structure

Table 1 — Description of DCMP Message Header Structure

Field Name	Size (Byte)	Description
Version	1	— Protocol Version — 0x01
Flag	1	— Urgent flag — Emergency: 0x01/Normal: 0x00
Payload Type	1	— Describes Payload Type (removes table) 0x01 Binary 0x02 XML 0x03 ~ 0xFF Reserved

Table 1 *(continued)*

Field Name	Size (Byte)	Description
Encoding Type	1	— Describes Encoding Type 0x01 ASCII 0x02 UNICODE 0x03 UTF-8 0x04 UTF-16 0x05 ~ 0xFF Reserved
Length	4	— Message Size (Including Header)
Source ID	16	— Source Device ID — Random generation using the UUID — Can be Installed when a device is manufactured
Destination ID	16	— Destination device ID (UUID)
Message Type	2	— Message type Code for the DCMP operations
TransactionID	8	— Transaction Identifier — The value will be NTP timestamp — time_t 8bytes
Optional Header/ DATA	16	— Reserved

6.2 Messages according to the Operations

Major operations, which can be provided by DCMP, are Information (ex: device properties and status) providing, remote device control, remote maintenance and device self-organization. These operations can be classified by 'Message Type' and more details about 'Message Type' are shown in Table 2.

In case that the REQUEST message is transmitted and the corresponding RESPONSE message is not returned within an appropriate time, the indication that the REQUEST was not successful will be delivered to the user without retransmission. The receiver who received the REQUEST shall respond if the transaction ID is valid.

Transaction ID in RESPONSE message will be same to the transaction ID of the corresponding REQUEST message.

Table 2 — DCMP Message and Message type

Operation	Message	Message Type	Description
Device Discovery	DEVICE_DISCOVERY_REQUEST DEVICE_DISCOVERY_RESPONSE	0x1111 0x1112	— Device Searching
Device Advertise-ment	DEVICE_ADVERTISEMENT	0x1121	— Plug in/out
Device Info	DEVICE_INFO_REQUEST DEVICE_INFO_RESPONSE	0x2011 0x2012	— Device Description — Device/Network State
Device Control	DEVICE_CONTROL_REQUEST DEVICE_CONTROL_RESPONSE	0x2111 0x2112	— Device Control

Table 2 *(continued)*

Operation	Message	Message Type	Description
Event	EVENT_NOTIFICATION	0x2FF1	— Report the changes of device status (Device/Network) — SENSOR Data — User defined Event
Event Subscription	EVENT_SUBSCRIPTION_REQUEST EVENT_SUBSCRIPTION_RESPONSE	0x2F11 0x2F12	— Event Subscription/ Cancelation
Get File Info	GET_FILEINFO_REQUEST GET_FILEINFO_RESPONSE	0x2211 0x2212	— File Info Request
Get File	GET_FILE_REQUEST GET_FILE_RESPONSE GET_FILE_RESULT	0x2221 0x2222 0x2223	— Get file Request/ Response — Transaction Result
Put File	PUT_FILE_REQUEST PUT_FILE_RESPONSE PUT_FILE_RESULT	0x2231 0x2232 0x2233	— Put file Request/ Response — Transaction Result
Apply	APPLY_REQUST APPLY_RESPONSE APPLY_RESULT	0x2311 0x2312 0x2313	Update, Rollback, Execution of FILE (Firmware, Configuration, etc.)
Device Registration	DEVICE_REGISTRATION_REQUEST DEVICE_REGISTRATION_RESPONSE	0x3111 0x3112	Device Registration (or Cancel)
Service Registration	SERVICE_REGISTRATION_REQUEST SERVICE_REGISTRATION_RESPONSE	0x3121 0x3122	User Registration/ Delete/Add
User Defined Transaction	USER Defined Operation	0x8000 ~ 0xFFFF	User defined transaction

6.3 Error Types of DCMP

DCMP defines various error codes, which can be occurred during the message operation or specific service. Error codes can be classified into three types; 'Common Error', 'Device Specific Error' and 'User Defined Error'. More details of each error type are listed below and the description of error codes is shown in Table 3.

Common Error

— Using the value from 0x0000 to 0x0FFF

— Common Error defines 4 different types of error

  — Message Error: errors of message header and payload

  — Transaction Error: transaction-related errors

  — File transmission Error: errors that occurred in file transmission

  — Apply Error: errors that occurred in apply action

Device Specific Error

— Using the value from 0x1000 to 0xEFFF

— DCM defines device specific errors according to the device type.

User Defined Error

— Using the value from 0xF000 to 0xFFFF

— Defines user defined errors

Table 3 — Error Types

Error Type (Hex)			Description
0000 ~ 0FFF (Common Error)	0000 ~ 09FF (Message Error)	0000	Success
		0100	Header Error
		0100	Header Start Message Error
		0101	Header Version Message Error
		0102	Header Length Message Error
		0103	Header Source ID Error
		0104	Header Destination ID Error
		0105	Header OP Code Error
		0106	Header TransactionID Error
		0107	Header CRC Error
		0108	Header End Message Error
		0200	Payload Error
		0201	No Payload
		0202	Whole Payload data can't be received
		0203	Payload size Mismatch between the received data and the size which is described in the Header
		0204	Payload Syntax Error
	0AXX (Transaction Error)	0000	Success
		0A01	Other operation is running
		0A02	Transaction is stopped
		0A03	Deadlock
		0A04	Transaction Time over
	0BXX (File transmission Error)	0000	Success
		0B01	Command not implemented
		0B02	Bad sequence of commands
		0B03	Syntax error, command unrecognized
		0B04	User not logged in
		0B05	Need account for storing files
		0B06	File unavailable (e.g., file not found, no access)
		0B07	Storage allocation exceeded
		0B08	Illegal file name

Table 3 — *(continued)*

Error Type (Hex)		Description	
0000 ~ 0FFF (Common Error)	0BXX (File transmission Error)	0B09	Service not available
		0B0A	Need account for login
		0B0B	Can't open data connection
		0B0C	Connection closed, transfer aborted
		0B0D	File unavailable (e.g., file busy)
		0B0E	Local error in processing
		0B0F	Insufficient storage space in system
		0B10	Service ready in (n) minutes
		0B11	Data connection already open, transfer starting
		0B12	Data connection open, no transfer in progress
		0B13	Closing data connection
	0CXX (Apply Error)	0000	Success
		0C01	Apply trying
		0C02	Apply request is ordered but Apply action can't be executed
		0C03	Another apply operation is running
		0C0A	No apply operation to be performed
		0C0B	Can't receive any apply file
		0C0C	Apply file is broken
	0D00 ~ 0FFF	Reserved	
F000 ~ FFFF	Device Specific Error Code		
1000 ~ EFFF	User defined Error code		

6.4 Payload Messages

6.4.1 Related subclause for each DCMP message structure

DCMP message is composed of header and payload. The payload data is described according to the Message Type in the header. Table 4 shows the related subclauses for each DCMP message structure.

Table 4 — Related Subclause for each Message Payload Structure

Operation	Message	Related Subclause
Device Discovery	DEVICE_DISCOVERY_REQUEST	subclause 6.4.2
	DEVICE_DISCOVERY_RESPONSE	subclause 6.4.3
Device Advertisement	DEVICE_ADVERTISEMENT	subclause 6.4.4
Device Info	DEVICE_INFO_REQUEST	subclause 6.4.5
	DEVICE_INFO_RESPONSE	subclause 6.4.6
Device Control	DEVICE_CONTROL_REQUEST	subclause 6.4.7
	DEVICE_CONTROL_RESPONSE	subclause 6.4.8
Event	EVENT_NOTIFICATION	subclause 6.4.9
Event Subscription	EVENT_SUBSCRIPTION_REQUEST	subclause 6.4.10
	EVENT_SUBSCRIPTION_RESPONSE	subclause 6.4.11

Table 4 *(continued)*

Operation	Message	Related Subclause
Get File Info	GET_FILEINFO_REQUEST	subclause 6.4.12
	GET_FILEINFO_RESPONSE	subclause 6.4.13
Get File	GET_FILE_REQUEST	subclause 6.4.14
	GET_FILE_RESPONSE	subclause 6.4.15
	GET_FILE_RESULT	subclause 6.4.16
Put File	PUT_FILE_REQUEST	subclause 6.4.17
	PUT_FILE_RESPONSE	subclause 6.4.18
	PUT_FILE_RESULT	subclause 6.4.19
Apply	APPLY_REQUST	subclause 6.4.20
	APPLY_RESPONSE	subclause 6.4.21
	APPLY_RESULT	subclause 6.4.22
Device Registration	DEVICE_REGISTRATION_REQUEST	subclause 6.4.23
	DEVICE_REGISTRATION_RESPONSE	subclause 6.4.24
Service Registration	SERVICE_REGISTRATION_REQUEST	subclause 6.4.25
	SERVICE_REGISTRATION_RESPONSE	subclause 6.4.26

6.4.2 DEVICE_DISCOVERY_REQUEST

— This message can be used when a device or service needs to know which devices are connected in the network. Device discovery request can be performed with specific conditions (i.e. DiscoveryReqType) such as device identifier, device type or device name.

— If 'DeviceName' or 'DeviceID' is selected for the discovery request condition, 'StrTypeReq' (20 Bytes string) would be described and 'DeviceType' is selected, 'HexTypeReq' (2 Bytes Hex code) would be described respectively.

— If discovery condition is 'All', 'StrTypeReq' and 'HexTypeReq' field shall not be described either.

— The payload message structure is shown in Figure 15 and Table 5.

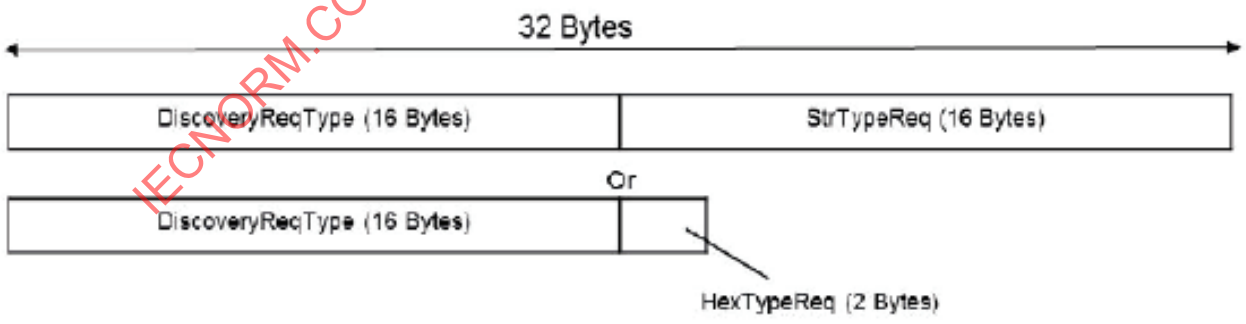


Figure 15 — DEVICE_DISCOVERY_REQUEST Message Structure

Table 5 — Description of DEVICE_DISCOVERY_REQUEST Message Structure

Field Name	Field Size	Description
DiscoveryReqType	16 Bytes (Char[16])	Describes one of the following — All — DeviceID — DeviceType — DeviceName
StrTypeReq	16 Bytes (Char[16])	Describes device name or device ID according to the DiscoveryRequestType
HexTypeReq	2 Bytes (uint16_t)	Describes device type according to the DiscoveryRequestType

6.4.3 DEVICE_DISCOVERY_RESPONSE

— This message is a response message for the DEVICE_DISCOVERY_REQUEST.

— Result: Describes whether discovery request message is reached to the destination well or not. If an error occurred, error code is described as shown in the Table 3.

— DeviceList : Describes device list including the information of 'Number of Devices', 'Device ID', 'Device Type', 'Device Name', 'Device Subname' and 'Device Capability'

— Each device has 'device type' according to its function. However, some devices have multi-functions and these devices can be expressed by combination of 'device type' (ex: Printer + Scanner). 'DeviceCapabilityList' describes a list of 'device type'.

— The payload message structure is shown in Figure 16 and Table 6.

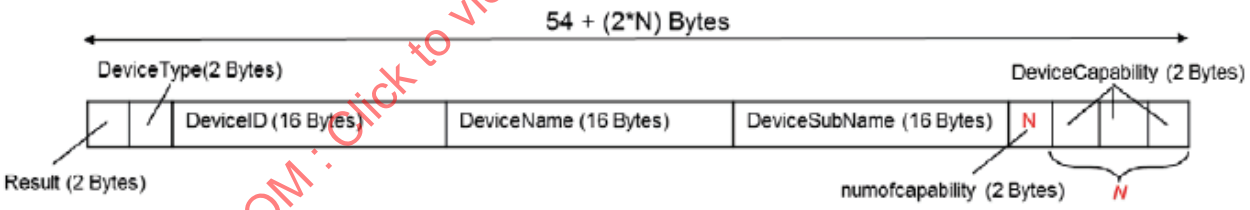


Figure 16 — DEVICE_DISCOVERY_RESPONSE Message Structure

Table 6 — Description of DEVICE_DISCOVERY_RESPONSE Message Structure

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Result code (Table 3)
DeviceType	2 Bytes (uint16_t)	Device Type (Annex B)
DeviceID	16 Bytes (Char[16])	Device ID
DeviceName	16 Bytes (Char[16])	Device Name

Table 6 *(continued)*

Field Name	Field Size	Description
DeviceSubName	16 Bytes (Char[16])	Device Sub name Describes in case of need
numofcapability	2 Bytes (uint16_t)	Number of Capability *M*
DeviceCapability	2 Bytes (uint16_t)	*1st* Device Type (Annex B)
…	…	…
DeviceCapability	2 Bytes (uint16_t)	*Mth* Device Type (Annex B)

6.4.4 DEVICE_ADVERTISEMENT

— This message is used to inform the device's plug-in or plug-out.

— This message structure is similar to the DEVICE_DISCOVERY_RESPONSE, but 'AdType' is added to distinguish between start(in) signal and end(out) signal.

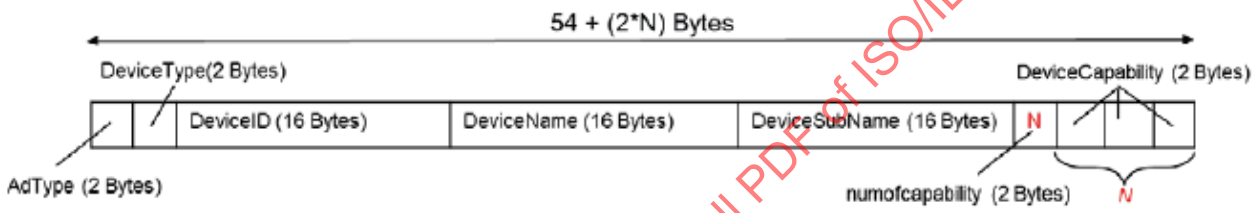


Figure 17 — DEVICE_ADVERTISEMENT Message Structure

Table 7 — Description of DEVICE_ADVERTISEMENT Message Structure

Field Name	Field Size	Description
AdType	16 Bytes (Char[16])	Describes one of the following — Start — End
DeviceType	2 Bytes (uint16_t)	Device Type (Annex B)
DeviceID	16 Bytes (Char[16])	Device ID
DeviceName	16 Bytes (Char[16])	Device Name
DeviceSubName	16 Bytes (Char[16])	Describes Device Sub name in case of need
numofcapability	2 Bytes (uint16_t)	Number of Capability *M*
DeviceCapability	2 Bytes (uint16_t)	*1st* Device Type (Annex B)
…	…	…
DeviceCapability	2 Bytes (uint16_t)	*Mth* Device Type (Annex B)

6.4.5 DEVICE_INFO_REQUEST

— A DEVICE_INFORMATION_REQUEST message is used when a device needs to know the system and network information of the other devices.

— In order to request device information, 'DeviceInfoReqType' shall be described and this request type is divided into three types;

  — BasicInfo: Basic information including the 'Device ID', 'Device Type' and etc.

  — FunctionList: Function list, which can be supported by device.

  — DeviceProperty:

    — CommonProperty: General information such as manufacture, version and so on.

    — ConfigProperty: Information about the setting parameters.

    — StatusProperty: Dynamic information such as CPU, memory or network usage.

    — DeviceSpecificProperty: Device specific properties, which are defined according to the device type.

— 'FullDescription' (includes above three types), 'CommonProperty', 'ConfigProperty', 'StatusProperty' and 'DeviceSpecificProperty' can be used for the 'DeviceInfoReqType' respectively.

— The payload message structure is shown in Figure 18 and Table 8.

Figure 18 — DEVICE_INFO_REQUEST Message Structure

Table 8 — Description of DEVICE_INFO_REQUEST Message Structure

Field Name	Field Size	Description
DeviceInfoReqType	32 Bytes (Char[32])	Describes one of the following — FullDescription — BasicInfo — FunctionList — DeviceProperty — CommonProperty — ConfigProperty — StausProperty — DeviceSpecificProperty

6.4.6 DEVICE_INFO_RESPONSE

— This message is a response message for the DEVICE_INFO_REQUEST.

— The response information can be classified into 'FullDescription', 'BasicInfo', 'FunctionList', 'DeviceProperty', 'CommonProperty', 'ConfigProperty', 'StatusProperty', and 'DeviceSpecificProperty' according to the request type.

6.4.6.1 When 'DeviceInfoReqType' is 'BasicInfo'

— 'BasicInfo' describes device basic information such as 'Number of Devices', 'Device ID', 'Device Type', 'Device Name' and 'Device Capability'

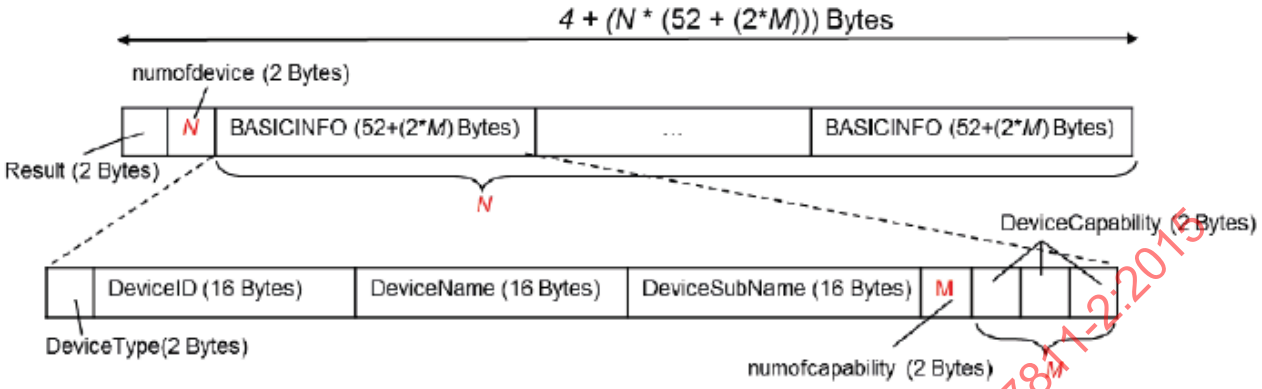


Figure 19 — DEVICE_INFO_RESPONSE Message Structure

Table 9 — Description of DEVICE_INFO_RESPONSE Message Structure (When 'DeviceInfoReqType' is 'BasicInfo')

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)
numofdevice	2 Bytes (uint16_t)	Number of device
DeviceType	2 Bytes (uint16_t)	Device Type (Annex B)
DeviceID	16 Bytes (Char[16])	Device ID
DeviceName	16 Bytes (Char[16])	Device Name
DeviceSubName	16 Bytes (Char[16])	Describes Device Sub name in case of need
numofcapability	2 Bytes (uint16_t)	Number of Capability M
DeviceCapability	2 Bytes (uint16_t)	1st Device Type (Annex B)
...	...	...
DeviceCapability	2 Bytes (uint16_t)	M^{th} Device Type (Annex B)

6.4.6.2 When 'DeviceInfoReqType' is 'FunctionList'

— 'FunctionList' describes a list of function, which can be supported by a certain device.

— Function ID: is defined according to each device type.

— FunctionCategory: let the service knows whether this message is for the control (needs response) or event (don't need response).

— 'InputList' is the input parameters needed in a certain device control. This information includes input size, id, name and so on.

— Input: describes the single input parameter information

— Inputs: describes list of 'Input's.

— 'OutputList' is the response parameters including the event data. This information includes output size, id, name and so on.

— Output: describes the single output parameter information

— Outputs: describes list of 'Output's.

— Data: describes data value with data name, value unit, minimum data value, maximum data value and so on

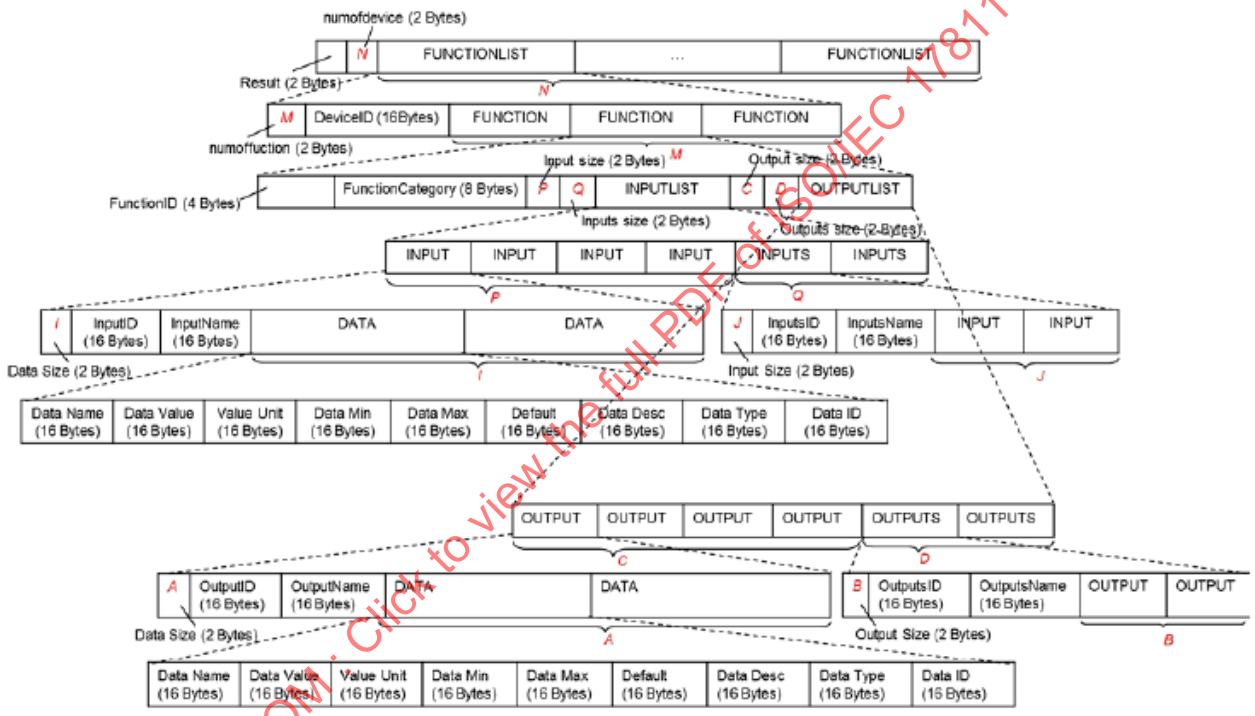


Figure 20 — DEVICE_INFO_RESPONSE Message Structure

Table 10-1 — Description of DEVICE_INFO_RESPONSE Message Structure (When 'DeviceInfoReqType' is 'FunctionList')

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)
numofdevice	2 Bytes (uint16_t)	Number of device
numoffunction	2 Bytes (uint16_t)	Number of Function (N)
DeviceID	16 Bytes (Char[16])	Device ID (UUID)

Table 10*(continued)*

Field Name	Field Size	Description
FUNCTION	Variable	1st Function (see table 10-2)
…	…	…
FUNCTION	Variable	Nth Function (see table 10-2)

Table 10-2 — FUNCTION Field Message Structure

Field Name	Field Size	Description
FunctionID	4 Bytes (uint32_t)	1st Function ID
FunctionCategory	8 Bytes (Char[8])	Describes one of the following · Control · Event
Input Size	2 Bytes (uint16_t)	Input Number of 1st function
Inputs Size	2 Bytes (uint16_t)	Inputs Number of 1st function
INPUT	Variable	Input Data (see table 10-3)
INPUTS	Variable	Inputs Data (see table 10-4)
Output Size	2 Bytes (uint16_t)	Output Number of 1st function
Outputs Size	2 Bytes (uint16_t)	Outputs Number of 1st function
OUTPUT	Variable	Output Data (see table 10-5)
OUTPUTS	Variable	Outputs Data (see table 10-6)

Table 10-3 — INPUT Field Message Structure

Field Name	Field Size	Description
Data Size	2 Bytes (uint16_t)	Number of **Data** P
Input ID	16 Bytes (Char[16])	Input ID
Input Name	16 Bytes (Char[16])	Input Name
DATA	130 Bytes	1st Data (See table 10-7)
…	…	…
DATA	130 Bytes	Pth Data (See table 10-7)

Table 10-4 — INPUS Field Message Structure

Field Name	Field Size	Description
Inputs Size	2 Bytes (uint16_t)	Number of **INPUT** Q

Table 10*(continued)*

Field Name	Field Size	Description
Inputs ID	16 Bytes (Char[16])	Inputs ID
Inputs Name	16 Bytes (Char[16])	Inputs Name
INPUT	variable	1st Input Data (See table 10-3)
...	...	...
INPUT	variable	Qth Input Data (See table 10-3)

Table 10-5 — OUTPUT Field Message Structure

Field Name	Field Size	Description
Data Size	2 Bytes (uint16_t)	Number of **Data** P
Output ID	16 Bytes (Char[16])	Output ID
Output Name	16 Bytes (Char[16])	Output Name
DATA	130 Bytes	1st Data (See table 10-7)
...	...	...
DATA	130 Bytes	Pth Data (See table 10-7)

Table 10-6 — OUTPUS Field Message Structure

Field Name	Field Size	Description
Outputs Size	2 Bytes (uint16_t)	Number of **OUTPUT** Q
Outputs ID	16 Bytes (Char[16])	Outputs ID
Outputs Name	16 Bytes (Char[16])	Outputs Name
OUTPUT	variable	1st Output Data (See table 10-5)
...	...	...
OUTPUT	variable	Qth Output Data (See table 10-5)

Table 10-7 — DATA Field Message Structure

Field Name	Field Size	Description
Data Name	16 Bytes (Char[16])	Data Name
Data Value	16 Bytes (Char[16])	Data Value
Data Valueunit	2 Bytes (uint16_t)	UnitType (Annex A)

Table 10*(continued)*

Field Name	Field Size	Description
Data Min	16 Bytes (Char[16])	Data Minimum Value
Data Max	16 Bytes (Char[16])	Data Maximum Value
Data Default	16 Bytes (Char[16])	Data Default Value
Data Desc	16 Bytes (Char[16])	Describes any Description in case of need
Data Type	16 Bytes (Char[16])	Describes Data Type in case of need
Data ID	16 Bytes (Char[16])	Describes Data ID in case of need

6.4.6.3 When 'DeviceInfoReqType' is 'CommonProperty'

— 'CommonProperty' describes general information such as manufacturer, version, physical size, weight and so on.

— The message structure is shown in Figure 21 and Table 11-1 to 11-8.

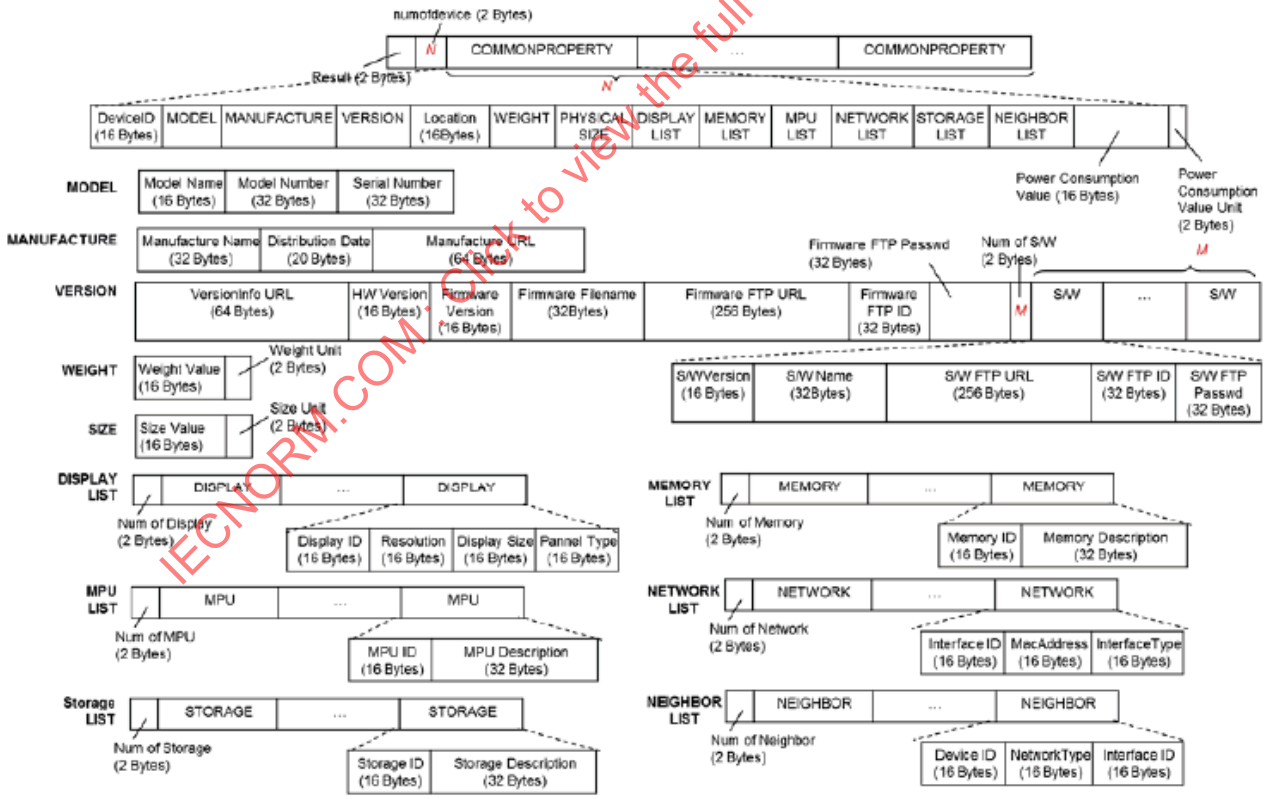


Figure 21 — DEVICE_INFO_RESPONSE Message Structure

Table 11-1 — Description of DEVICE_INFO_RESPONSE Message Structure (When 'DeviceInfoReqType' is 'CommonProperty')

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)
numofdevice	2 Bytes (uint16_t)	Number of device
DeviceID	16 Bytes (Char[16])	Device ID
ModelName	16 Bytes (Char[16])	Model Name
ModelNumber	32 Bytes (Char[32])	Model Number
SerialNumber	32 Bytes (Char[32])	Serial Number
ManufactureName	32 Bytes (Char[32])	Manufacture Name
DistributionDate	20 Bytes (Char[20])	Distribution Date YYYY-MM-DDTHH:MM:SS example: 2011-12-31T14:00:00
ManufactureURL	64 Bytes (Char[64])	Manufacture URL
VersionInfoURL	64 Bytes (Char[64])	Version Information URL
HardwareVersion	16 Bytes (Char[16])	Hardware Version
Firmware Version	16 Bytes (Char[16])	Firmware Version
Firmware FileName	32 Bytes (Char[32])	Firmware File Name
Firmware FTPURL	256 Bytes (Char[256])	Firmware FTP URL
Firmware FTPID	32 Bytes (Char[32])	FTP ID
Firmware FTPPasswd	32 Bytes (Char[32])	FTP Passwd
Num of Software	2 Bytes (uint16_t)	Number of Software N
SOFTWARE	144 Bytes (Char[144])	1st Software (see table 11-2)
…	…	…
SOFTWARE	144 Bytes (Char[144])	Nth Software (see table 11-2)

Table 11*(continued)*

Field Name	Field Size	Description
Location	16 Bytes (Char[16])	Describes Location in case of need
WeightValue	16 Bytes (Char[16])	Weight Value
WeightUnit	2 Bytes (uint16_t)	UnitType (Annex A)
SizeValue	16 Bytes (Char[16])	Physical Size Value
SizeUnit	2 Bytes (uint16_t)	UnitType (Annex A)
numofdisplay	2 Bytes (uint16_t)	Number of Display N
DISPLAY	64 Bytes (Char[64])	1st Display (see table 11-3)
...	...	...
DISPLAY	64 Bytes (Char[64])	Nth Display (see table 11-3)
numofmemory	2 Bytes (uint16_t)	Number of Memory N
MEMORY	48 Bytes (Char[48])	1st Memory (see table 11-4)
...	...	...
MEMORY	48 Bytes (Char[48])	Nth Memory (see table 11-4)
numofMPU	2 Bytes (uint16_t)	Number of MPU N
MPU	48 Bytes (Char[48])	1st MPU (see table 11-5)
...	...	...
MPU	48 Bytes (Char[48])	Nth MPU (see table 11-5)
numofnetwork	2 Bytes (uint16_t)	Number of Network N
NETWORK	48 Bytes (Char[48])	1st network (see table 11-6)
...	...	...
NETWORK	48 Bytes (Char[48])	Nth network (see table 11-6)
numofstorage	2 Bytes (uint16_t)	Number of Storage N

Table 11*(continued)*

Field Name	Field Size	Description
STORAGE	48 Bytes (Char[48])	1st Storage (see table 11-7)
...	...	...
STORAGE	48 Bytes (Char[48])	Nth Storage (see table 11-7)
numofneighbor	2 Bytes (uint16_t)	Number of Neighbor N
NEIGHBOR	48 Bytes (Char[48])	1st Neighbor (see table 11-8)
...	...	...
NEIGHBOR	48 Bytes (Char[48])	Nth Neighbor (see table 11-8)
ConsumptionValue	16 Bytes (Char[16])	Power Consumption Value
Power Unit	2 Bytes (uint16_t)	UnitType (Annex A)

Table 11-2 — SOFTWARE field Message Structure

Field Name	Field Size	Description
Software Version	16 Bytes (Char[16])	Software Version
Software FileName	32 Bytes (Char[32])	Software File Name
Software FTPURL	256 Bytes (Char[256])	Software FTP URL
Software FTPID	32 Bytes (Char[32])	Software FTP ID
Software FTPPasswd	32 Bytes (Char[32])	Software FTP Passwd

Table 11-3 — DISPLAY field Message Structure

Field Name	Field Size	Description
DisplayID	16 Bytes (Char[16])	Display ID (16bytes UUID)
Resolution	16 Bytes (Char[16])	Display Resolution (ex:1920x1080)
DisplaySize	16 Bytes (Char[16])	Display Size (inch)
PannelType	16 Bytes (Char[16])	Display Pannel Type

Table 11-4 — MEMORY field Message Structure

Field Name	Field Size	Description
MemoryID	16 Bytes (Char[16])	Memory ID (16bytes UUID)
MemoryDescription	32 Bytes (Char[32])	Memory Description

Table 11-5 — MPU field Message Structure

Field Name	Field Size	Description
MPUID	16 Bytes (Char[16])	MPU ID (16bytes UUID)
MPUDescription	32 Bytes (Char[32])	MPU Description

Table 11-6 — NETWORK field Message Structure

Field Name	Field Size	Description
InterfaceID	16 Bytes (Char[16])	Interface ID (16bytes UUID)
MacAddress	16 Bytes (Char[16])	Mac Address
InterfaceType	16 Bytes (Char[16])	Interface Type

Table 11-7 — STORAGE field Message Structure

Field Name	Field Size	Description
StorageID	16 Bytes (Char[16])	Storage ID (16bytes UUID)
StorageDescription	32 Bytes (Char[32])	Storage Description

Table 11-8 — NEIGHBOR field Message Structure

Field Name	Field Size	Description
DeviceID	16 Bytes (Char[16])	Device ID (16bytes UUID)
NetworkType	16 Bytes (Char[16])	Describes one of the following · Ethernet · RS485 · RS232
InterfaceID	16 Bytes (Char[16])	Interface ID (16bytes UUID)

6.4.6.4 When 'DeviceInfoReqType' is 'ConfigProperty'

— 'ConfigProperty' describes setting parameters for network or device configuration. Each manufacturer can describe the configuration file name and file URL.

— The message structure is shown in Figure 22 and Table 12-1 to 12-3.

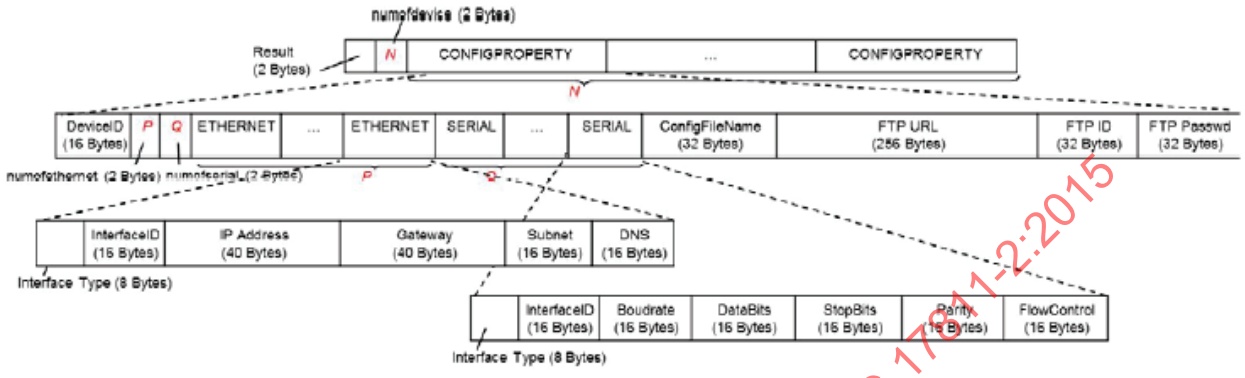


Figure 22 — DEVICE_INFO_RESPONSE Message Structure

Table 12-1 — Description of DEVICE_INFO_RESPONSE Message Structure (When 'DeviceInfoReqType' is 'ConfigProperty')

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)
numofdevice	2 Bytes (uint16_t)	Number of device
DeviceID	16 Bytes (Char[16])	Device ID
Num of Ethernet interface	2 Bytes (uint16_t)	Number of Ethernet Interface
Num of Serial interface	2 Bytes (uint16_t)	Number of Serial Interface
ETHERNET	116 Bytes (Char[116])	1st Ethernet (see table 12-2)
...	...	...
ETHERNET	116 Bytes (Char[116])	Nth Ethernet (see table 12-2)
SERIAL	108 Bytes (Char[108])	1st Serial (see table 12-3)
...	...	...
SERIAL	108 Bytes (Char[108])	Nth Serial (see table 12-3)
ConfigurationFileName	32 Bytes (Char[32])	Configuration File Name

Table 12*(continued)*

Field Name	Field Size	Description
FTPURL	256 Bytes (Char[256])	FTP URL
FTPID	32 Bytes (Char[32])	FTP ID
FTPPassword	32 Bytes (Char[32])	FTP Password

Table 12-2 — ETHERNET field Message Structure

Field Name	Field Size	Description
InterfaceType	8 Bytes (Char[8])	Describes one of the following · IPV4 · IPV6
InterfaceID	16 Bytes (Char[16])	Interface ID (16 bytes UUID)
IPAddress	40 Bytes (Char[40])	IP Address
Gateway	16 Bytes (Char[40])	Gateway IP
Subnet	16 Bytes (Char[16])	IPV4-SubnetMask, IPV6-SubnetPrefix
DNS	16 Bytes (Char[16])	N^{th} Domain Name Server IP

Table 12-3 — SERIAL field Message Structure

Field Name	Field Size	Description
InterfaceType	8 Bytes (Char[8])	Interface Type
InterfaceID	16 Bytes (Char[16])	Interface ID(16 bytes UUID)
Boudrate	16 Bytes (Char[16])	Boudrate
DataBits	16 Bytes (Char[16])	Data Bits
StopBits	16 Bytes (Char[16])	Stop Bits
Parity	16 Bytes (Char[16])	Parity Bits
FlowControl	16 Bytes (Char[16])	Flow Control

6.4.6.5 When 'DeviceInfoReqType' is 'StatusProperty'

— ‘StatusProperty’ describes dynamic information such as mpu, memory or network usage and so on.

— The message structure is shown in Figure 23 and Table 13-1 to 13-5.

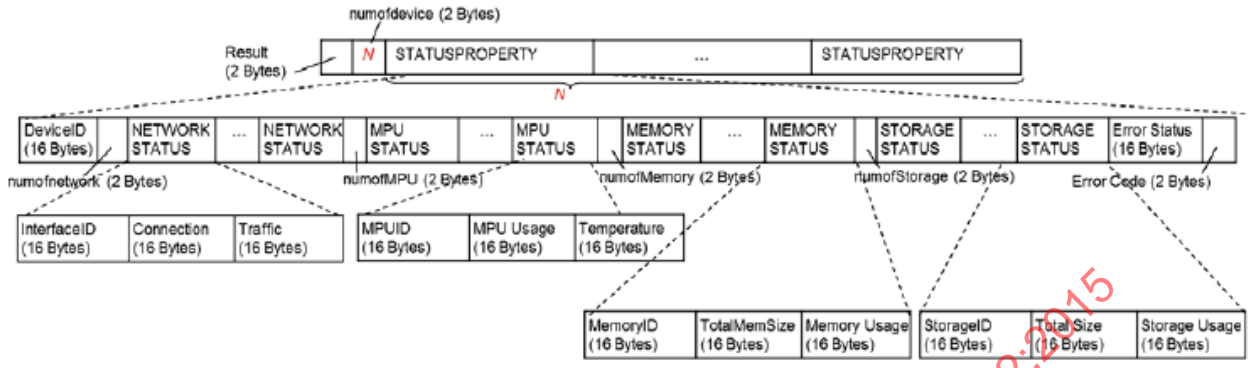


Figure 23 — DEVICE_INFO_RESPONSE Message Structure

Table 13-1 — Description of DEVICE_INFO_RESPONSE Message Structure (When ‘DeviceInfoReqType’ is ‘StatusProperty’)

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)
numofdevice	2 Bytes (uint16_t)	Number of device
DeviceID	16 Bytes (Char[16])	Device ID
Num of Network	2 Bytes (uint16_t)	Number of Network Interface N
NETWORKSTATUS	48 Bytes (Char[48])	1st Network Status (see table 13-2)
...	...	...
NETWORKSTATUS	48 Bytes (Char[48])	Nth Network Status (see table 13-2)
Num of MPU	2 Bytes (uint16_t)	Number of MPU N
MPUSTATUS	48 Bytes (Char[48])	1st MPU (see table 13-3)
...	...	...
MPUSTATUS	48 Bytes (Char[48])	Nth MPU (see table 13-3)
Num of Memory	2 Bytes (uint16_t)	Number of Memory N
MEMORY	48 Bytes (Char[48])	1st Memory (see table 13-4)
...	...	...

Table 13*(continued)*

Field Name	Field Size	Description
MEMORY	48 Bytes (Char[48])	Nth Memory (see table 13-4)
Num of Storage	2 Bytes (uint16_t)	Number of Storage N
STORAGE	48 Bytes (Char[48])	1st Storage (see table 13-5)
…	…	…
STORAGE	48 Bytes (Char[48])	Nth Storage (see table 13-5)
Error Status	16 Bytes (Char[16])	Describes one of the following · Normal · Abnormal
Error Code	2 Bytes (uint16_t)	Error Code (Table 3)

Table 13-2 — NETWORKSTATUS Message Structure

Field Name	Field Size	Description
InterfaceID	16 Bytes (Char[16])	Interface ID (16 bytes UUID)
Connection	16 Bytes (Char[16])	Describes one of the following · Online · Offline
Traffic	16 Bytes (Char[16])	Current network traffic and describes in Kbps

Table 13-3 — MPUSTATUS Message Structure

Field Name	Field Size	Description
MPUID	16 Bytes (Char[16])	MPU ID (16 bytes UUID)
MPUUsage	16 Bytes (Char[16])	MPU Usage (%)
Temperature	16 Bytes (Char[16])	Temperature (°C)

Table 13-4 — MEMORYSTATUS Message Structure

Field Name	Field Size	Description
MemoryID	16 Bytes (Char[16])	Memory ID (16 bytes UUID)

Table 13*(continued)*

Field Name	Field Size	Description
TotalMemSize	16 Bytes (Char[16])	Memory Size (Mbytes)
MemUsage	16 Bytes (Char[16])	Memory Usage (%)

Table 13-5 — STORAGESTATUS Message Structure

Field Name	Field Size	Description
StorageID	16 Bytes (Char[16])	Storage ID (16 bytes UUID)
TotalStorageSize	16 Bytes (Char[16])	Total Storage Size (Mbytes)
StorageUsage	16 Bytes (Char[16])	Storage Usage (%)

6.4.6.6 When 'DeviceInfoReqType' is 'DeviceSpecificProperty'

— According to the device type, 'DeviceSpecificProperty' can describe the various device specific properties.

— The message structure is shown in Figure 24 and Table 14-1 to 14-3.

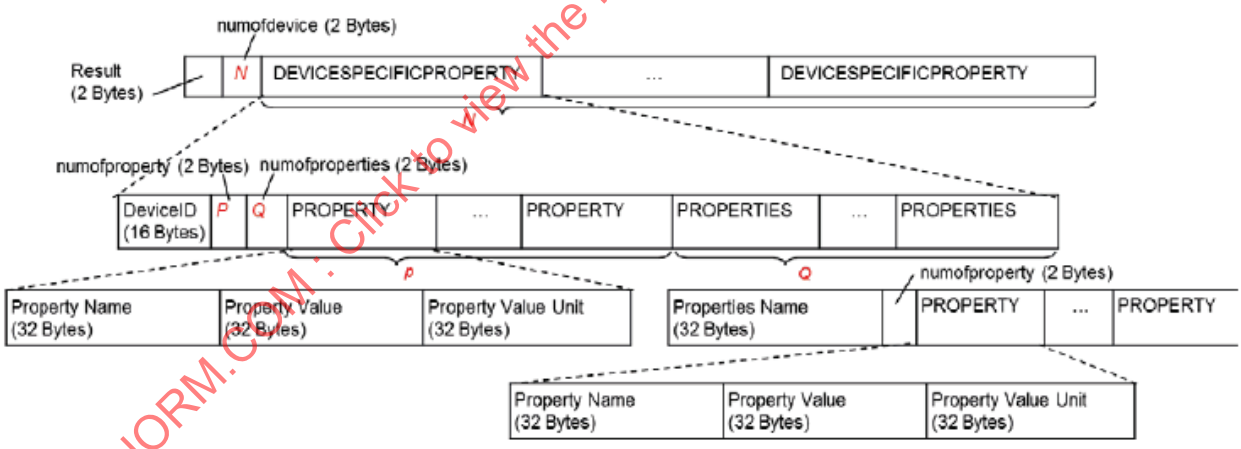


Figure 24 — DEVICE_INFO_RESPONSE Message Structure

Table 14-1 — Description of DEVICE_INFO_RESPONSE Message Structure (When 'DeviceInfoReqType' is 'DeviceSpecificProperty')

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)
numofdevice	2 Bytes (uint16_t)	Number of device
DeviceID	16 Bytes (Char[16])	Device ID

Table 14*(continued)*

Field Name	Field Size	Description
Num of property	2 Bytes (uint16_t)	Number of single property
Num of properties	2 Bytes (uint16_t)	Number of multi property
Property	96 Bytes (Char[96])	1st Property (See Table 14-2)
…	…	…
Property	96 Bytes (Char[96])	Pth Property (See Table 14-2)
Properties	Variable	1st Property (See Table 14-3)
…	…	…
Properties	Variable	Qth Property (See Table 14-3)

Table 14-2 — Property Message Structure

Field Name	Field Size	Description
Property Name	32 Bytes (Char[32])	Property Name
Property Value	32 Bytes (Char[32])	Property Value Unit
Property Value Unit	32 Bytes (Char[32])	Property Value Unit

Table 14-3 — Properties Message Structure

Field Name	Field Size	Description
Properties Name	32 Bytes (Char[32])	Properties Name
Num of property	2 Bytes (uint16_t)	Number of single property P
Property	96 Bytes (Char[96])	1st Property (See Table 14-2)
…	…	…
Property	96 Bytes (Char[96])	Pth Property (See Table 14-2)

6.4.6.7 When 'DeviceInfoReqType' is 'DeviceProperty'

— 'DeviceProperty' can be classified into four types of properties

  — CommonProperty: General information such as manufacturer, version and so on.

  — ConfigProperty: Information about the setting parameters.

  — StatusProperty: Dynamic information such as CPU, memory or network usage.

— DeviceSpecificProperty: Device specific properties are defined according to the device type.

— The message structure is shown in Figure 25 and Table 15 and more details about each property are listed below

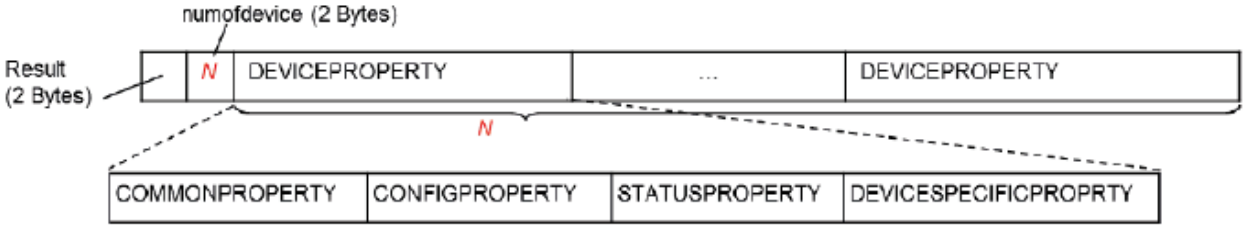


Figure 25 — DEVICE_INFO_RESPONSE Message Structure

Table 15 — Description of DEVICE_INFO_RESPONSE Message Structure (When 'DeviceInfoReqType' is 'DeviceProperty')

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)
numofdevice	2 Bytes (uint16_t)	Number of device
CommonProperty	variable	See table 11-1 without 'Result' field
ConfigProperty	variable	See table 12-1 without 'Result' field
StatusProperty	variable	See table 13-1 without 'Result' field
DeviceSpecificProperty	variable	See table 14-1 without 'Result' field

6.4.6.8 When 'DeviceInfoReqType' is 'FullDescription

— 'FullDescription' includes all of information above.

— The message structure is shown in Figure 26 and Table 16.

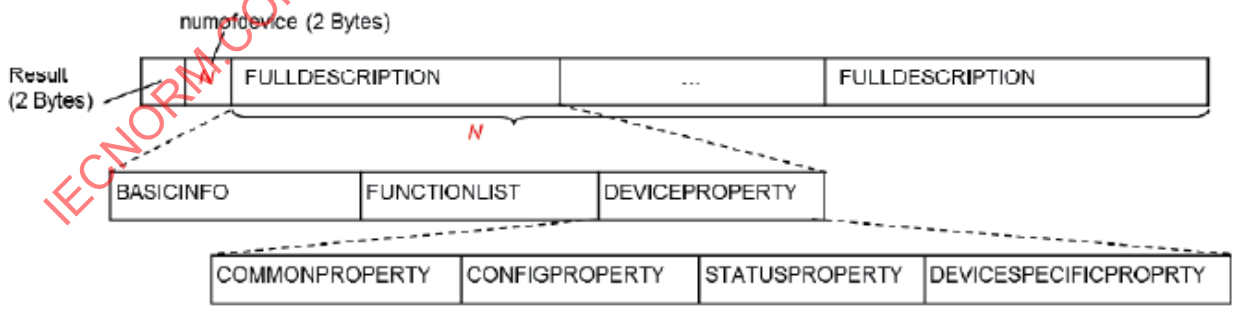


Figure 26 — DEVICE_INFO_RESPONSE Message Structure

Table 16 — Description of DEVICE_INFO_RESPONSE Message Structure (When 'DeviceInfoReqType' is 'FullDescription')

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)
numofdevice	2 Bytes (uint16_t)	Number of device
BasicInfo	variable	See table 9 without 'Result' field
FunctionList	variable	See table 10-1 without 'Result' field
DeviceProperty	variable	See table 15 without 'Result' field

6.4.7 DEVICE_CONTROL_REQUEST

— DEVICE_CONTROL_REQUEST message can be used when a device or service want to control the target device. When a device receives a DEVICE_CONTROL_REQUEST message, the device executes the requested control and returns the result.

— FunctionID: is defined according to the each device type. 4 Bytes are allocated for the 'FunctionID' and high 2 Bytes means the device type code.

— FunctionCategory: let the service knows whether this message is for the control (needs response) or event (don't need response).

— The payload message structure is shown in Figure 27 and Table 17.

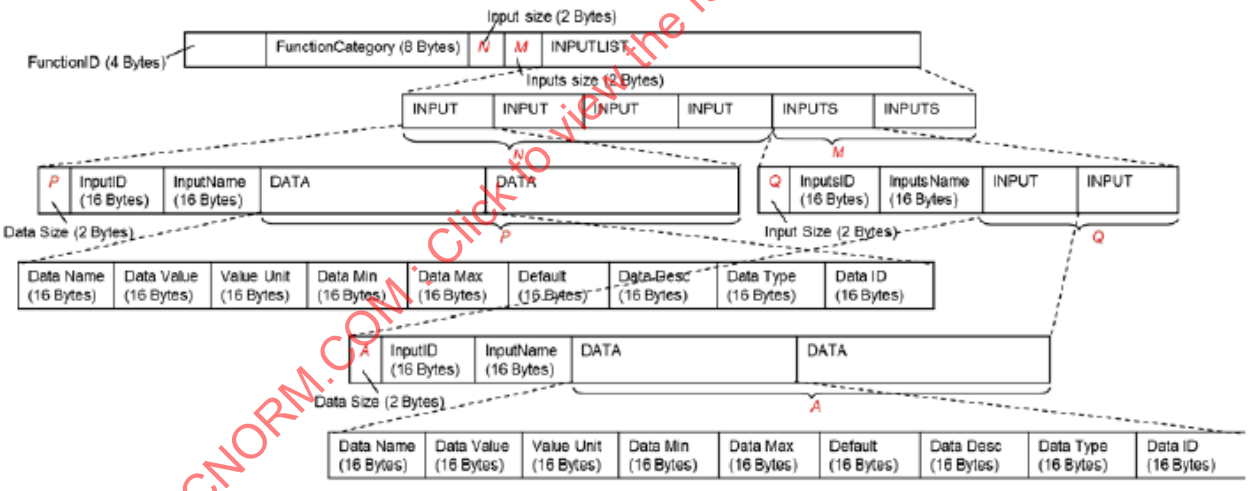


Figure 27 — DEVICE_CONTROL_REQUEST Message Structure

Table 17 — Description of DEVICE_CONTROL_REQUEST Message Structure

Field Name	Field Size	Description
FunctionID	4 Bytes (uint32_t)	Function ID
FunctionCategory	8 Bytes (Char[8])	Describes one of the following — Control — Event

Table 17 *(continued)*

Field Name	Field Size	Description
InputList Size	2 Bytes (uint16_t)	InputList Number of 1st function (Number of INPUT and INPUTS)
INPUT	Variable	Input Data (see table 10-3)
INPUTS	Variable	Inputs Data (see table 10-4)

6.4.8 DEVICE_CONTROL_RESPONSE

— This message is a response message for the DEVICE_CONTROL_REQUEST.

— 'OutputList' describes the result or response value.

— The message structure is shown in Figure 28 and Table 18.

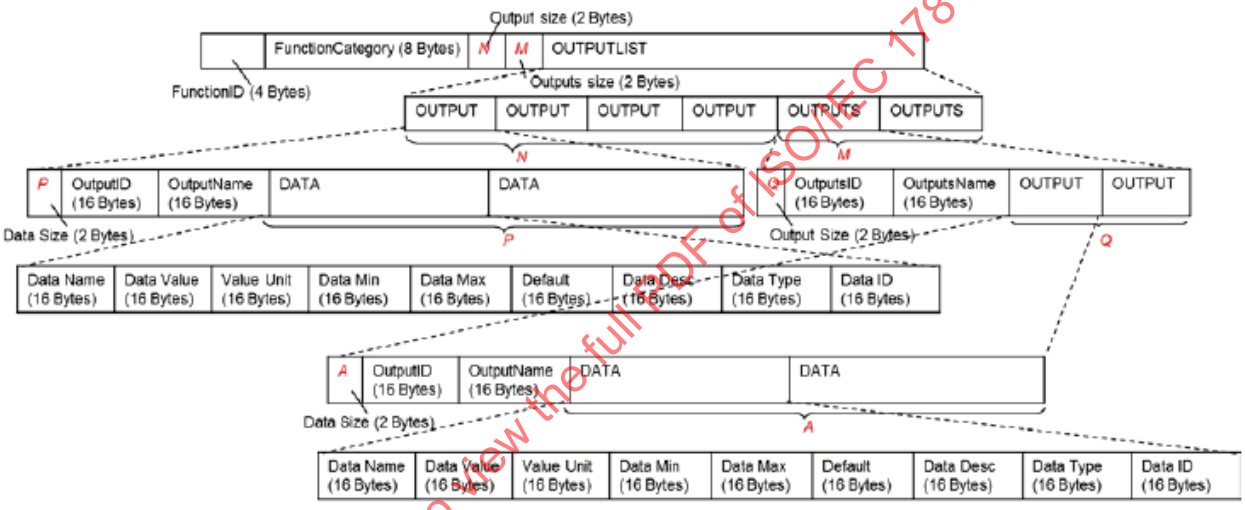


Figure 28 — DEVICE_CONTROL_RESPONSE Message Structure

Table 18 — Description of DEVICE_CONTROL_RESPONSE Message Structure

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)
FunctionID	4 Bytes (uint32_t)	Function ID
FunctionCategory	8 Bytes (Char[8])	Describes one of the following — Control — Event
OutputList Size	2 Bytes (uint16_t)	OutputList Number of 1st function (Number of OUTPUT and OUTPUTS)
OUTPUT	Variable	Output Data (see table 10-5)
OUTPUTS	Variable	Outputs Data (see table 10-6)

6.4.9 EVENT_NOTIFICATION

— When a sensor or device needs to transfer some data periodically without response, the EVENT_NOTIFICATION message can be used.

— FunctionID: is defined according to the each device type similar to the DEVICE_CONTROL_REQUEST/RESPONSE.

— FunctionCategory: let the service knows whether this message is for the control (needs response) or event (don't need response).

— The payload message structure is shown in Figure 29 and Table 19.

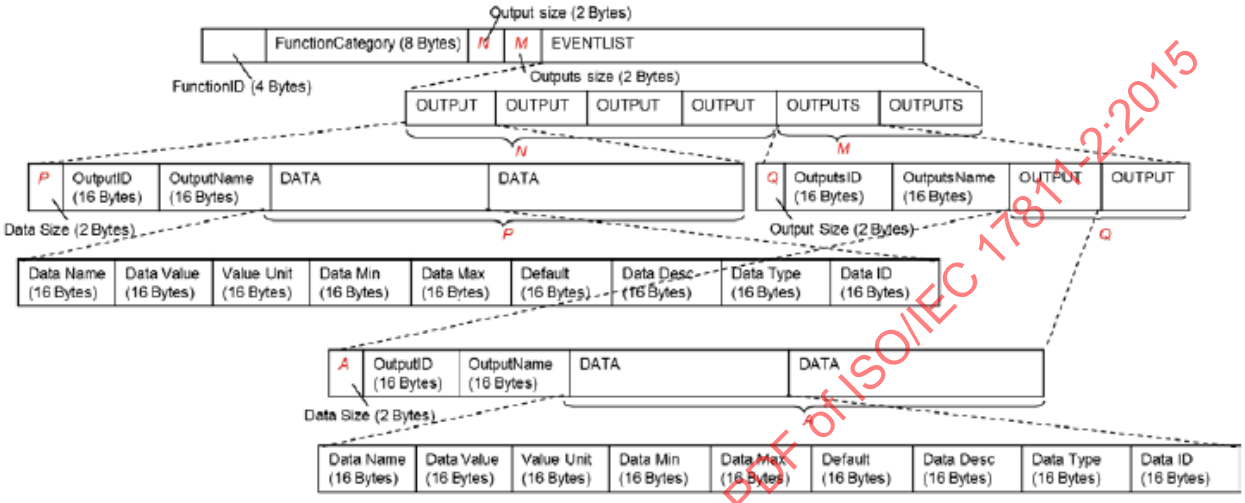


Figure 29 — EVENT_NOTIFICATION Message Structure

Table 19 — Description of EVENT_NOTIFICATION Message Structure

Field Name	Field Size	Description
FunctionID	4 Bytes (uint32_t)	Function ID
FunctionCategory	8 Bytes (Char[8])	Describes one of the following — Control — Event
EventList Size	2 Bytes (uint16_t)	EventList Number of 1st function (Number of EVENT and EVENTS)
OUTPUT	Variable	Event Data (see table 10-5)
OUTPUTS	Variable	Events Data (see table 10-6)

6.4.10 EVENT_SUBSCRIPTION_REQUEST

— Event information shall be reported to only interested devices. For this reason, event handling operation includes event subscription/un-subscription operations.

— FunctionID: is defined according to the each device type.

— SubscriptionInterval: subscription interval time and describes in millisecond

— SubscriptionType: subscription type, i.e. 'Renew', 'Registration', 'Cancel'

— The message structure is shown in Figure 30 and Table 20.

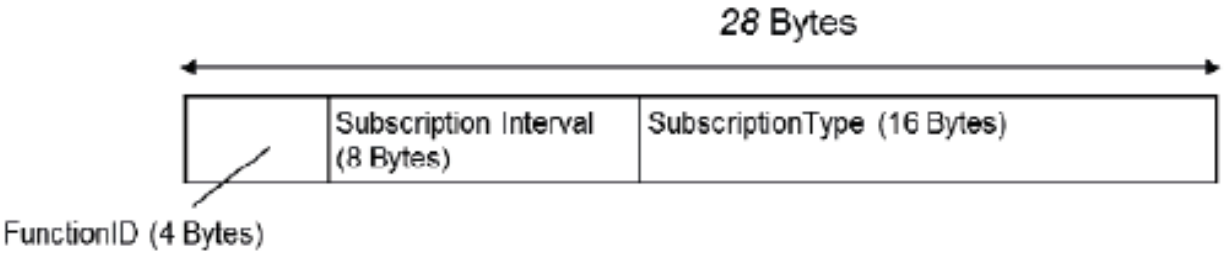


Figure 30 — EVENT_SUBSCRIPTION_REQUEST Message Structure

Table 20 — Description of EVENT_SUBSCRIPTION_REQUEST Message Structure

Field Name	Field Size	Description
FunctionID	4 Bytes (uint32_t)	Function ID
SubscriptionInterval	8 Bytes (Char[8])	Subscription Interval (ms)
subscriptionType	16 Bytes (Char[16])	Describes one of the following — Renew — Registration — Cancel

6.4.11 EVENT_SUBSCRIPTION_RESPONSE

— This message is a response message for the EVENT_SUBSCRIPTION_REQUEST.

— Result: Describes whether subscription request is performed with success or not. If an error occurred, error code can be described as shown in the Table 3.

— The message structure is shown in Figure 31 and Table 21.

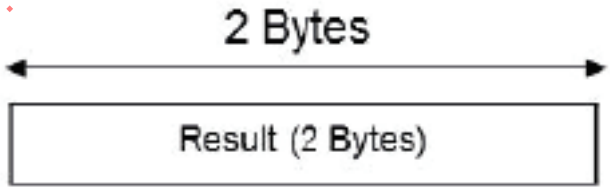


Figure 31 — EVENT_SUBSCRIPTION_RESPONSE Message Structure

Table 21 — Decription of EVENT_SUBSCRIPTION_RESPONSE Message Structure

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Result Code (Table 3)

6.4.12 GET_FILEINFO_REQUEST

— GET_FILEINFO_REQUEST message can be used when a service or device want to know about information of files which are saved in target device.

— ‘File Type’, ‘File Name’ and ‘Searching Period’ are described for the request condition.

— The message structure is shown in Figure 32 and Table 22.

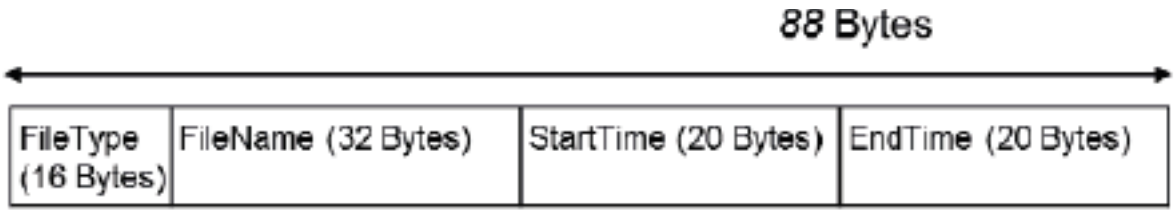


Figure 32 — GET_FILEINFO_REQUEST Message Structure

Table 22 — Description of GET_FILEINFO_REQUEST Message Structure

Field Name	Field Size	Description
FileType	16 Bytes (Char[16])	Describes one of the following — All — Log — Execution — Firmware — Debug — Configure — Image — Audio — Video — Text
Name	32 Bytes (Char[32])	File Name
StartTime	20 Bytes (Char[20])	Search Start Time YYYY-MM-DDTHH:MM:SS example: 2011-12-31T14:00:00
EndTime	20 Bytes (Char[20])	Search End Time YYYY-MM-DDTHH:MM:SS example: 2011-12-31T15:00:00

6.4.13 GET_FILEINFO_RESPONSE

— This message is response message for the GET_FILEINFO_REQUEST.

— ‘FileInfoList’ describes ‘File Type’, ‘File Name’, ‘File Size’ and ‘Latest Version Date’ and so on.

— The message structure is shown in Figure 33 and Table 23-1, 23-2.

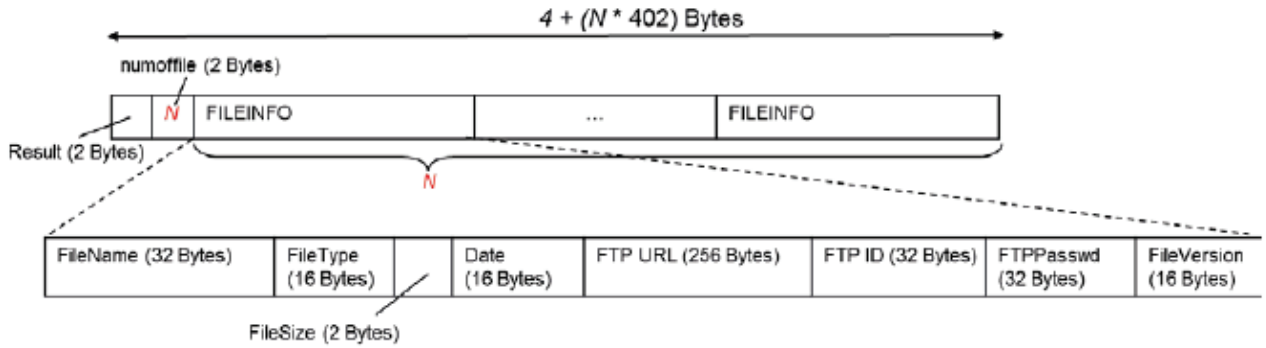


Figure 33 — GET_FILEINFO_RESPONSE Message Structure

Table 23-1 — Description of GET_FILEINFO_RESPONSE Message Structure

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Result code (Table 3)
numoffile	2 Bytes (uint16_t)	Number of Files N
FILEINFO	Variable	1st File Info (see table 23-2)
...	...	...
FILEINFO	Variable	Nth File Info (see table 23-2)

Table 23-2 — FILEINFO Field Message Structure

Field Name	Field Size	Description
FileName	32 Bytes (Char[32])	File Name
FileType	16 Bytes (Char[16])	Describes one of followings — All — Log — Execution — Firmware — Debug — Configure — Image — Audio — Video — Text
FileSize	8 Bytes (Char[8])	File Size (Mbytes)
Date	20 Bytes (Char[20])	Date YYYY-MM-DDTHH:MM:SS example: 2011-12-31T14:00:00

Table 23*(continued)*

Field Name	Field Size	Description
FTP URL	256 Bytes (Char[256])	File FTP URL (FTP URL)
FTPID	32 Bytes (Char[32])	FTP ID
FTPPasswd	32 Bytes (Char[32])	FTP Passwd
FileVersion	16 Bytes (Char[16])	File Version

6.4.14 GET_FILE_REQUEST

— A GET_FILE_REQUEST message is an essential function for the device management since the software update and firmware update requires the updated file to be transferred to the target device.

— 'FTP URL', 'FTP ID', 'FTP password' and 'File Name' are described and file transmission shall be performed by FTP channel.

— The message structure is shown in Figure 34 and Table 24.

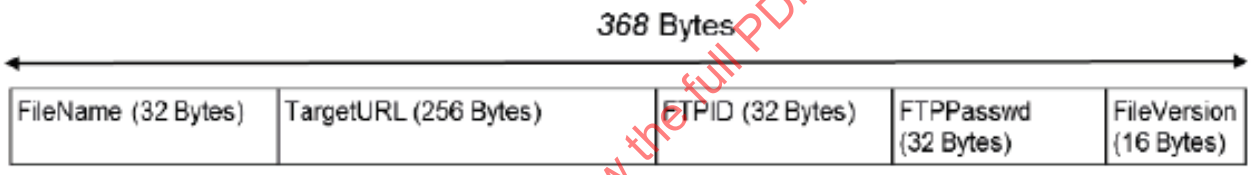


Figure 34 — GET_FILE_REQUEST Message Structure

Table 24 — Description of GET_FILE_REQUEST Message Structure

Field Name	Field Size	Description
TargetURL	256 Bytes (Char[256])	Target URL
FTPID	32 Bytes (Char[32])	FTP Account
FTPPasswd	32 Bytes (Char[32])	FTP Password
FileName	32 Bytes (Char[32])	File Name
FileVersion	16 Bytes (Char[16])	File Version

6.4.15 GET_FILE_RESPONSE

— This message is a response message for the GET_FILE_REQUEST and the message structure is shown in Figure 35 and Table 25.

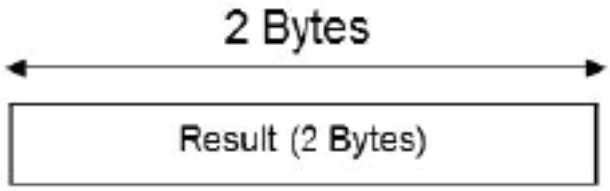


Figure 35 — GET_FILE_RESPONSE Message Structure

Table 25 — Description of GET_FILE_RESPONSE Message Structure

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)

6.4.16 GET_FILE_RESULT

— The device shall announce the transmission result when the file transmission is completed, because file transmission might take too much time.

— If an error occurred, error code can be described as shown in the Table 3.

— The message structure is shown in Figure 36 and Table 26.

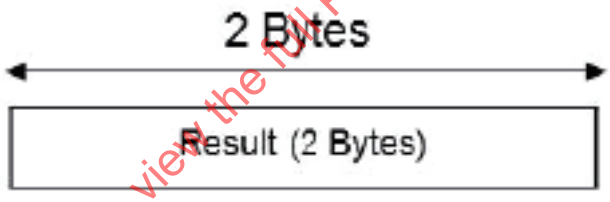


Figure 36 — GET_FILE_RESULT Message Structure

Table 26 — Description of GET_FILE_RESULT Message Structure

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)

6.4.17 PUT_FILE_REQUEST

— PUT_FILE_REQUEST message can be used when a service or device needs to upload some files to the target device.

— For the request, 'FTP URL', 'FTP ID', 'FTP password' and 'File Name' are described and file transmission shall be performed by FTP channel.

— The message structure is shown in Figure 37 and Table 27.

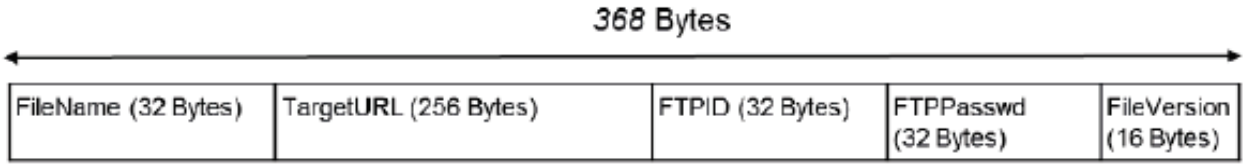


Figure 37 — PUT_FILE_REQUEST Message Structure

Table 27 — Description of PUT_FILE_REQUEST Message Structure

Field Name	Field Size	Description
FileName	32 Bytes (Char[32])	File Name
TargetURL	256 Bytes (Char[256])	Target URL
FTPID	32 Bytes (Char[32])	FTP Account
FTPPasswd	32 Bytes (Char[32])	FTP Password
FileVersion	16 Bytes (Char[16])	File Version

6.4.18 PUT_FILE_RESPONSE

— This message is a response message for the PUT_FILE_REQUEST and the message structure is shown in Figure 38 and Table 28.

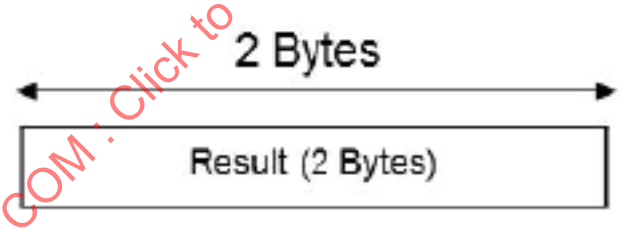


Figure 38 — PUT_FILE_RESPONSE Message Structure

Table 28 — Description of PUT_FILE_RESPONSE Message Structure

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)

6.4.19 PUT_FILE_RESULT

— The device shall announce the transmission result when the file transmission is completed, because file transmission might take too much time.

— If an error occurred, error code is described as shown in the Table 3.

— The message structure is shown in Figure 39 and Table 29.

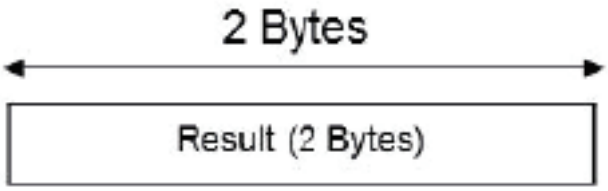


Figure 39 — PUT_FILE_RESULT Message Structure

Table 29 — Description of PUT_FILE_RESULT Message Structure

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)

6.4.20 APPLY_REQUEST

— APPLY_REQUEST message can be used when a service want to perform management function such as 'Update', 'Rollback', 'Execution file' and 'Delete a file'. These functions are described by 'ApplyType'

— A specific file can be needed in an apply process, so this message includes the information about those kind of files.

— The message structure is shown in Figure 40 and Table 30.

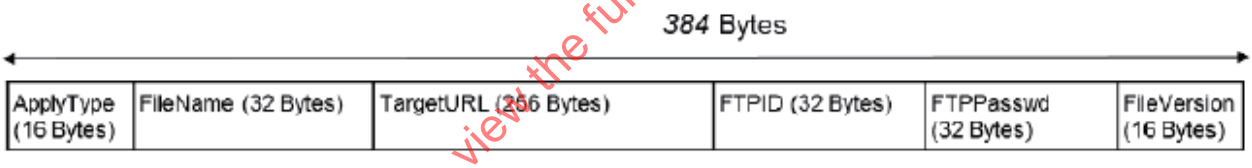


Figure 40 — APPLY_REQUEST Message Structure

Table 30 — Description of APPLY_REQUEST Message Structure

Field Name	Field Size	Description
ApplyType	16 Bytes (Char[16])	Describes one of followings — Execution — Update — Rollback — FileDelete
FileName	32 Bytes (Char[32])	File Name
TargetURL	256 Bytes (Char[256])	Target URL
FTPID	32 Bytes (Char[32])	FTP ID

Table 30 *(continued)*

Field Name	Field Size	Description
FTPPasswd	32 Bytes (Char[32])	FTP Password
FileVersion	16 Bytes (Char[16])	File Version

6.4.21 APPLY_RESPONSE

— This message is a response message for the APPLY_REQUEST and message structure is shown in Figure 41 and Table 31.

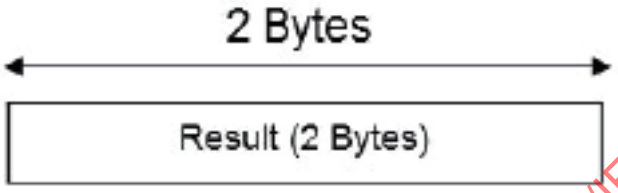


Figure 41 — APPLY_RESPONSE Message Structure

Table 31 — Description of APPLY_RESPONSE Message Structure

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)

6.4.22 APPLY_RESULT

— The device shall announce the apply result when the applying process is completed, because file transmission might take too much time.

— If an error occurred, error code is described as shown in the Table 3.

— The message structure is shown in Figure 42 and Table 32.

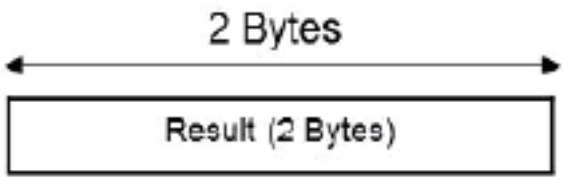


Figure 42 — APPLY_RESULT Message Structure

Table 32 — Description of APPLY_RESULT Message Structure

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)

6.4.23 DEVICE_REGISTRATION_REQUEST

— DEVICE_REGISTRATION_REQUEST message might be used when a device register its own information to the manufacturer server.

— The message structure is shown in Figure 43 and Table 33.

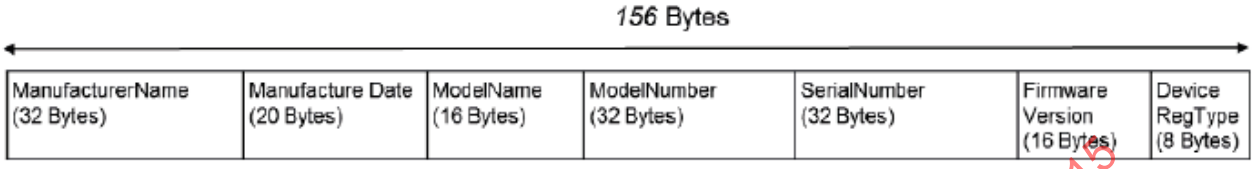


Figure 43 — DEVICE_REGISTRATION_REQUEST Message Structure

Table 33 — Description of DEVICE_REGISTRATION_REQUEST Message Structure

Field Name	Field Size	Description
Manufacturer Name	32 Bytes (Char[32])	Manufacturer Name
Manufacture Date	20 Bytes (Char[20])	Manufacture Date (Local time) - YYYY-MM-DDTHH:MM:SS Example: 2001-12-31T14:00:00
ModelName	16 Bytes (Char[16])	Model Name
ModelNumber	32 Bytes (Char[32])	Model Number
SerialNumber	32 Bytes (Char[32])	Serial Number
Firmware Version	16 Bytes (Char[16])	Firmware Version
DeviceRegType	8 Bytes (Char[8])	Describes one of followings — Reg — UnReg

6.4.24 DEVICE_REGISTRATION_RESPONSE

— This message is a response message for the DEVICE_REGISTRATION_REQUEST and message structure is shown in Figure 44 and Table 34.

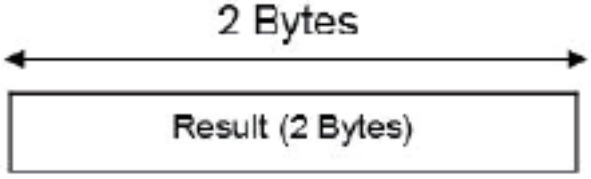


Figure 44 — DEVICE_REGISTRATION_RESPONSE Message Structure

Table 34 — Description of DEVICE_REGISTRATION_RESPONSE Message Structure

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)

6.4.25 SERVICE_REGISTRATION_REQUEST

— A SERVICE_REGISTRATION_REQUEST message is used when a user wants to register personal information to the manufacturer server.

— If user set the update field 'Auto', device would update its firmware automatically.

— The message structure is shown in Figure 45 and Table 35.

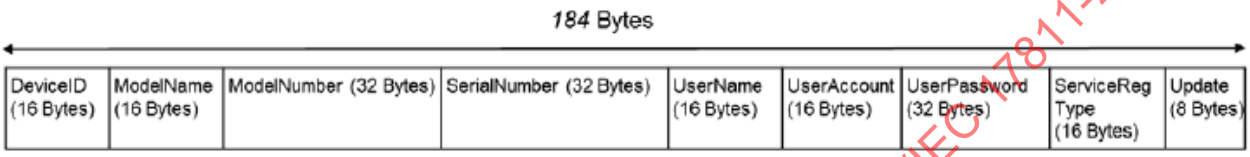


Figure 45 — SERVICE_REGISTRATION_REQUEST Message Structure

Table 35 — Description of SERVICE_REGISTRATION_REQUEST Message Structure

Field Name	Field Size	Description
DeviceID	16 Bytes (Char[16])	Device ID
ModelName	16 Bytes (Char[16])	Model Name
ModelNumber	32 Bytes (Char[32])	Model Number
SerialNumber	32 Bytes (Char[32])	Serial Number
UserName	16 Bytes (Char[16])	User Name
UserAccount	16 Bytes (Char[16])	User Account
UserPassword	32 Bytes (Char[16])	User Password
ServiceRegType	16 Bytes (Char[16])	Describes one of followings — New — Add — Delete
Update	8 Bytes (Char[8])	Describes one of followings — Auto — Manual

6.4.26 SERVICE_REGISTRATION_RESPONSE

— This message is a response message for the SERVICE_REGISTRATION_REQUEST and message structure is shown in Figure 46 and Table 36.

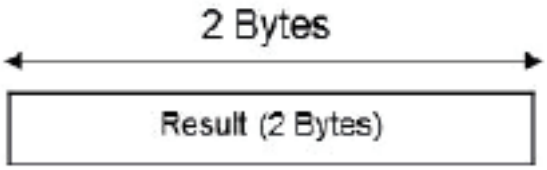


Figure 46 — SERVICE_REGISTRATION_RESPONSE Message Structure

Table 36 — Description of SERVICE_REGISTRATION_RESPONSE Message Structure

Field Name	Field Size	Description
Result	2 Bytes (uint16_t)	Error Code (Table 3)

Annex A
(normative)

Unit Types and Codes

unit-Type (2 Bytes)	Term ID of unit	unitType (2 Bytes)	Term ID of unit	unitType (2 Bytes)	Term ID of unit	unitType (2 Bytes)	Term ID of unit
0000	micrometer	0016	cm/min	002C	micormeter/ min^2	0043	volt
0001	mm	0017	meter/min	002D	mm/min^2	0044	millivolt
0002	cm	0018	km/min	002E	cm/min^2	0045	ampere
0003	meter	0019	inch/min	0030	$meter/min^2$	0046	milliampere
0004	km	001A	yard/min	0031	$km/smin^2$	0047	milliwatt
0005	inch	001B	mile/min	0032	$inch/min^2$	0048	watt
0006	yard	001C	micrometer/ hour	0033	$yard/min^2$	0049	kilowatt
0007	mile	001D	mm/hour	0034	$mile/min^2$	004A	lux
0008	mg	001E	cm/hour	0035	micormeter/ $hour^2$	004B	celsius
0009	gram	001F	meter/hour	0036	$mm/hour^2$	004C	fahrenheit
000A	kg	0020	km/hour	0037	$cm/hour^2$	004D	radian
000B	ton	0021	inch/hour	0038	$meter/hour^2$	004E	degree
000C	micrometer/sec	0022	yard/hour	0039	$km/hour^2$	004F	rad/sec
000D	mm/sec	0023	mile/hour	003A	$inch/hour^2$	0050	deg/sec
000E	cm/sec	0024	micrometer/ sec^2	003B	$yard/hour^2$	0051	rad/sec^2
000F	meter/sec	0025	mm/sec^2	003C	$mile/hour^2$	0052	deg/sec^2
0010	Km/sec	0026	cm/sec^2	003D	Nmm	0053	N/mm^2
0011	inch/sec	0027	$meter/sec^2$	003E	Npmm	0054	m^3
0012	yard/sec	0028	km/sec^2	003F	Hz	0055	kcal
0013	mile/sec	0029	$inch/sec^2$	0040	KHz	0056 ~ FFFF	Reserved
0014	micrometer/ min	002A	$yard/sec^2$	0041	MHz		
0015	mm/min	002B	$mile/sec^2$	0042	GHz		

Annex B
(normative)

Device Types

Device Type	Home Gateway/ Server	Home Information	Internet Consumer	Automation	Audio/ Video	Shared Unit	Miscellaneous
Code	11XX	12XX	13XX	14XX	15XX	17XX	1FXX
01	Home Gateway	Phone	Refrigerator	Light	Audio Speaker	VIOCS	UPS
02	Home Server	PDA	Air Conditioner	Gas Valve	HIFI Audio	Main Entrance	Complex Server
03	Digital Cable STB	PC	Microwave	Curtain	Camcorder	Remote Meter Reading Server	Network Device
04	Digital Satellite STB	Home Pad	Boiler	Remote Meter Reading	Camera	Elevator	Open/Close Sensor
05	Digital DMB STB	Video Phone	Oven	Door Lock	Scanner	CCTV	AP
06	Digital IP STB	Smart TV	Laundry	HVAC	Printer	Auto Parcel	Etc.
07	Wall Pad	Smart Phone	Running Machine	Batch Breaker	Display	Mobile Server	
08	Etc.	Etc.	Health Device	Expansion Security	Etc.	Lobby Phone	
09			Coffee Machine	Thermostat		Etc.	
0A			Fan	Door Bell			
0B			Heater	Ventilation			
0C			Etc.	Temperature Control			
0F				Etc.			

Annex C
(informative)

XML Schema and Example of DCMP Payload Message

DCMP message is composed of header and payload the payload data might be described in XML according to the message type in the header. Annex C shows the XML example of DCMP payload for each message type.

C.1 DEVICE_DISCOVERY_REQUEST

— XML schema structure of DEVICE_DISCOVERY_REQUEST Payload is shown in Figure C.1 and XML example is shown in Table C.1, respectively.

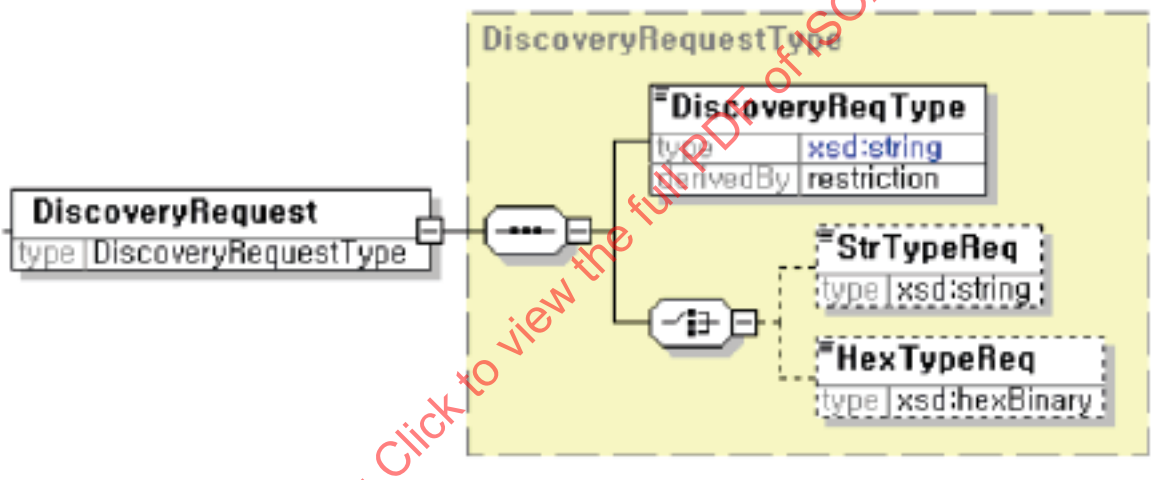


Figure C.1 — Device Discovery Request Schema Structure

— The following XML example shows the device discovery request message with specific condition, i.e. 'DeviceType' is 0x1605(CCTV)'

Table C.1 — XML Example of DEVICE_DISCOVERY_REQUEST Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <DiscoveryRequest>
      <DiscoveryReqType>DeviceType</DiscoveryReqType>
      <HexTypeReq>1605</HexTypeReq>
   </DiscoveryRequest>
</DMAP>
```

C.2 DEVICE_DISCOVERY_RESPONSE

— XML schema structure of DEVICE_DISCOVERY_RESPONSE message is shown in Figure C.2 and XML example is shown in Table C.2, respectively.

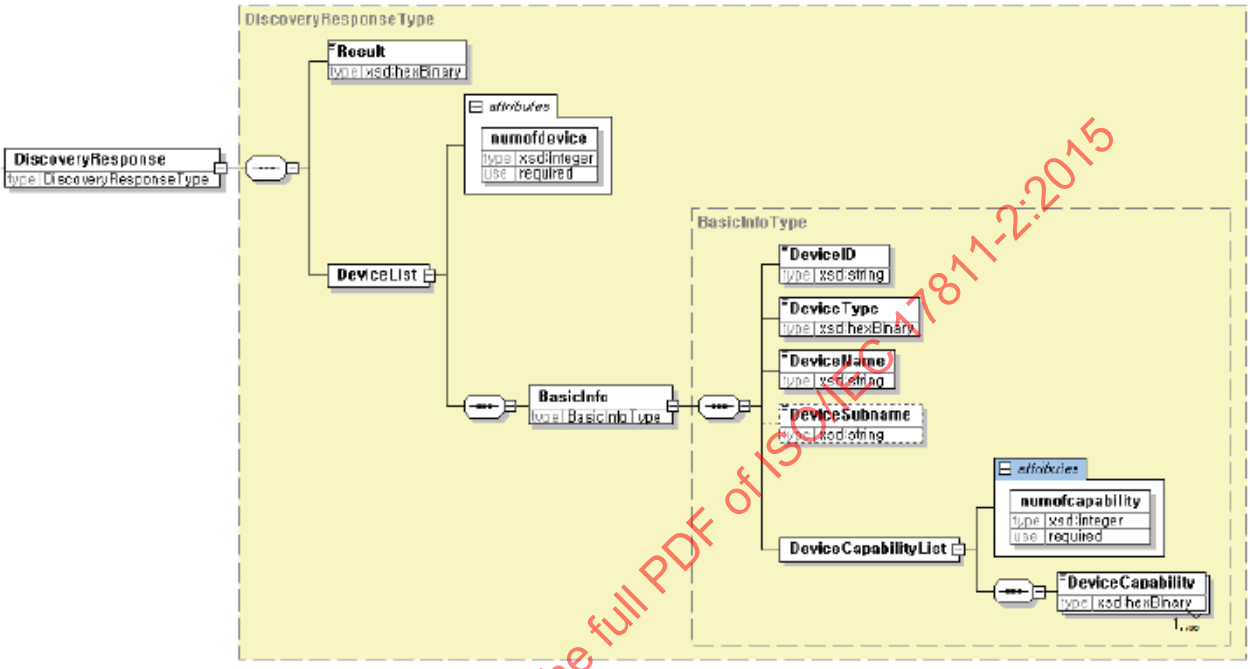


Figure C.2 — Device Discovery Response Schema Structure

— The following XML example shows the device discovery response message which provides the information of 'Number of Devices', 'Device ID', 'Device Type', 'Device Name' and 'Device Capability'

Table C.2 — XML Example of DEVICE_DISCOVERY_RESPONSE Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
< DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <DiscoveryResponse>
     <Result>0000</Result>
     <DeviceList numofdevice="1">
       <BasicInfo>
         <DeviceID>12345678901234567890</DeviceID>
         <DeviceType>1605</DeviceType>
         <DeviceName>CCTV</DeviceName>
         <DeviceSubname>Temp</DeviceSubname>
         <DeviceCapabilityList numofcapability="1">
                 <DeviceCapability>1605</DeviceCapability>
         </DeviceCapabilityList>
       </BasicInfo>
     </DeviceList>
   </DiscoveryResponse>
</ DMAP >
```

C.3 DEVICE_ADVERTISEMENT

— XML schema structure of DEVICE_ADVERTISEMENT message is shown in Figure C.3 and XML example is shown in Table C.3, respectively.

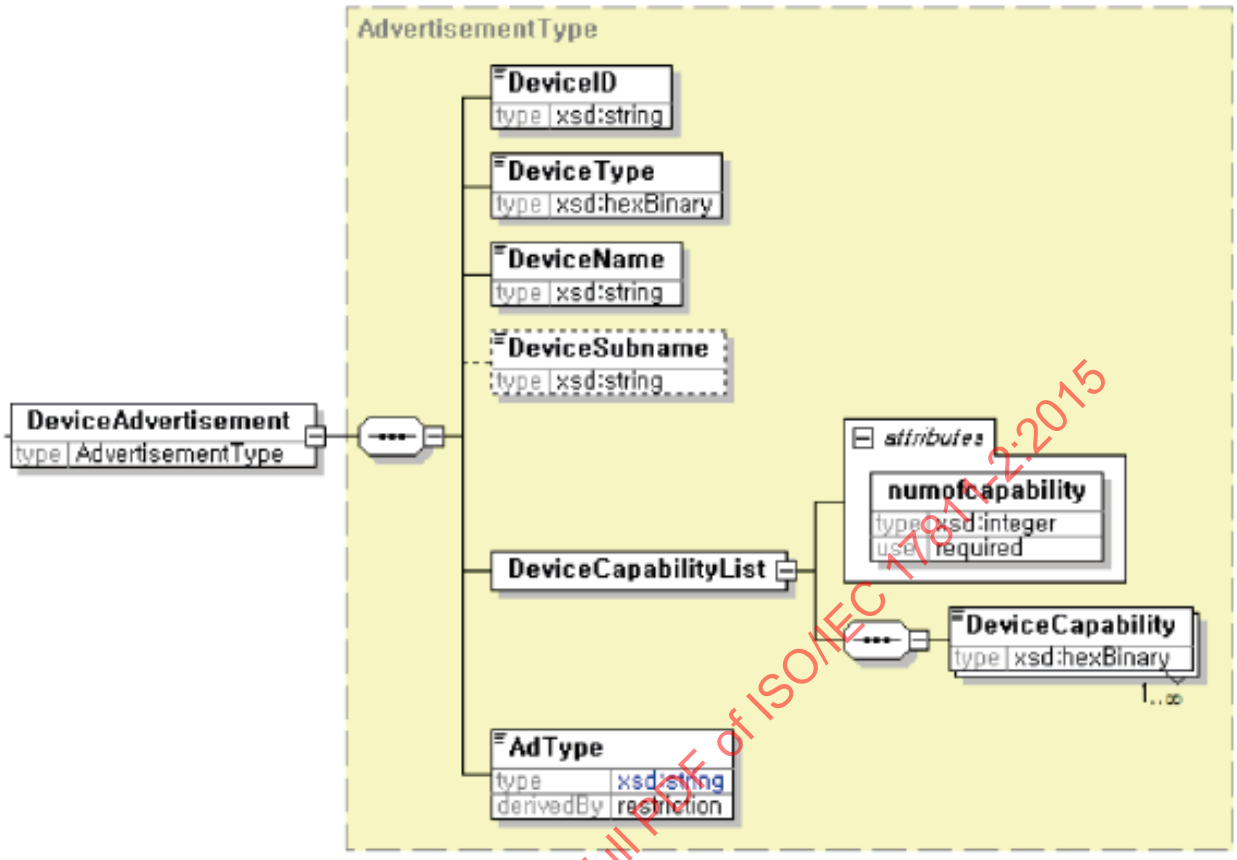


Figure C.3 — Device Advertisement Schema Structure

— The following XML example shows the device advertisement message which provides the information of 'Number of Devices', 'Device ID', 'Device Type', 'Device Name' and 'Device Capability'

Table C.3 — XML Example of DEVICE_ADVERTISEMENT Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <DeviceAdvertisement>
            <DeviceID>12345678901234567890</DeviceID>
            <DeviceType>1605</DeviceType>
            <DeviceName>CCTV</DeviceName>
            <DeviceCapabilityList numofcapability="1">
                  <DeviceCapability>1605</DeviceCapability>
            </DeviceCapabilityList>
            <AdType>Start</AdType>
      </DeviceAdvertisement>
</DMAP>
```

C.4 DEVICE_INFO_REQUEST

— XML schema structure of DEVICE_INFO_REQUEST message is shown in Figure C.4 and XML example is shown in Table C.4, respectively.

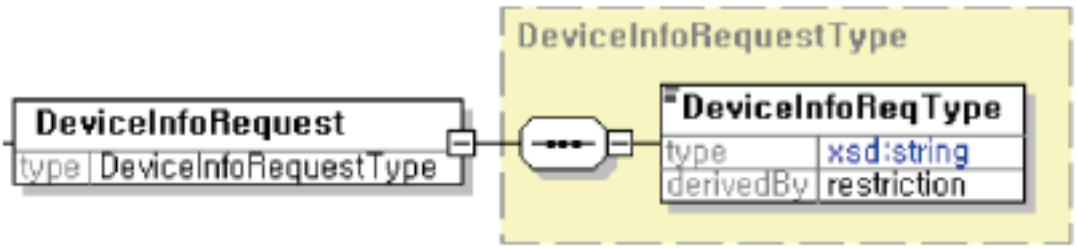


Figure C.4 — Device Info Request Schema Structure

— The following XML example shows the **full** device information request message.

Table C.4 — XML Example of DEVICE_INFO_REQUEST Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DeviceInfoRequest>
    <DeviceInfoReqType>FullDescription</DeviceInfoReqType>
  </DeviceInfoRequest>
</DMAP>
```

C.5 DEVICE_INFO_RESPONSE

— XML schema structure of DEVICE_INFO_RESPONSE message is shown in Figure C.5.

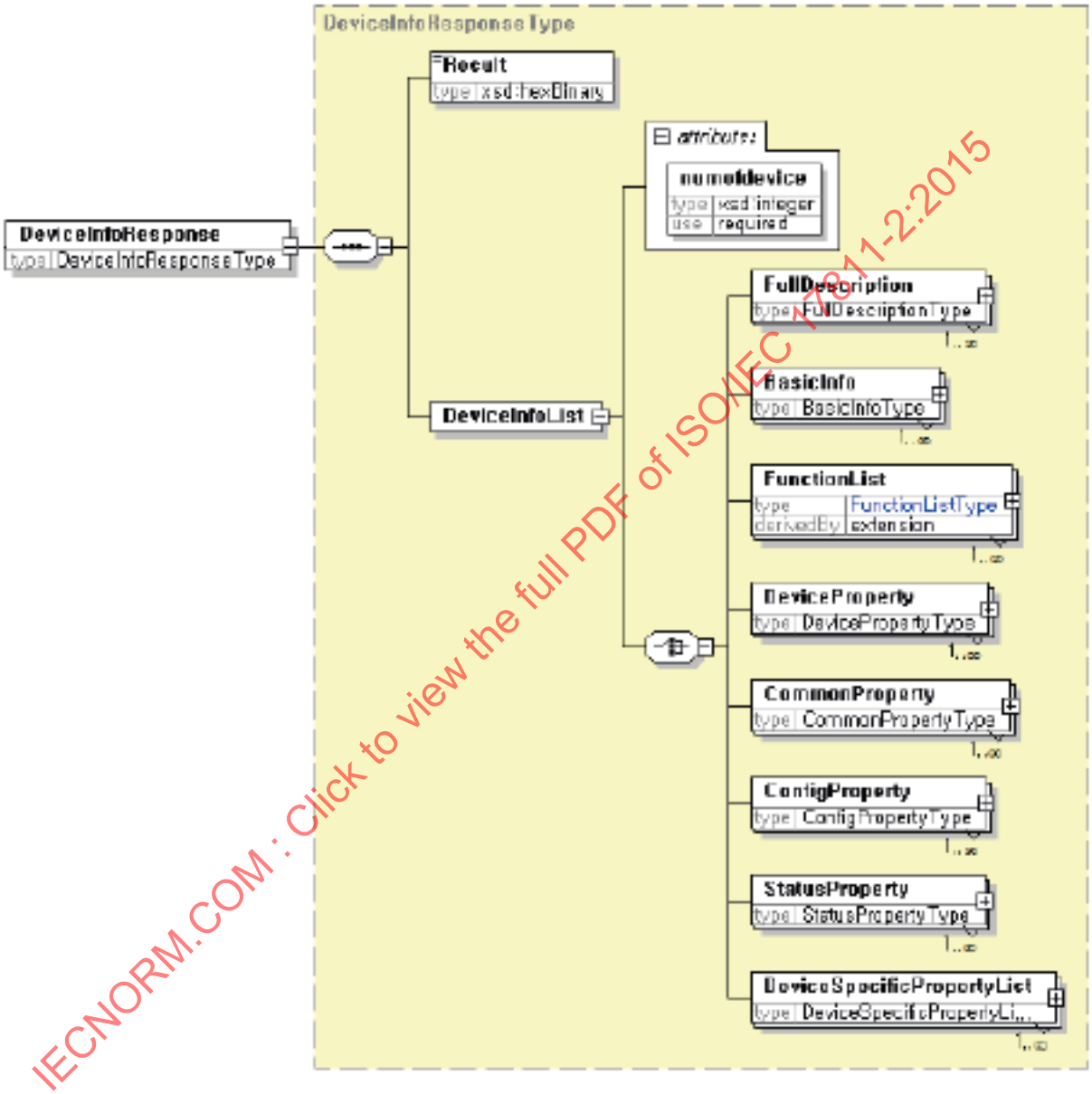


Figure C.5 — Device Info Response Schema Structure

C.5.1 BasicInfo

— XML schema structure of Basic Info Message is shown in Figure C.6 and XML example is shown in Table C- 5, respectively.

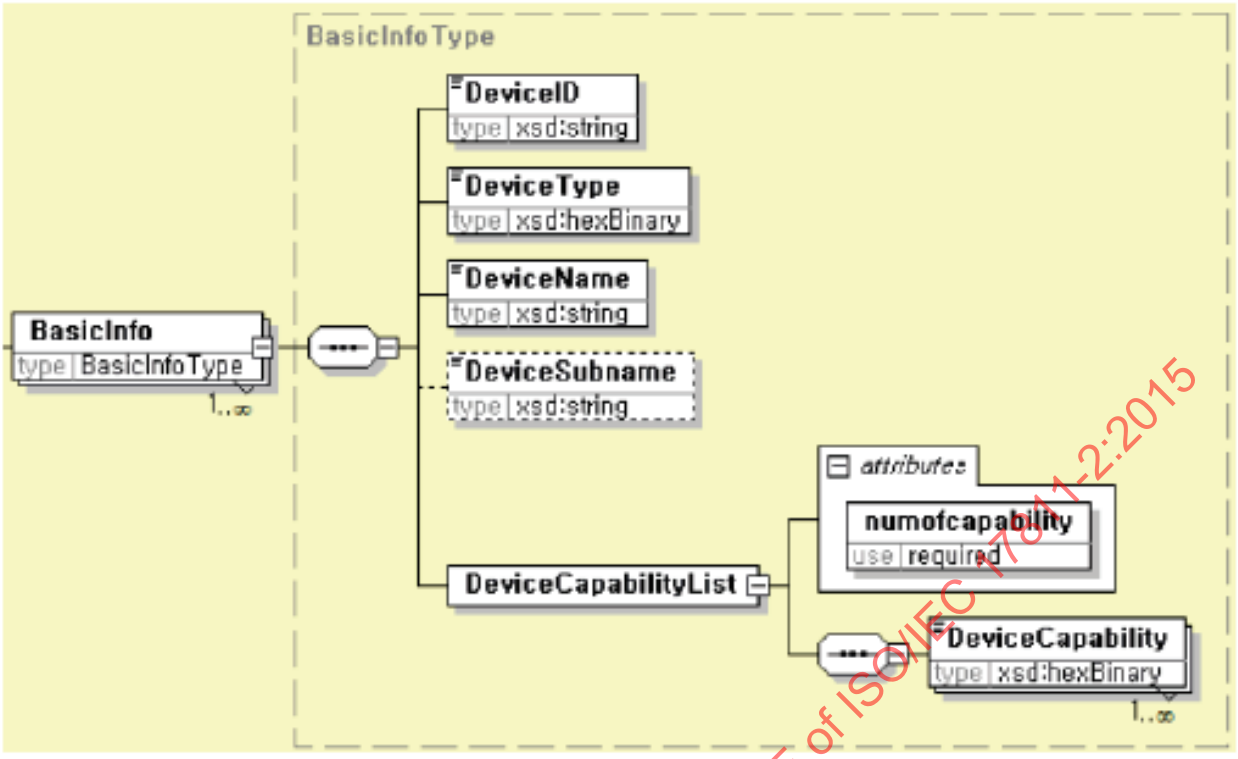


Figure C.6 — Basic Info Message Schema Structure

Table — C-5 — XML Example of Basic Info Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <DeviceInfoResponse>
               <Result>0000</Result>
               <DeviceInfoList numofdevice="1">
                        <BasicInfo>
                                 <DeviceID>1234567890123456789O</DeviceID>
                                 <DeviceType>1605</DeviceType>
                                 <DeviceName> CCTV</DeviceName>
                                 <DeviceCapabilityList numofcapability="1">
                                          <DeviceCapability>1605</DeviceCapabil-
ity>
                                 </DeviceCapabilityList>
                        </BasicInfo>
                        </DeviceInfoList>
      </DeviceInfoResponse>
</DMAP>
```

C.5.2 FunctionList

— XML schema structure of FunctionList Message is shown in Figure C.7.

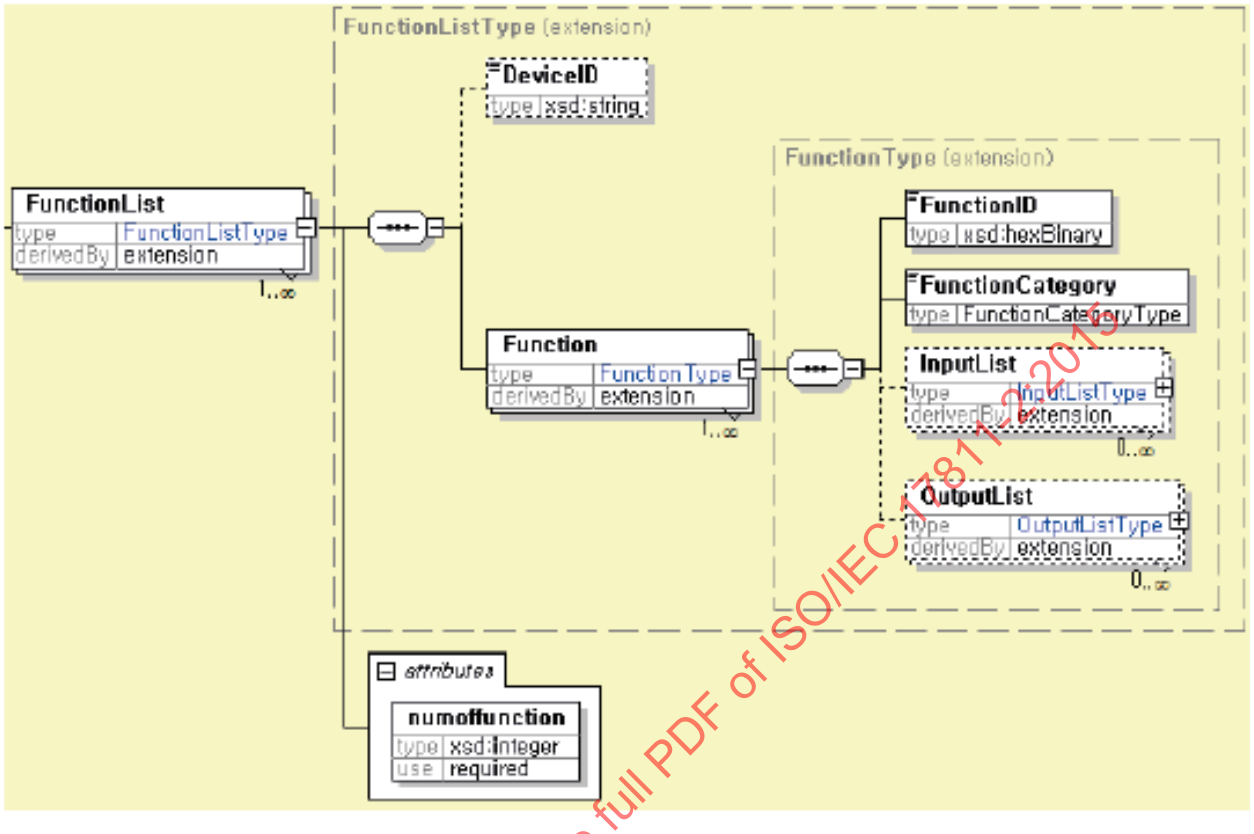


Figure C.7 — FunctionList Message Schema Structure

— XML schema structure of Input/OutList Message is shown in Figure C.8/C.9 and XML example is shown in Table C.6, respectively.

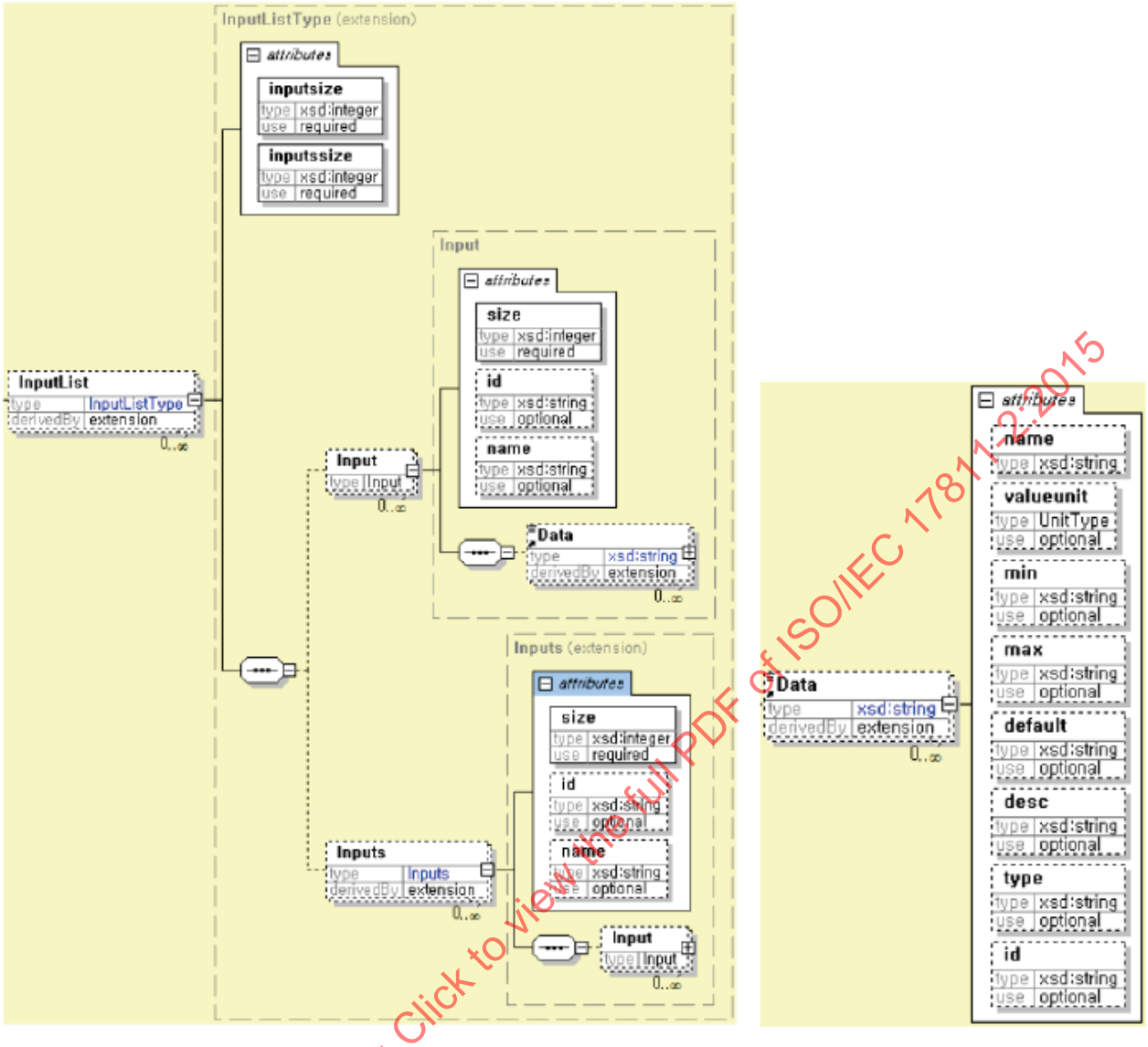


Figure C.8 — InputList Message Schema Structure

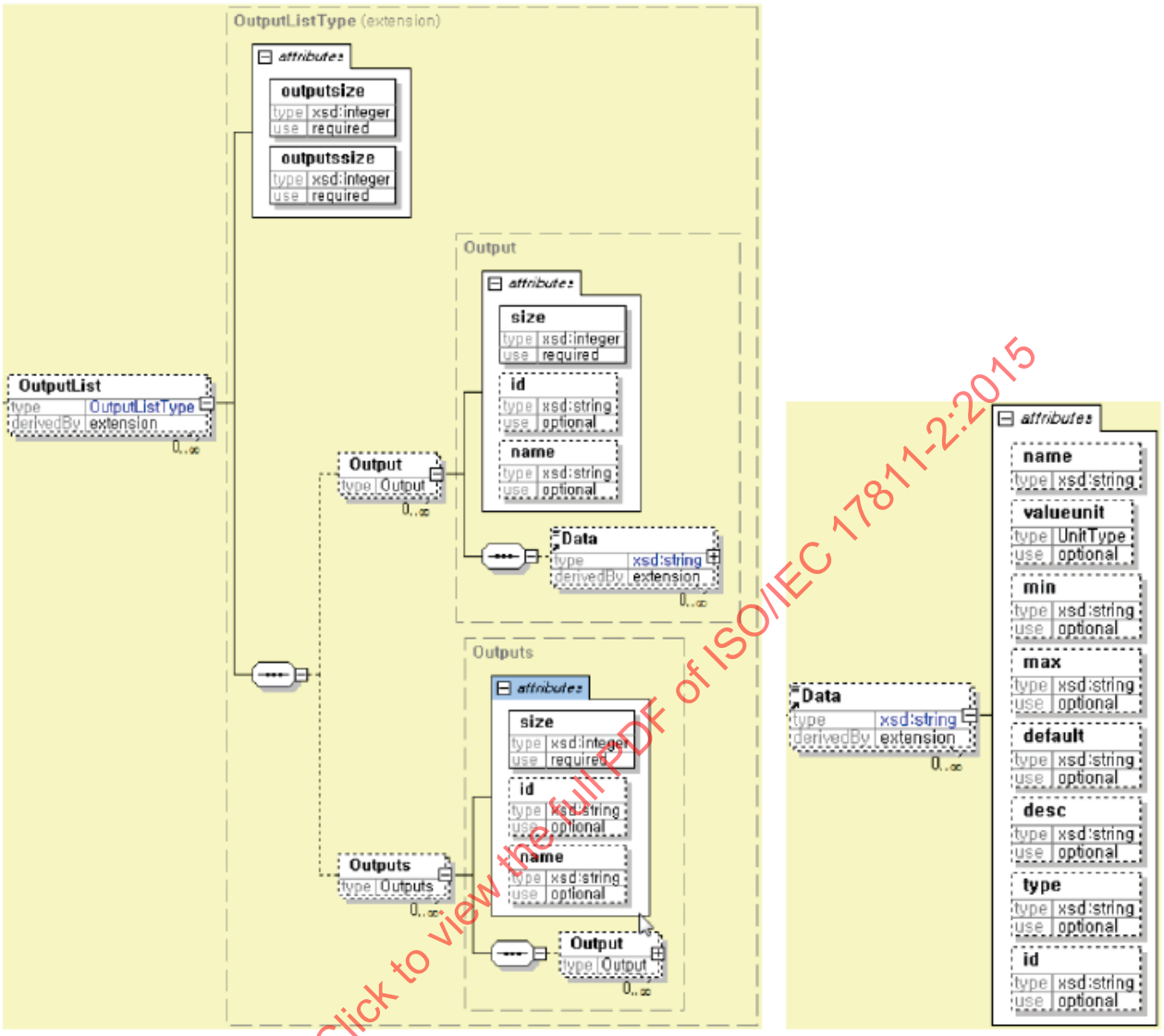


Figure C.9 — OutputList Message Schema Structure

The following XML example shows a device has two functions, and each function has an each input list

Table C.6 — XML Example of FunctionList Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <DeviceInfoResponse>
  <Result>0000</Result>
  <DeviceInfoList numofdevice="1">
   <FunctionList numoffunction="2">
    <Function>
     <FunctionID>15080001</FunctionID>
     <FunctionCategory>Control</FunctionCategory>
     <InputList inputsize="1" inputssize="0">
      <Input size="1">
        <Data name="Status">Ok</Data>
      </Input>
     </InputList>
     <OutputList outputsize="1" outputssize="0">
      <Output size="1">
        <Data name="Status">Ok</Data>
      </Output>
     </OutputList>
    </Function>
    <Function>
     <FunctionID>15080002</FunctionID>
     <FunctionCategory>Control</FunctionCategory>
      <InputList inputsize="1" inputssize="0">
       <Input size="3">
        <Data name="Intensity">5</Data>
        <Data name="Water">5</Data>
        <Data name="Type">Americano</Data>
       </Input>
      </InputList>
      <OutputList outputsize="1" outputssize="0">
       <Output size="1">
        <Data name="Status">Ok</Data>
       </Output>
      </OutputList>
    </Function>
   </FunctionList>
  </DeviceInfoList>
 </DeviceInfoResponse>
</DMAP>
```

C.5.3 DeviceProperty

— XML schema structure of Device Property Message is shown in Figure C.10.

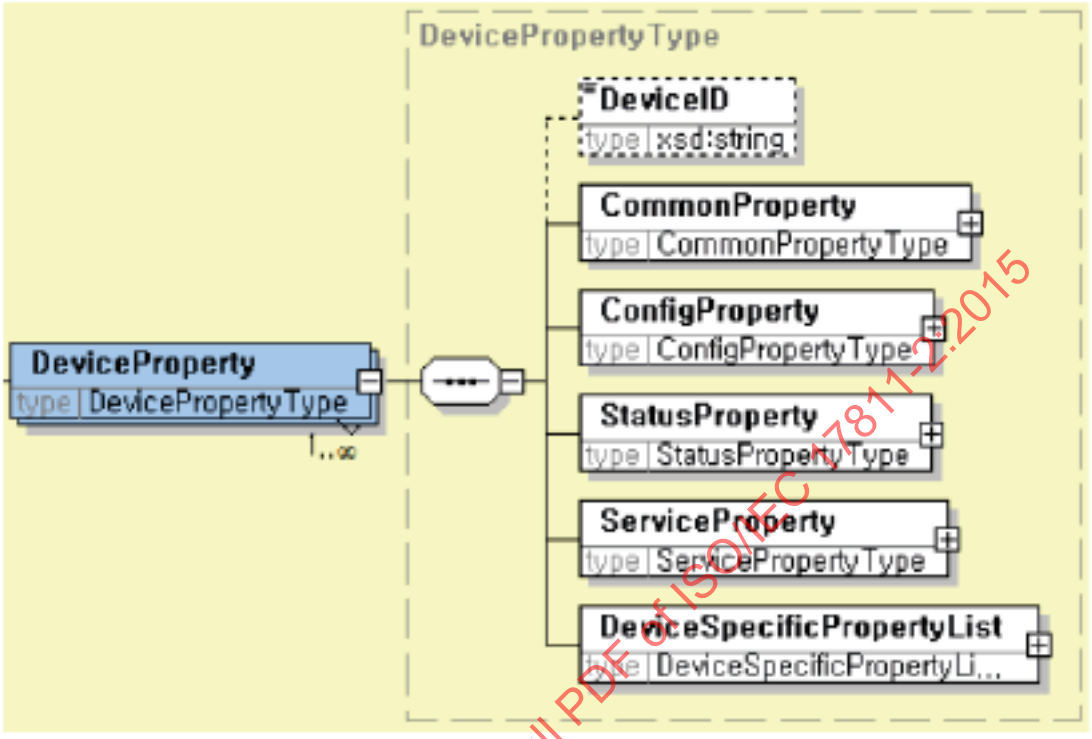


Figure C.10 — Device Property Message Schema Structure

Table C.7 — XML Example of Common Property Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DeviceInfoResponse>
    <Result>0000</Result>
    <DeviceInfoList numofdevice="1">
      <CommonProperty>
        <DeviceID>0123456789AB0101</DeviceID>
        <Model>
          <ModelName>Coffee Machine</ModelName>
          <ModelNumber>CM-01</ModelNumber>
          <SerialNumber>1234567890</SerialNumber>
        </Model>
        <Manufacture>
          <ManufactureName>Example Company</ManufactureName>
          <DistributionDate>2011-12-31T14:00:00</DistributionDate>
          <ManufactureURL>www.example.co.kr</ManufactureURL>
        </Manufacture>
```

Table C.7 — *(continued)*

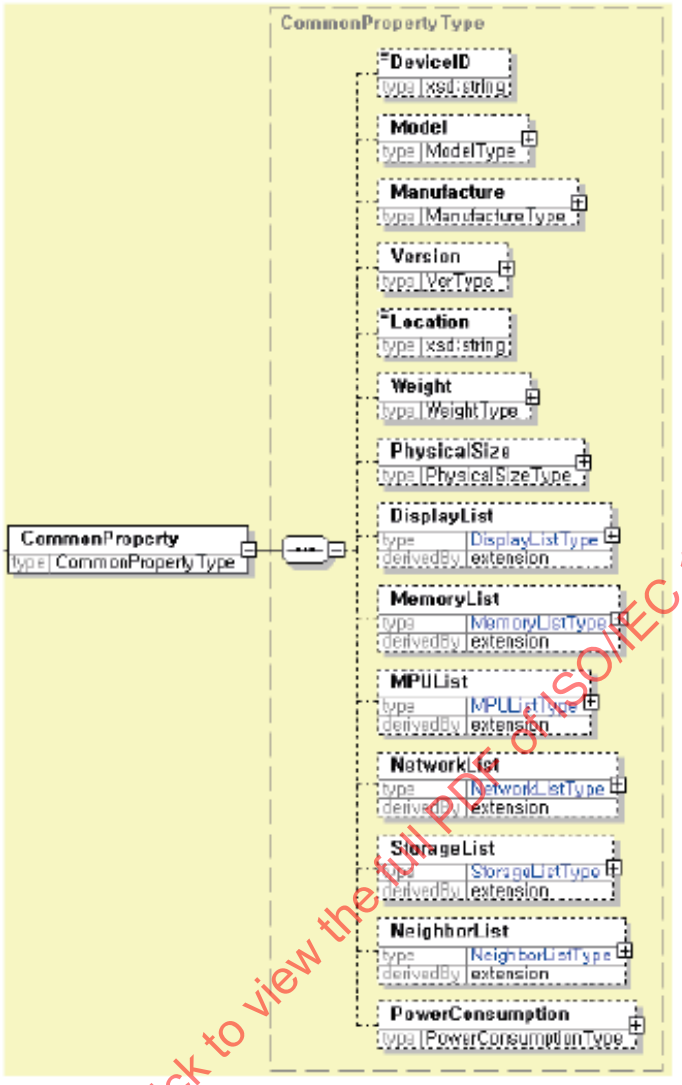
```
<Version>
  <VersionInfoURL>versioninfo.example.co.kr</VersionInfoURL>
  <HardwareVersion>0.1.0</HardwareVersion>
  <Firmware>
    <Version>0.3.0</Version>
    <FileName>firmware.bin</FileName>
    <FTPURL>.</FTPURL>
    <FTPID>id</FTPID>
    <FTPPassword>password</FTPPassword>
  </Firmware>
</Version>
<DisplayList numofdisplay="1">
  <Display>
    <DisplayID>12345678901234567890</DisplayID>
    <Resolution>1024x768</Resolution>
    <DisplaySize>15</DisplaySize>
    <PannelType>LCD</PannelType>
  </Display>
</DisplayList>
<MemoryList numofmemory="1">
  <Memory>
    <MemoryID>12345678901234567891</MemoryID>
    <MemoryDescription> ExampleB DDR3 64000</MemoryDescription>
  </Memory>
</MemoryList>
<MPUList numofMPU="1">
  <MPU>
    <MPUID>12345678901234567892</MPUID>
    <MPUDescription> ExampleC </MPUDescription>
  </MPU>
</MPUList>
<NetworkList numofnetwork="2">
  <Network>
    <InterfaceID>12345678901234567890</InterfaceID>
    <MacAddress>1234545656</MacAddress>
    <InterfaceType>Ethernet</InterfaceType>
  </Network>
  <Network>
    <InterfaceID>12345678901234567877</InterfaceID>
```

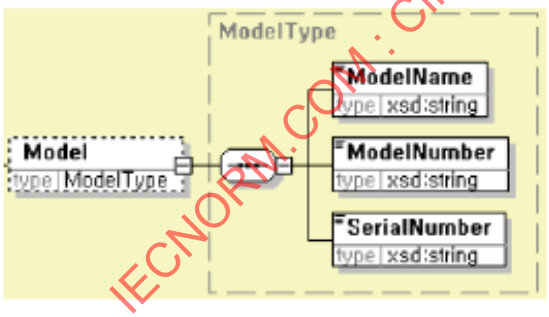
Table C.7 — *(continued)*

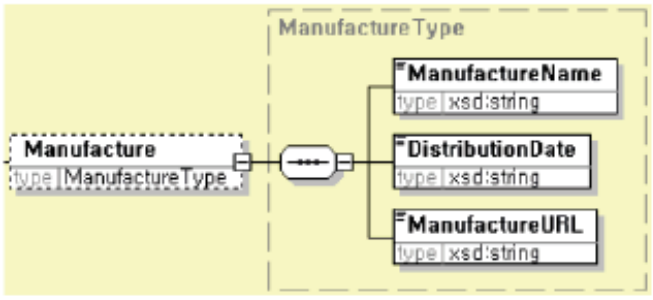
```
            <MacAddress>1234545657</MacAddress>
            <InterfaceType>RS485</InterfaceType>
          </Network>
          <Network>
            <InterfaceID>12345678901234567877</InterfaceID>
            <MacAddress>1234545657</MacAddress>
            <InterfaceType>RS485</InterfaceType>
          </Network>
        </NetworkList>
        <StorageList numofstorage=”1”>
          <Storage>
            <StorageID>12345678901234567866</StorageID>
            <StorageDescription> ExampleC HDD 7200P </StorageDescription>
          </Storage>
        </StorageList>
        <PoweConsumption>
          <ConsumptionValue>123</ConsumptionValue>
          <Unit>0005</Unit>
        </PowerConsumption>
        </CommonProperty>
    </DeviceInfoList>
  </DeviceInfoResponse>
</DMAP>
```

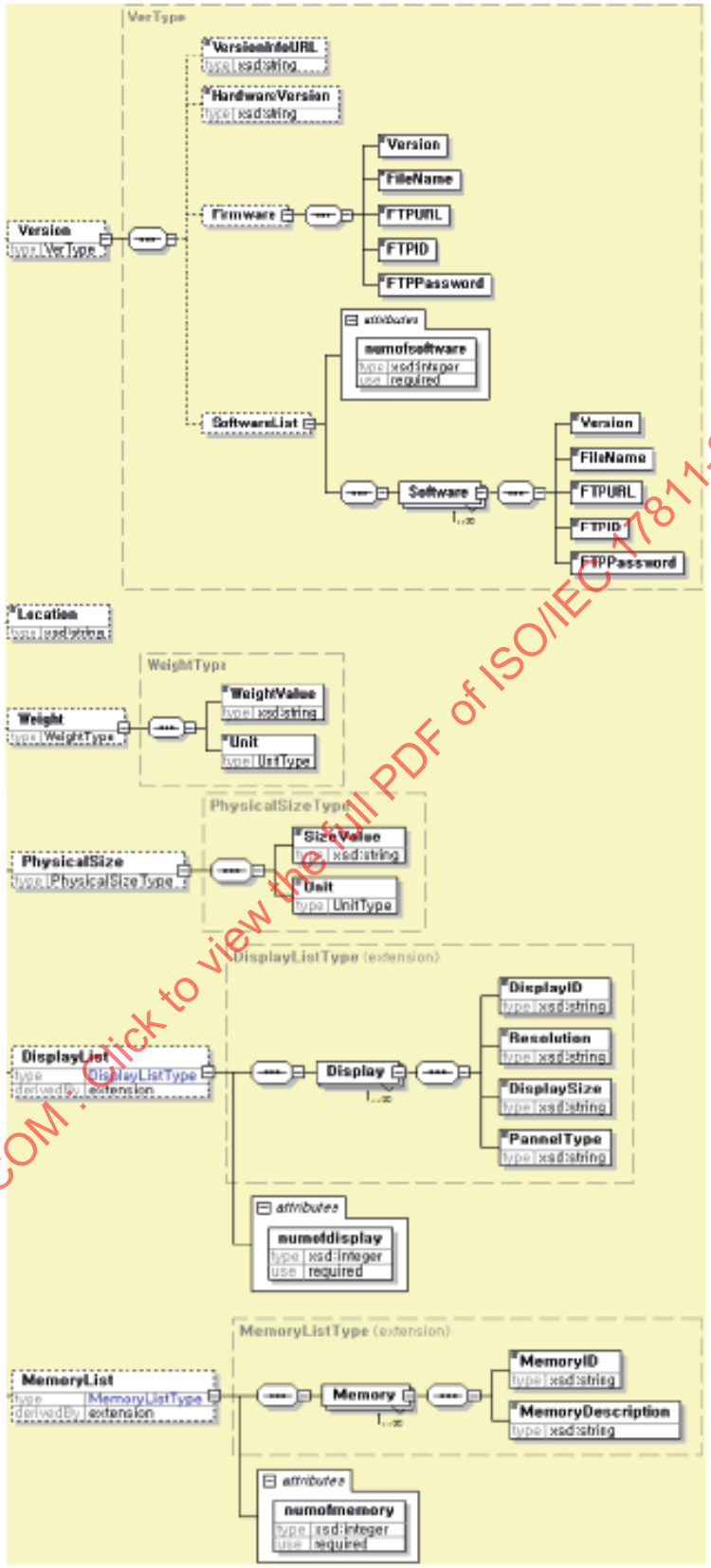
CommonProperty

— XML schema structure of CommonProperty Message is shown in Figure C.11.









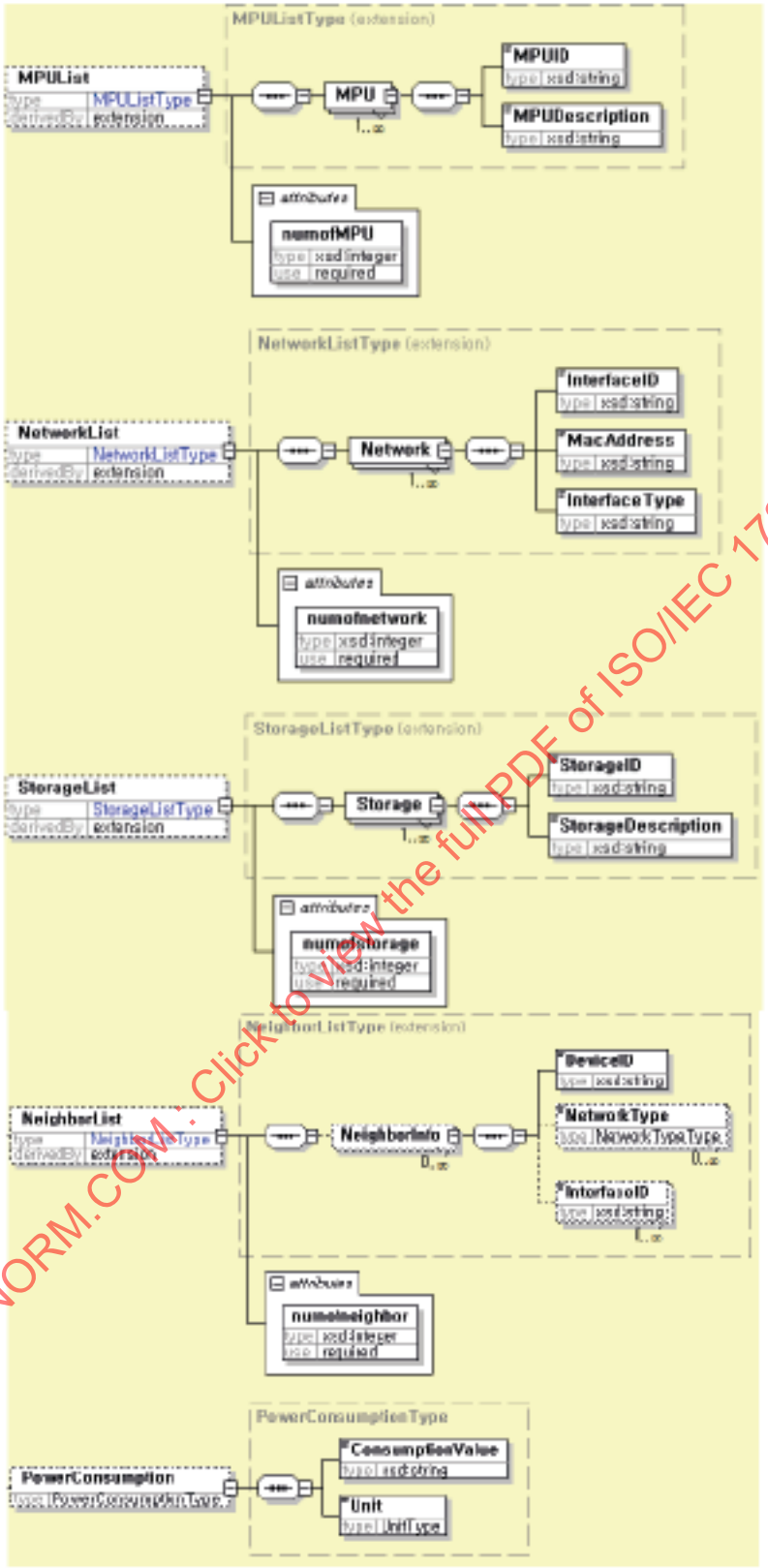


Figure — C-11 — Common Property Message Schema Structure

— The following XML example shows the various information of device, which is described by 'CommonProperty'.

ConfigProperty

— XML schema structure of 'ConfigProperty' Message is shown in Figure C.12.

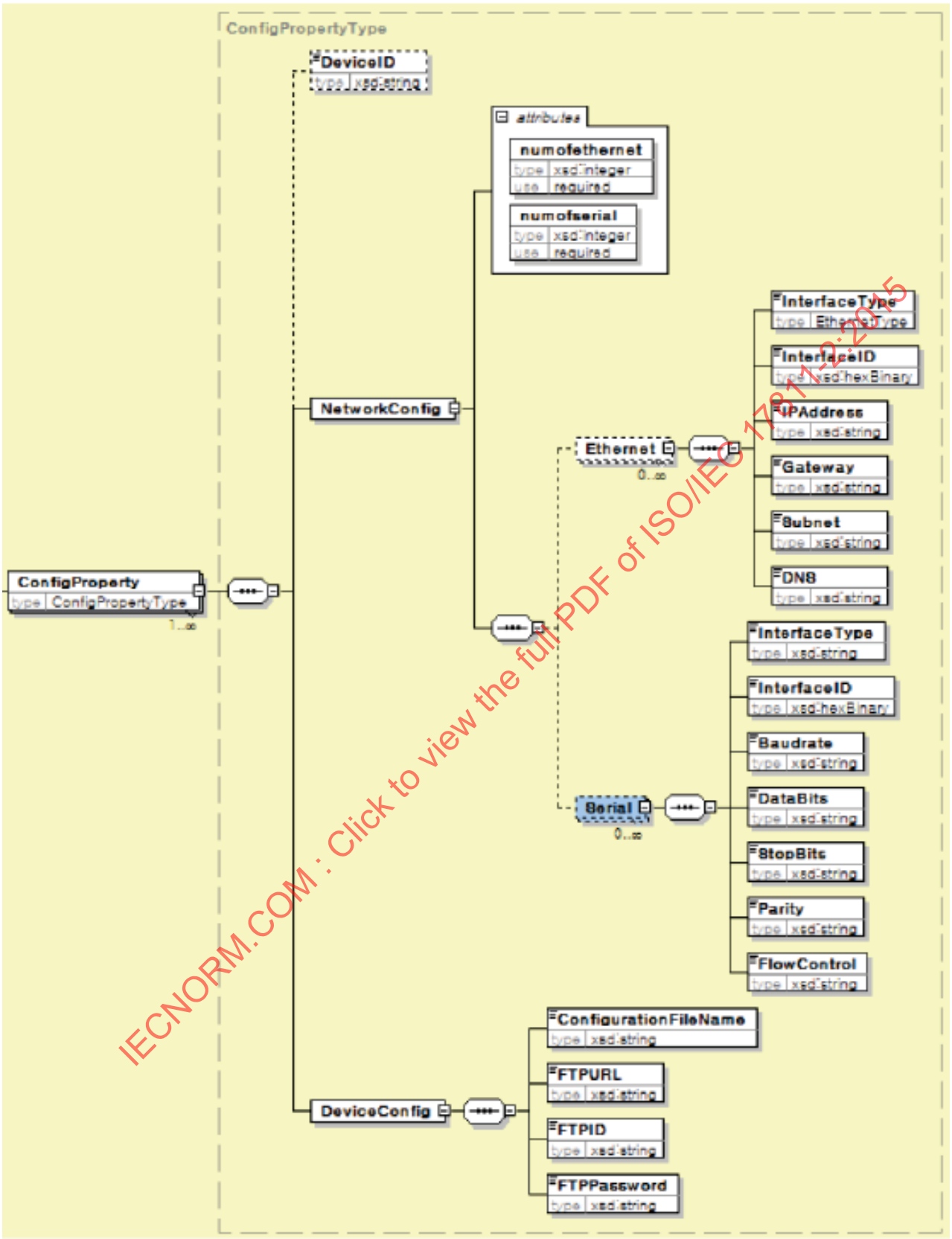


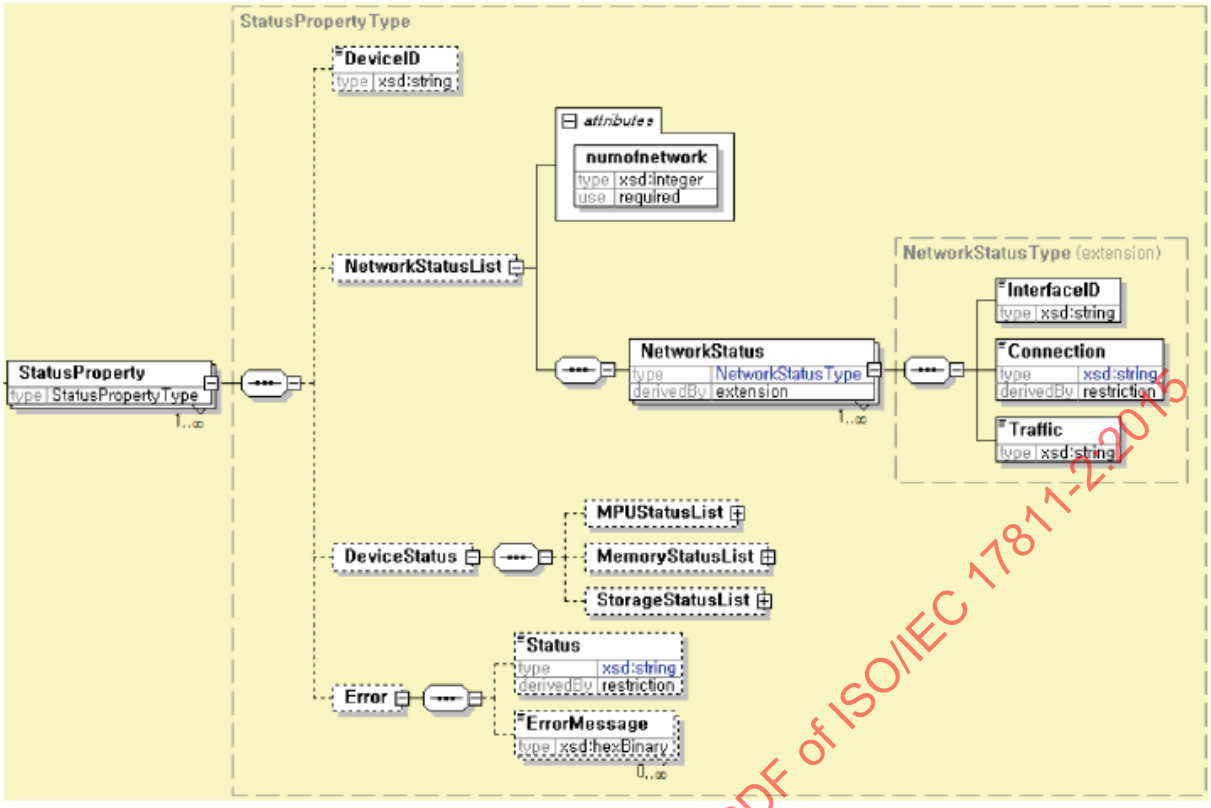
Figure C.12 — Config Property Message Schema Structure

Table C.8 — XML Example of Config Property Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DeviceInfoResponse>
  <Result>0000</Result>
    <DeviceInfoList numofdevice="1">
      <ConfigProperty>
        <NetworkConfig numofethernet="1" numofserial="1">
          <Ethernet>
            <InterfaceType>IPV4</InterfaceType>
            <InterfaceID>1123456789AB0104</InterfaceID>
            <IPAddress>192.168.1.107</IPAddress>
            <Gateway>192.168.1.1</Gateway>
            <Subnet>255.255.255.0</Subnet>
            <DNS>129.254.15.15</DNS>
          </Ethernet>
          <Serial>
            <InterfaceType>RS232</InterfaceType>
            <InterfaceID>1123456789AB0105</InterfaceID>
            <Baudrate>2400</Baudrate>
            <DataBits>1</DataBits>
            <StopBits>1</StopBits>
            <Parity>1</Parity>
            <FlowControl>no</FlowControl>
          </Serial>
        </NetworkConfig>
        <DeviceConfig>
          <ConfigurationFileName>config.sh</ConfigurationFileName>
          <FTPURL>.</FTPURL>
          <FTPID>id</FTPID>
          <FTPPassword>password</FTPPassword>
        </DeviceConfig>
      </ConfigProperty>
    </DeviceInfoList>
  </DeviceInfoResponse>
</DMAP>
```


StatusProperty

— XML schema structure of 'StatusProperty' Message is shown in Figure C.13.



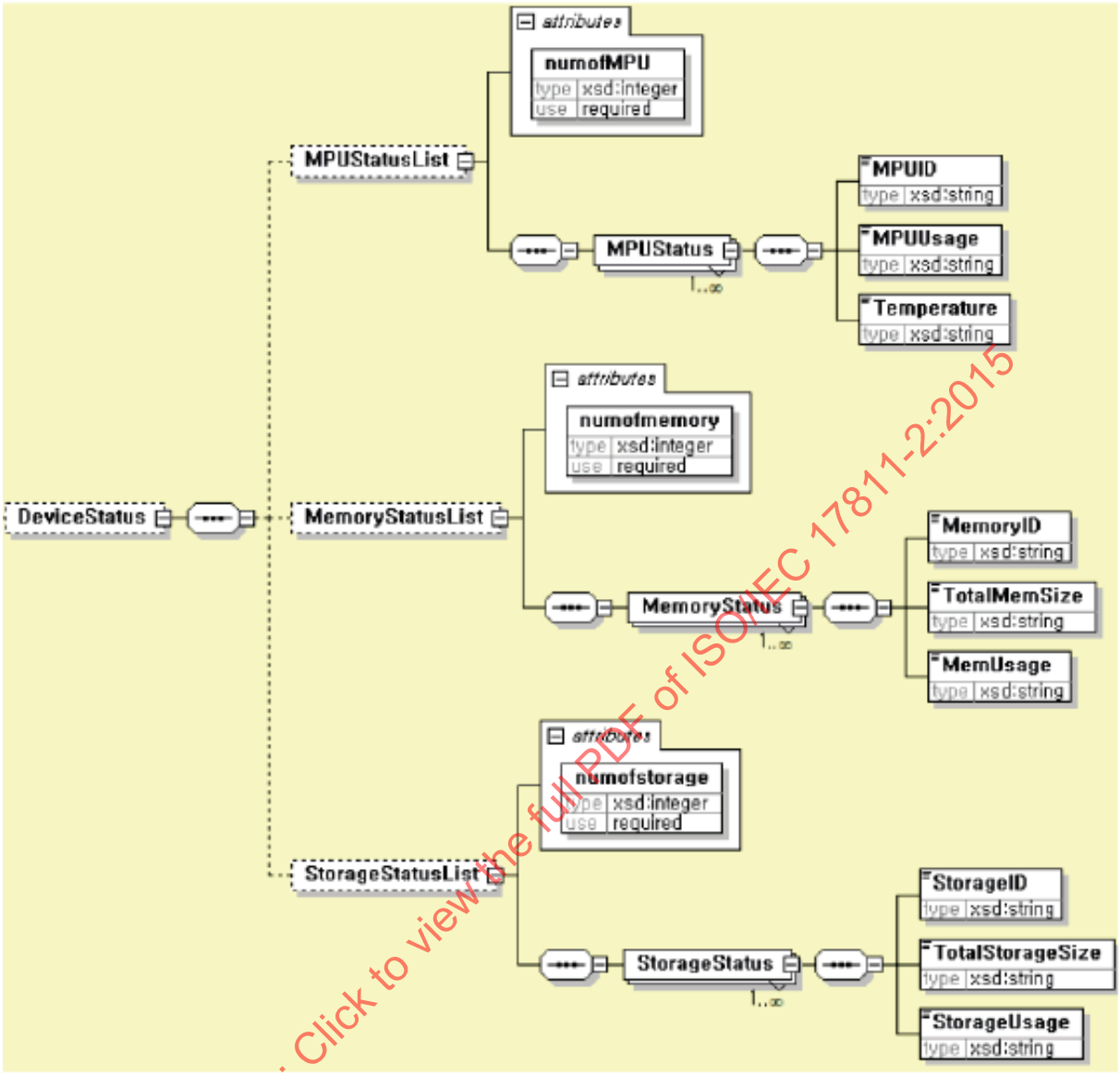


Figure C.13 — Status Property Message Schema Structure

— The following XML example shows the device status information such as ‘network connectivity’, ‘traffic’, ‘MPU usage’ and so on

Table C.9 — XML Example of Status Property Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DeviceInfoResponse>
    <Result>0000</Result>
    <DeviceInfoList numofdevice="1">
      <StatusProperty>
        <NetworkStatusList numofnetwork="1">
          <NetworkStatus>
            <InterfaceID>00000000000000000001</InterfaceID>
            <Connection>Online</Connection>
            <Traffic>27</Traffic>
          </NetworkStatus>
        </NetworkStatusList>
        <DeviceStatus>
          <MPUStatusList numofMPU="1">
            <MPUStatus>
              <MPUID>10000000000000000000</MPUID>
              <MPUUsage>80</MPUUsage>
              <Temperature>270</Temperature>
            </MPUStatus>
          </MPUStatusList>
        </DeviceStatus>
          <Error>
            <Status>Normal</Status>
            <ErrorMessage>0000</ErrorMessage>
          </Error>
      </StatusProperty>
    </DeviceInfoList>
  </DeviceInfoResponse>
</DMAP>
```


DeviceSpecificPropertyList

— XML schema structure of 'DeviceSpecificProperty' Message is shown in Figure C.14.

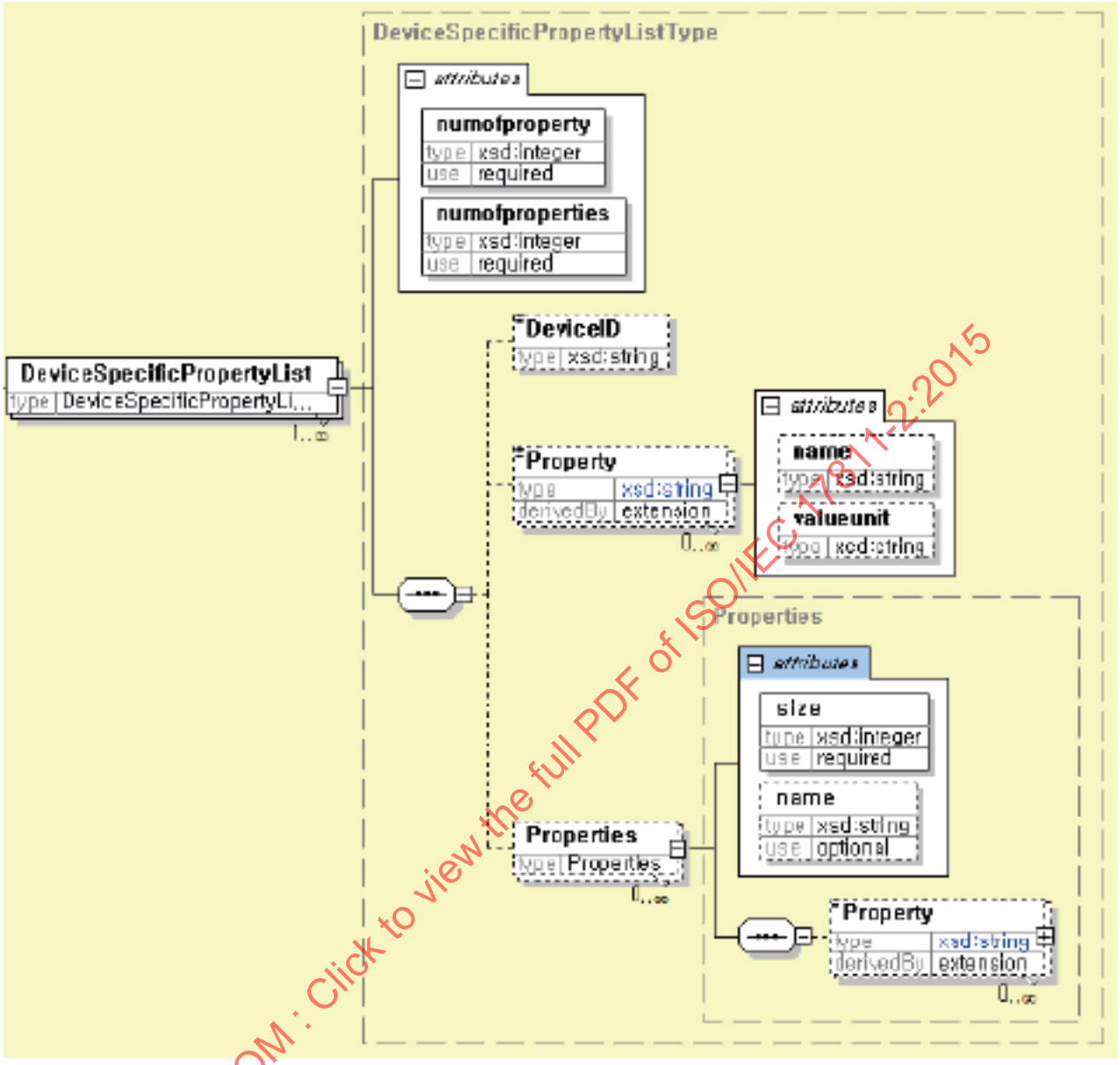


Figure C.14 — Status Property Message Schema Structure

— The following XML example shows the device specific properties about the CCTV device. The focal length of this CCTV device is 16 meters and view angle is 30 degrees.

Table C.10 — XML Example of Device Specific Property Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DeviceInfoResponse>
    <Result>0000</Result>
    <DeviceInfoList numofdevice="1">
      <DeviceSpecificPropertyList numofproperties="0" numofproperty="2">
                  <Property name="FocalLength">16</Property>
                  <Property name="ViewAngle">30</Property>
      </DeviceSpecificPropertyList>
    </DeviceInfoList>
  </DeviceInfoResponse>
</DMAP>
```

C.6 DEVICE_CONTROL_REQUEST

— XML schema structure of DEVICE_CONTROL_REQUEST is shown in Figure C.15 and message structure is shown in Table C- 11, respectively.

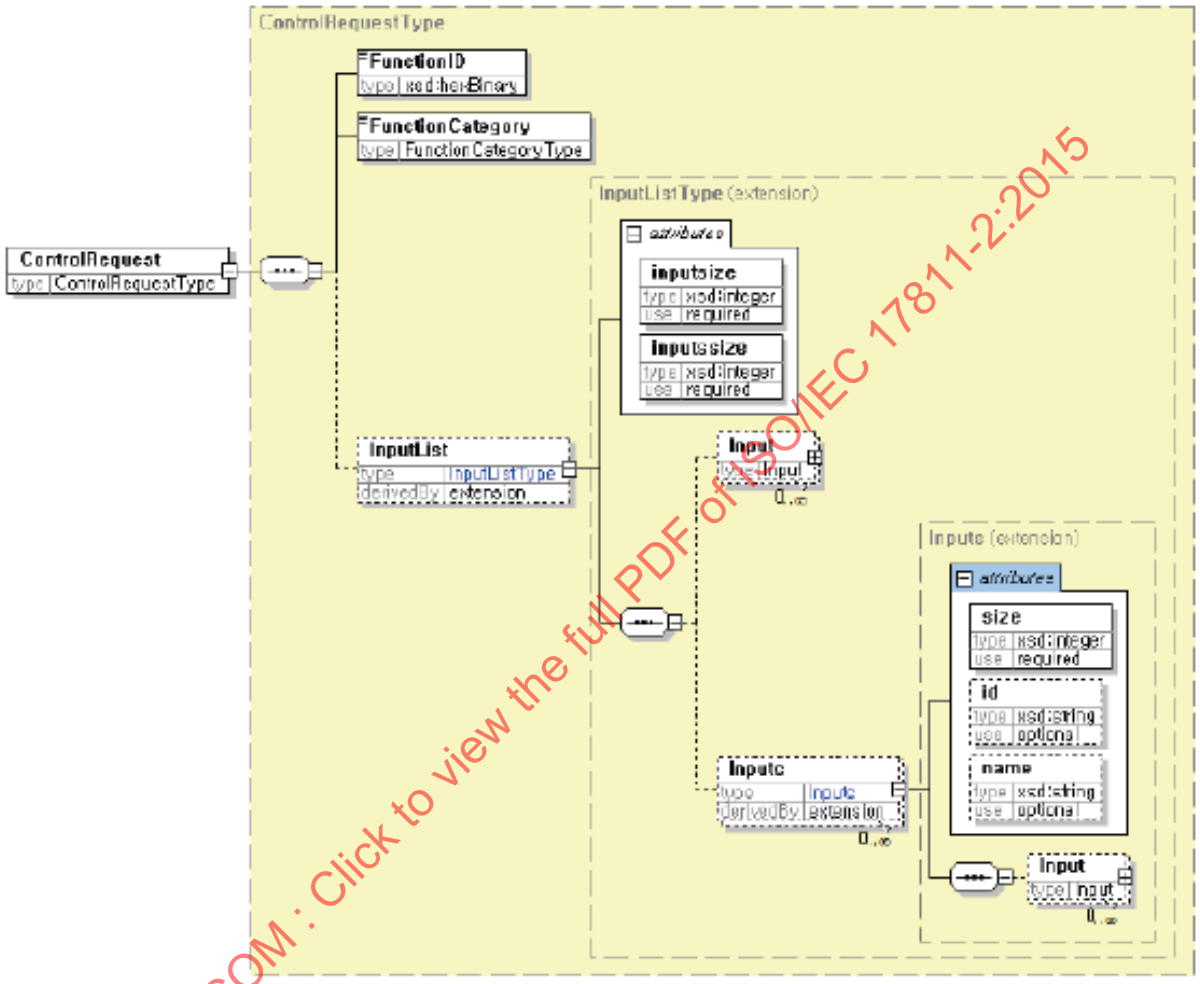


Figure C.15 — Device Control Request Schema structure

— The following XML example shows the control request message for light device control.

Table C.11 — XML Example of DEVICE_CONTROL_REQUEST Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP>
       <ControlRequest>
                 <FunctionID>11071102</FunctionID>
                 <FunctionCategory>Control</FunctionCategory>
                 <InputList inputsize="1" inputssize="0">
                     <Input size="4">
                              <Data name="ControlType">RS485</Data>
<Data name="SubID">11</Data>
                              <Data name="DimmingLevel">5</Data>
                              <Data name="OnOff">On</Data>
                     </Input>
                 </InputList>
       </ControlRequest>
</DMAP>
```

C.7 DEVICE_CONTROL_RESPONSE

— XML schema structure of DEVICE_CONTROL_RESPONSE is shown in Figure C.16 and message structure is shown in Table C- 12, respectively.

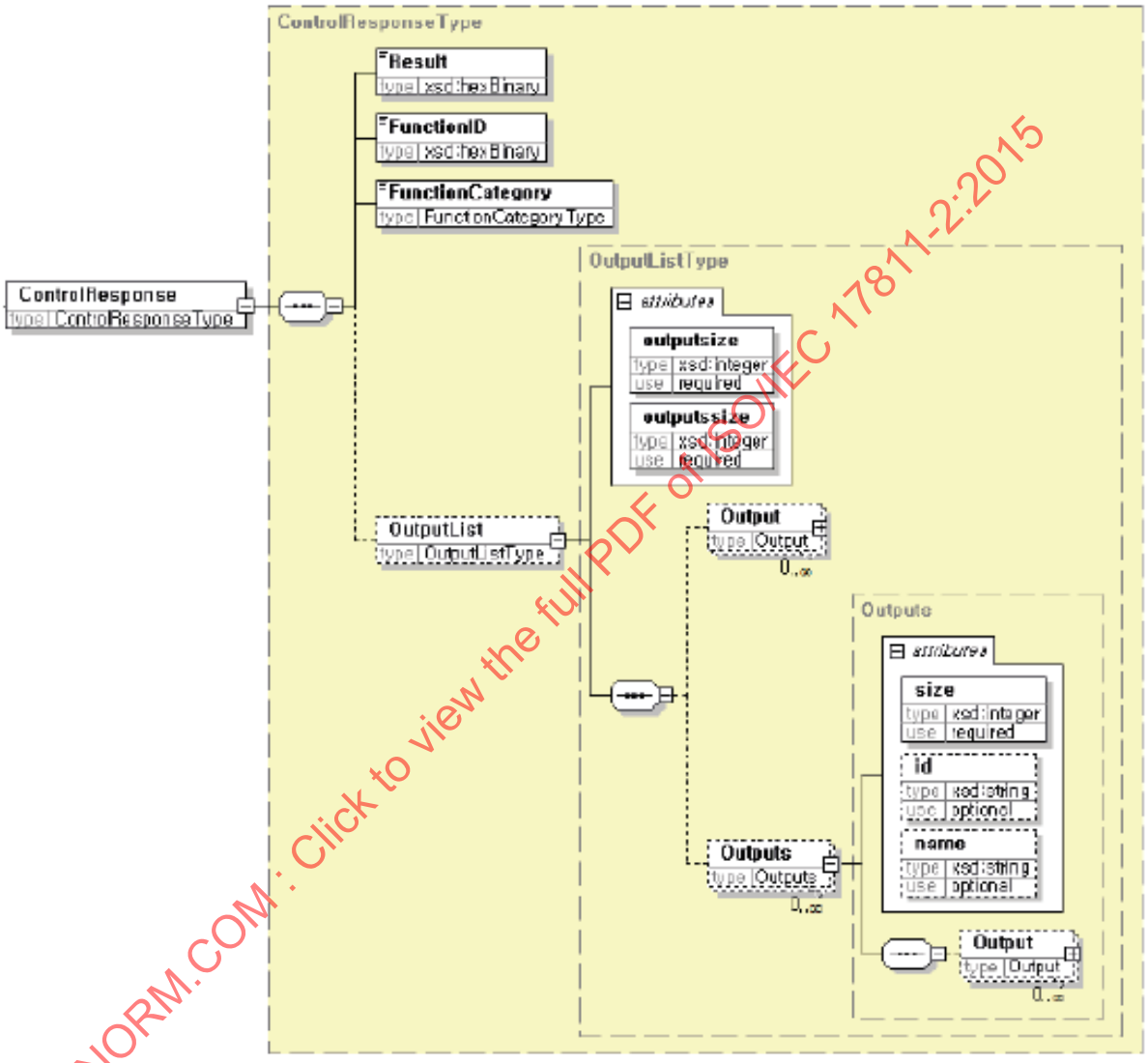


Figure C.16 — Device Control Response Schema Structure

— The following XML example shows the control response message for light device control request.

Table C.12 — XML Example of DEVICE_CONTROL_RESPONSE Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP>
       <ControlResponse>
                 <Result>0000</Result>
                 <FunctionID>11071102</FunctionID>
                 <FunctionCategory>Control</FunctionCategory>
                 <OutputList outputsize="1" outputssize="0">
                    <Output size="3">
                           <Data name="OnOffStatus">On</Data>
                           <Data name="DimmingLevel">5</Data>
                           <Data name="DimmingFunction">1</Data>
                 </Output>
       </OutputList>
       </ControlResponse>
</DMAP>
```

C.8 EVENT_NOTIFICATION

— XML schema structure of EVENT_NOTIFICATION is shown in Figure C.17 and message structure is shown in Table C- 13, respectively.

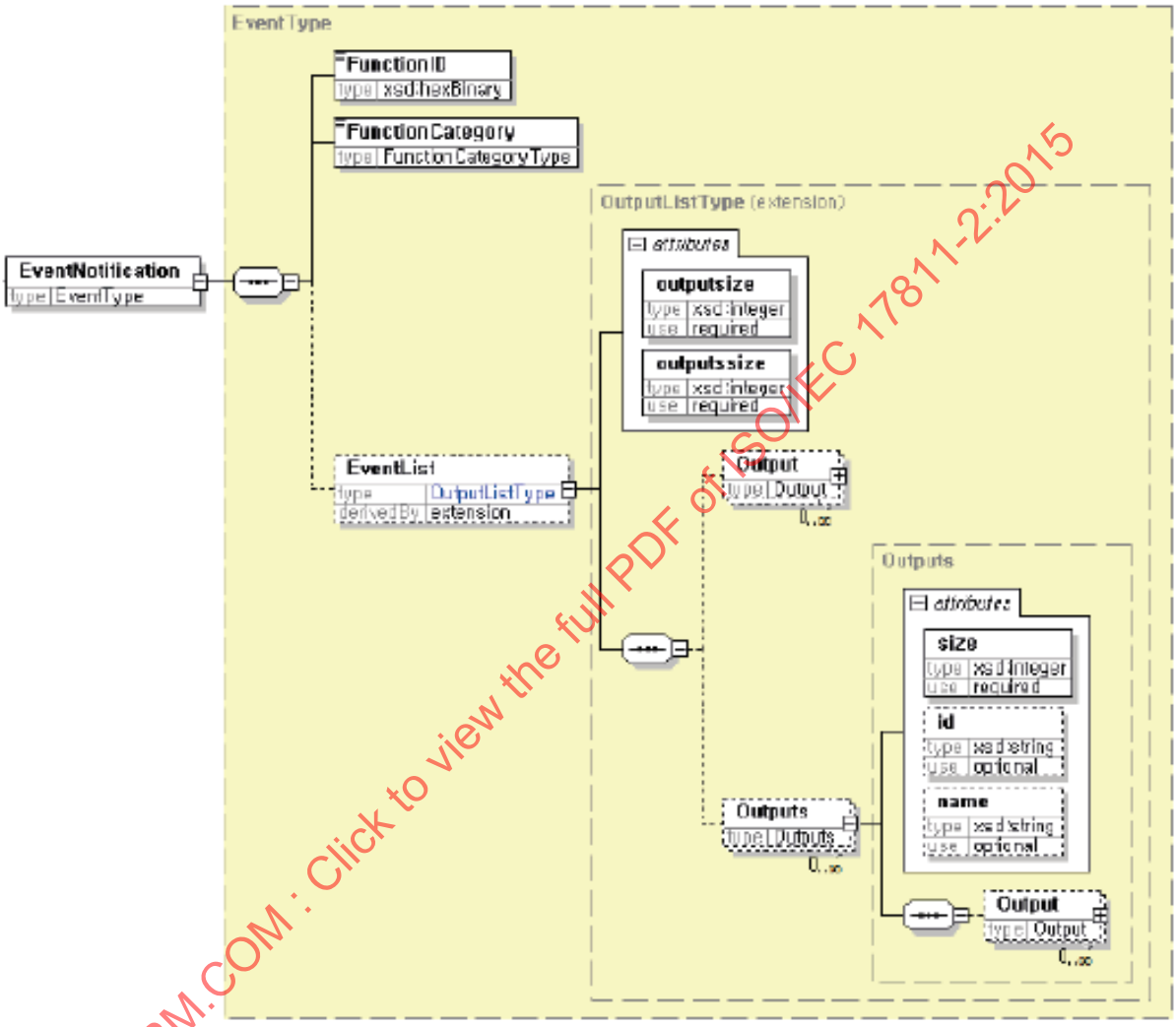


Figure C.17 — Event Notification Schema Structure

— The following XML example shows the event message of gas sensor (report aliveness).

Table C.13 — XML Example of Event Notification Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <EventNotification>
                <FunctionID>1107F000</FunctionID>
                <FunctionCategory>Event</FunctionCategory>
                <EventList outputsize="1" outputssize="0">
                        <Output size="1">
                                <Data name="Alive">1</Data>
                        </Output>
                </EventList>
        </EventNotification>
</DMAP>
```

C.9 EVENT_SUBSCRIPTION_REQUEST

— XML schema structure of EVENT SUBSCRIPTION REQUEST Message is shown in Figure

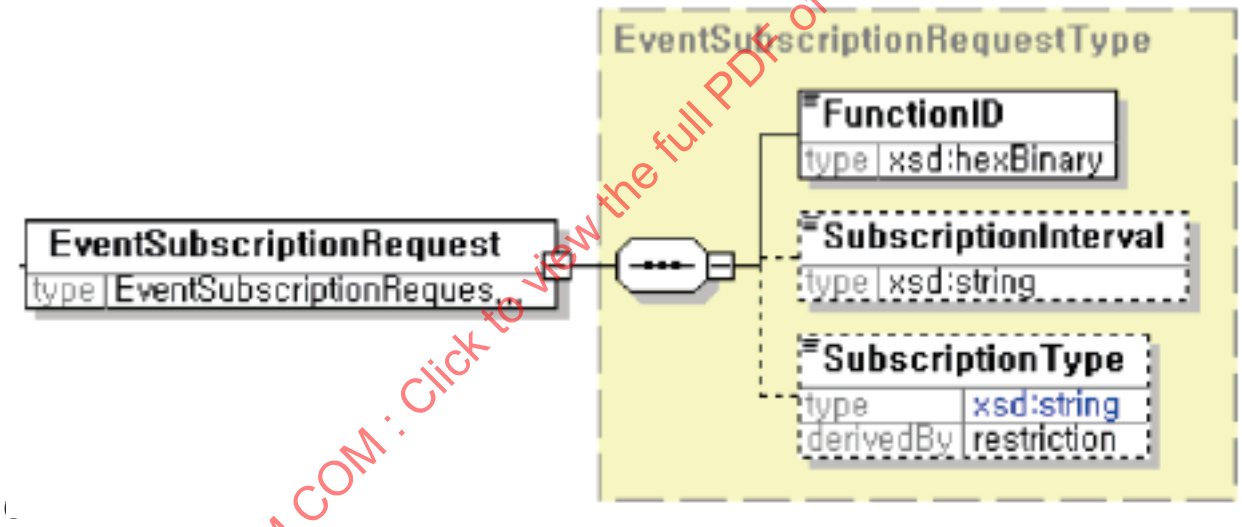


Figure C.18 — Event Subscription Request Schema Structure

— The following XML example shows the event subscription message for the gas sensor. (first registration and interval time is 1sec)

Table C.14 — XML Example of Event Subscription Request Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <EventSubscriptionRequest>
       <FunctionID>14020001</FunctionID>
       <SubscriptionInterval>1000</SubscriptionInterval>
       <SubscriptionType>Registration</SubscriptionType>
   </EventSubscriptionRequest>
</DMAP>
```

C.10 EVENT_SUBSCRIPTION_RESPONSE

— XML schema structure of EVENT_SUBSCRIPTION_RESPONSE Message is shown in Figure C.19.

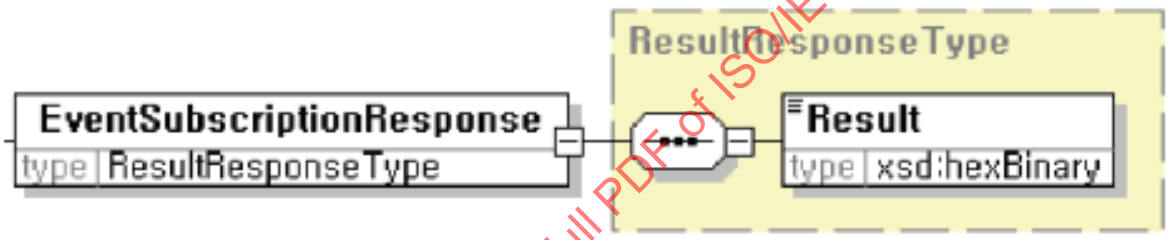


Figure C.19 — Event Subscription Response Schema Structure

— The following XML example shows the event subscription request is performed with success.

Table C.15 — XML Example of Event Subscription Response Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <EventSubscriptionResponse>
    <Result>0000</Result>
   </EventSubscriptionResponse>
</DMAP>
```

C.11 GET_FILEINFO_REQUEST

— XML schema structure of GET_FILEINFO_REQUEST Message is shown in Figure C.20.

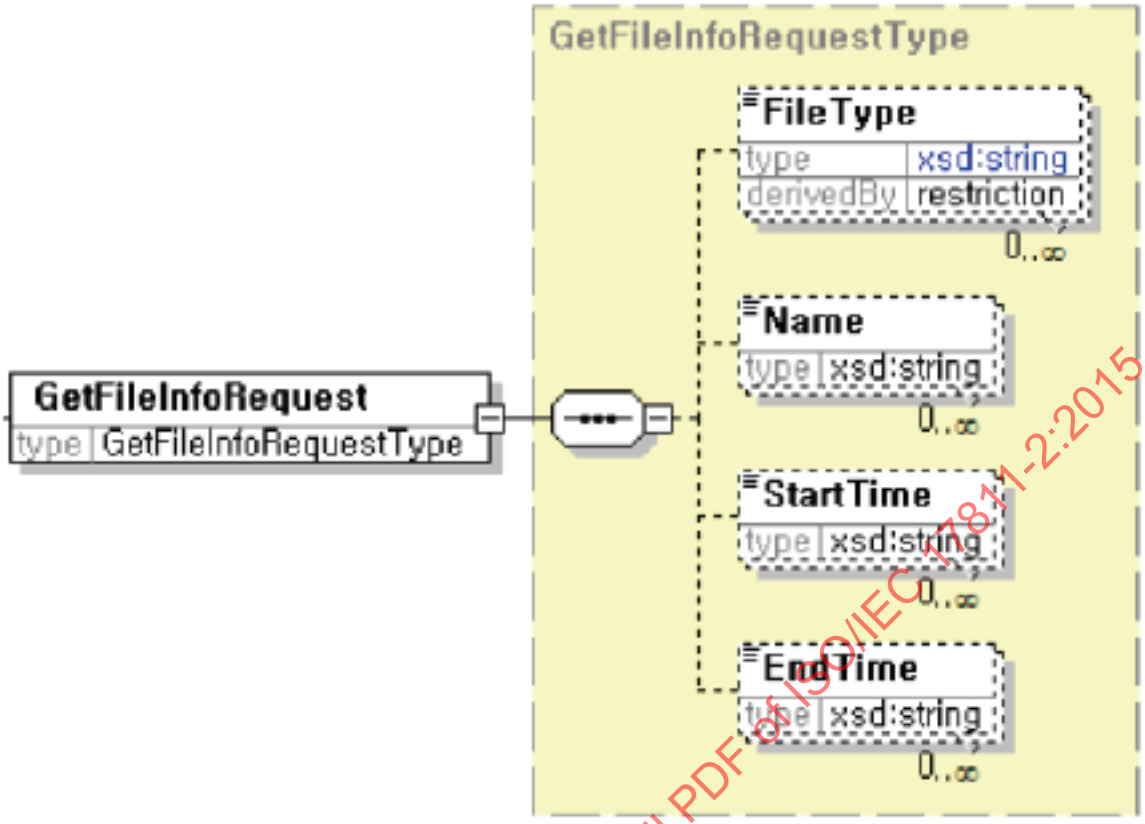


Figure C.20 — Get FileInfo Request Schema Structure

— The following XML example shows the get file information request message with searching condition.

Table C.16 — XML Example of Get FileInfo Request Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <GetFileInfoRequest>
        <FileType>Firmware</FileType>
        <StartTime>2011-12-31T14:00:00</StartTime>
        <EndTime>2011-12-31T15:00:00</EndTime>
   </GetFileInfoRequest>
</DMAP>
```


C.12 GET_FILEINFO_RESPONSE

— XML schema structure of GET_FILEINFO_RESPONSE Message is shown in Figure C.21.

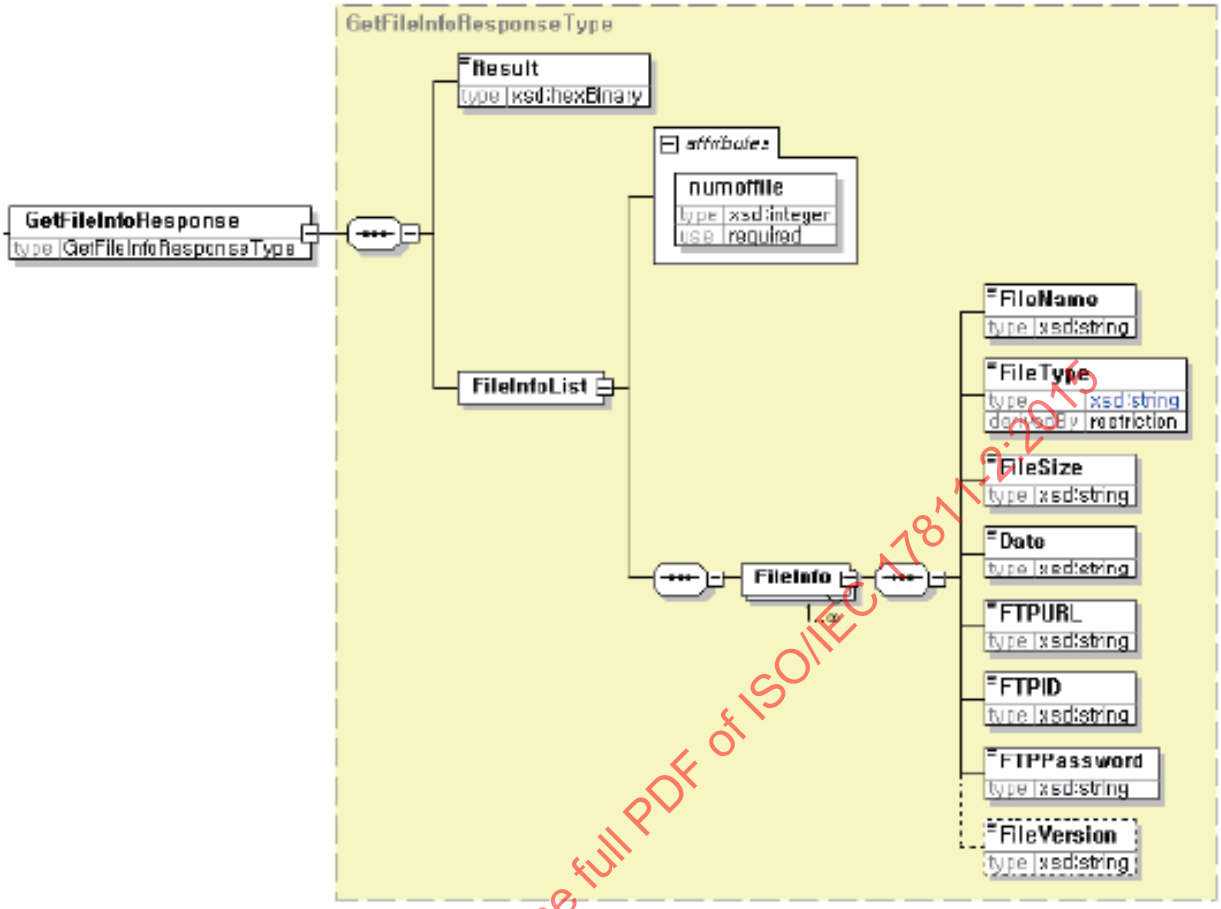


Figure C.21 — GetFileInfo Response Schema Structure

— The following XML example shows the file information.

Table C.17 — XML Example of Get FileInfo Response Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <GetFileInfoResponse>
  <Result>0000</Result>
  <FileInfoList numoffile="1">
   <FileInfo>
    <FileName>ExampleA</FileName>
    <FileType>Firmware</FileType>
    <FileSize>25.3</FileSize>
    <Date>2011-12-31T14:00:00</Date>
    <FTPURL>.</FTPURL>
    <FTPID>id</FTPID>
    <FTPPassword>password</FTPPassword>
   </FileInfo>
  </FileInfoList>
 </GetFileInfoResponse>
</ DMAP >
```

C.13 GET_FILE_REQUEST

— XML schema structure of GET_FILE_REQUEST Message is shown in Figure C.22.

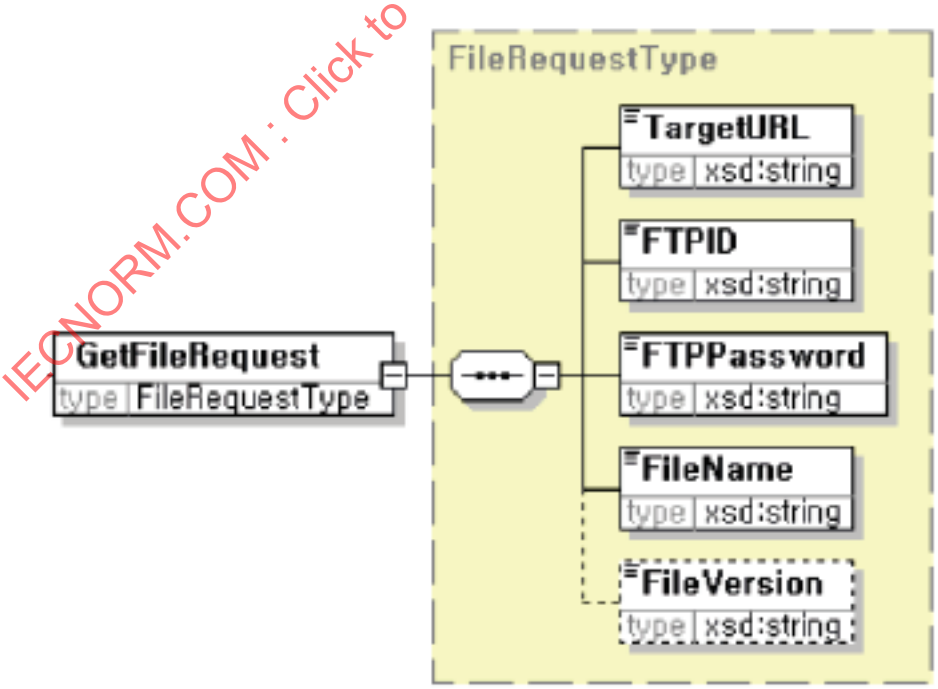


Figure C.22 — Get File Request Schema Structure

— The following XML example shows the get file request message including the FTP information.

Table C.18 — XML Example of Get File Request Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <GetFileRequest>
            <TargetURL>www.test.co.kr</TargetURL>
            <FTPID>.</FTPID>
            <FTPPassword>password</FTPPassword>
            <FileName>test</FileName>
   </GetFileRequest>
</DMAP>
```

C.14 GET_FILE_RESPONSE

— XML schema structure of GET_FILE_RESPONSE Message is shown in Figure C.23.

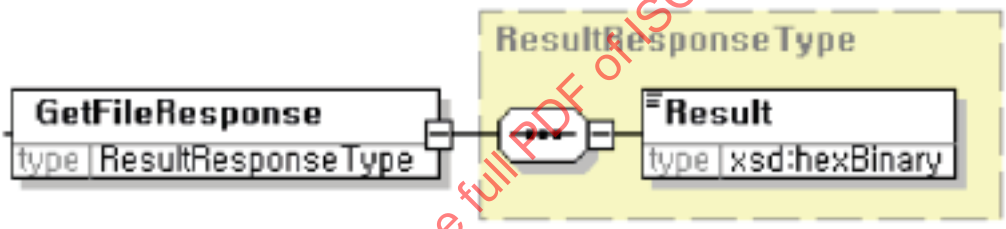


Figure C.23 — Get File Response Schema Structure

— The following XML example shows the get file response message for the file download request.

Table C.19 — XML Example of Get File Response Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <GetFileResponse>
  <Result>0000</Result>
 </GetFileResponse>
</ DMAP >
```

C.15 GET_FILE_RESULT

— XML schema structure of GET_FILE_RESULT Message is shown in Figure C.24.

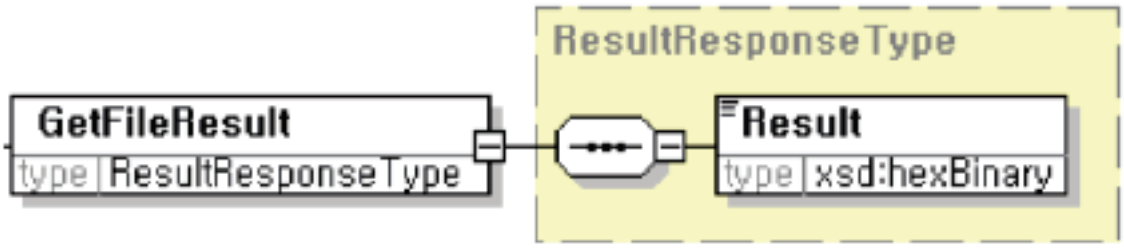


Figure C.24 — Get File Result Schema Structure

— The following XML example shows the get file result message, which announces transmission error, i.e. 'Can't open the target file'.

Table C.20 — XML Example of Get File Response Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
< DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <GetFileResult>
  <Result>0B0D </Result>
 </GetFileResult>
</ DMAP >
```


C.16 PUT_FILE_REQUEST

— XML schema structure of PUT_FILE_REQUEST Message is shown in Figure C.25.

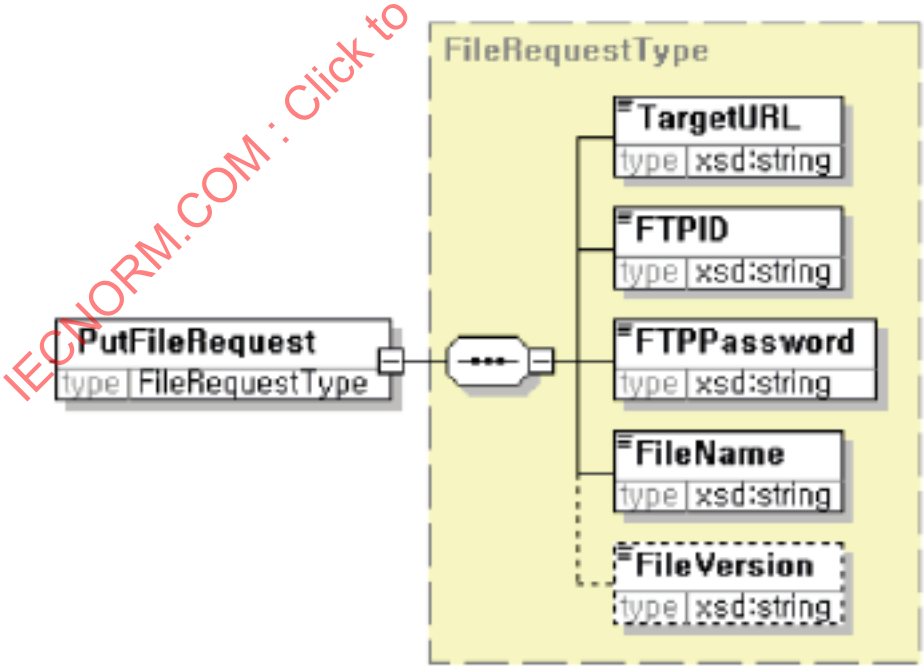


Figure C.25 — Put File Request Schema Structure

— The following XML example shows the put file request message including the FTP information.

Table C.21 — XML Example of Put File Request Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
      <PutFileRequest>
                  <TargetURL>test</TargetURL>
                  <FTPID>id</FTPID>
                  <FTPPassword>password</FTPPassword>
                  <FileName>test</FileName>
      </PutFileRequest>
</DMAP>
```

C.17 PUT_FILE_RESPONSE

— XML schema structure of PUT_FILE_RESPONSE Message is shown in Figure C.26.

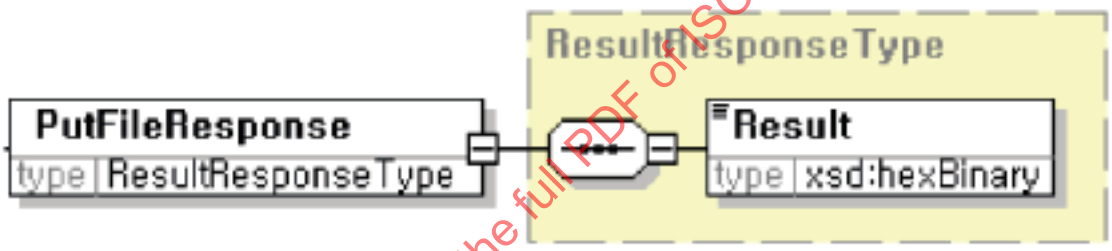


Figure C.26 — Get File Response Schema Structure

— The following XML example shows the put file response message for the file upload request.

Table C.22 — XML Example of Put File Response Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <PutFileResponse>
  <Result>0000</Result>
 </PutFileResponse>
</DMAP>
```

C.18 PUT_FILE_RESULT

— XML schema structure of PUT_FILE_RESULT Message is shown in Figure C.27.

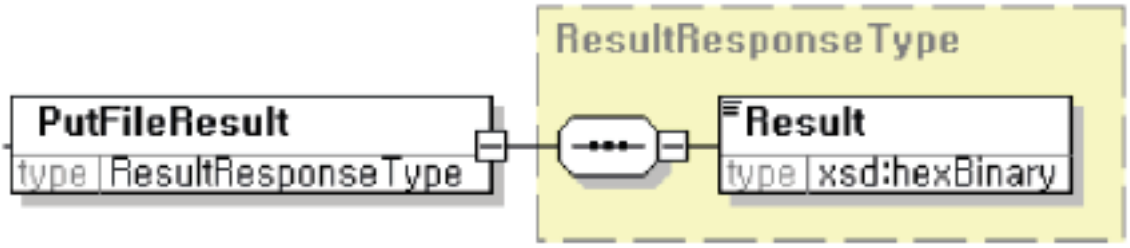


Figure C.27 — Put File Result Schema Structure

— The following XML example shows the put file result message, which announce transmission error, i.e. 'Can't open the target file'.

Table C.23 — XML Example of Put File Result Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <PutFileResult>
    <Result>0B0D </Result>
  </PutFileResult>
</DMAP>
```

C.19 APPLY_REQUST

— XML schema structure of APPLY_REQUEST Message is shown in Figure C.28.

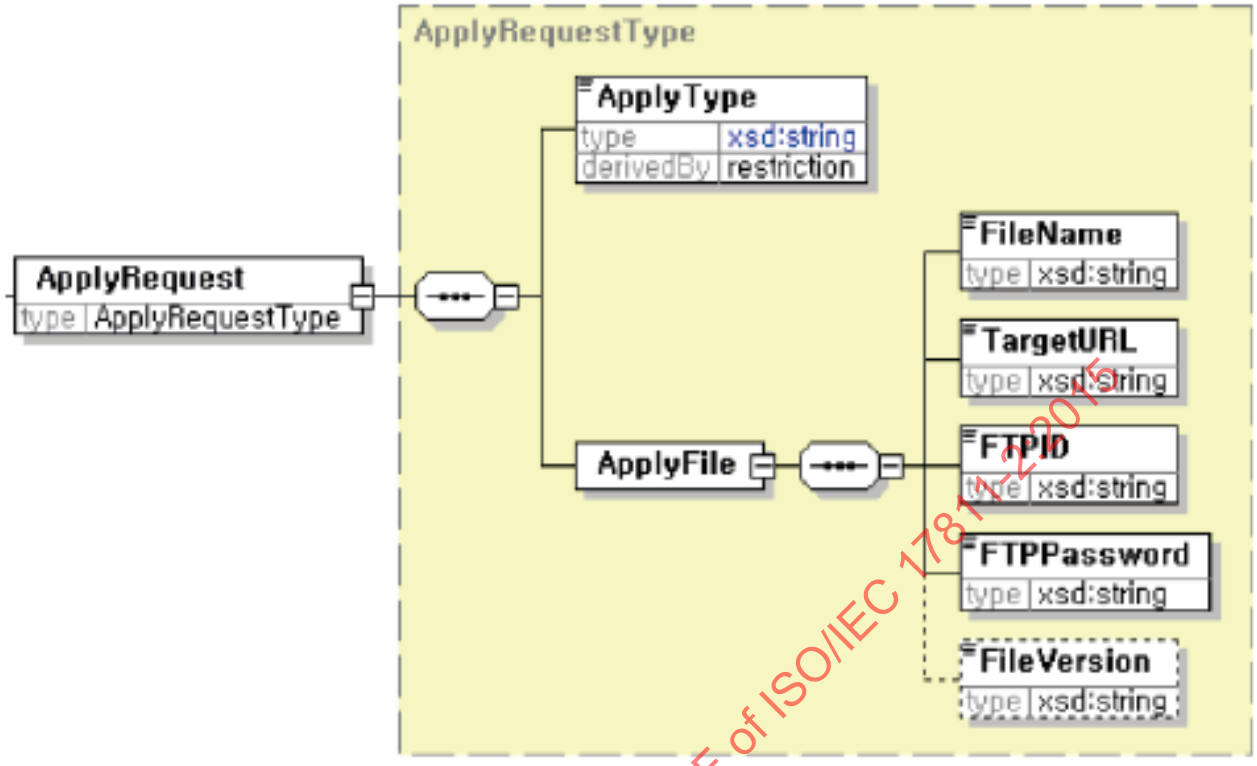


Figure C.28 — Apply Request Schema Structure

— The following XML example shows the apply request message including the FTP information.

Table C.24 — XML Example of Apply Request Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <ApplyRequest>
                <ApplyType>Execution</ApplyType>
                <ApplyFile>
                          <FileName>test.app</FileName>
                          <TargetURL>123.456.789</TargetURL>
                          <FTPID>test</FTPID>
                          <FTPPassword>test</FTPPasswd>
                </ApplyFile>
      </ApplyRequest>
</DMAP>
```

C.20 APPLY_RESPONSE

— XML schema structure of APPLY_RESPONSE Message is shown in Figure C.29.

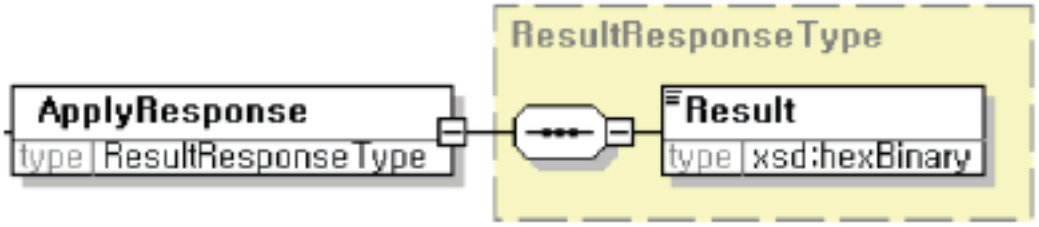


Figure C.29 — Apply Response Schema Structure

— The following XML example shows the apply response message with no error.

Table C.25 — XML Example of Apply Response Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <ApplyResponse>
  <Result>0000</Result>
 </ApplyResponse>
</DMAP>
```

C.21 APPLY_RESULT

— XML schema structure of APPLY_RESULT Message is shown in Figure C.30.

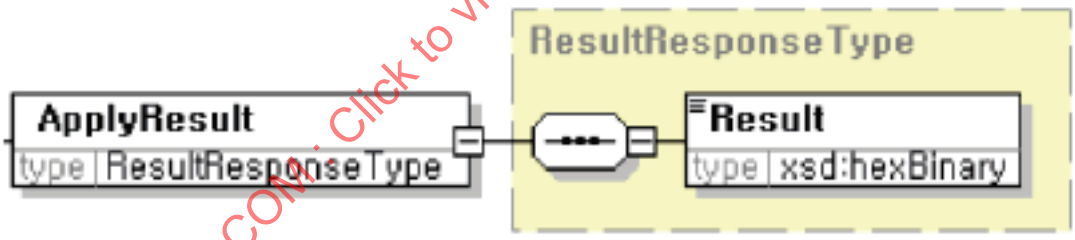


Figure C.30 — Apply Result Schema Structure

— The following XML example shows the apply result message which announce apply error, i.e. 'the other apply transaction is running'.

Table C.26 — XML Example of Apply Result Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <ApplyResult>
  <Result>0C03 </Result>
 </ApplyResult>
</DMAP>
```

C.22 DEVICE_REGISTRATION_REQUEST

— XML schema structure of DEVICE_REGISTRATION_REQUEST Message is shown in Figure C.31.

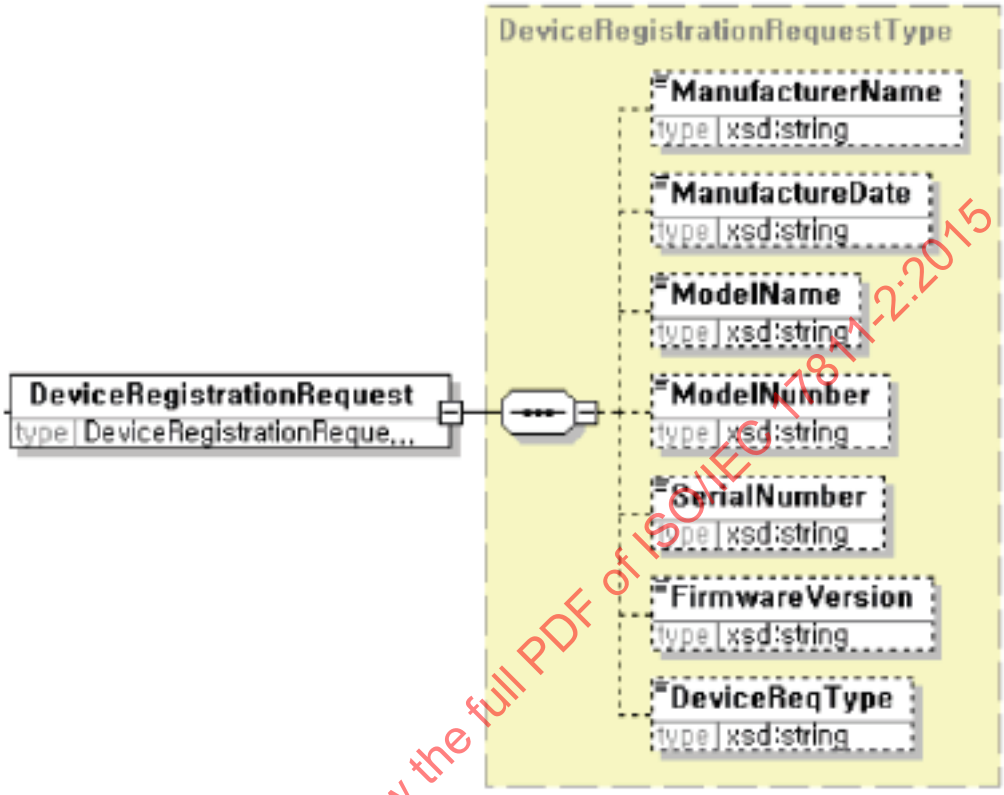


Figure C.31 — Device Registration Request Schema Structure

— The following XML example shows the device registration request message.

Table C.27 — XML Example of Device Registration Request Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <DeviceRegistrationRequest>
                  <ManufacturerName>test</ManufacturerName>
                  <ManufactureDate>2012-10-10</ManufactureDate>
                  <ModelName>test</ModelName>
                  <ModelNumber>1234fd</ModelNumber>
                  <SerialNumber>sadfasdfsaf</SerialNumber>
                  <FirmwareVersion>0.1</FirmwareVersion>
                  <DeviceReqType>new</DeviceReqType>
      </DeviceRegistrationRequest>
</DMAP>
```

C.23 DEVICE_REGISTRATION_RESPONSE

— XML schema structure of DEVICE_REGISTRATION_RESPONSE Message is shown in Figure C.32.

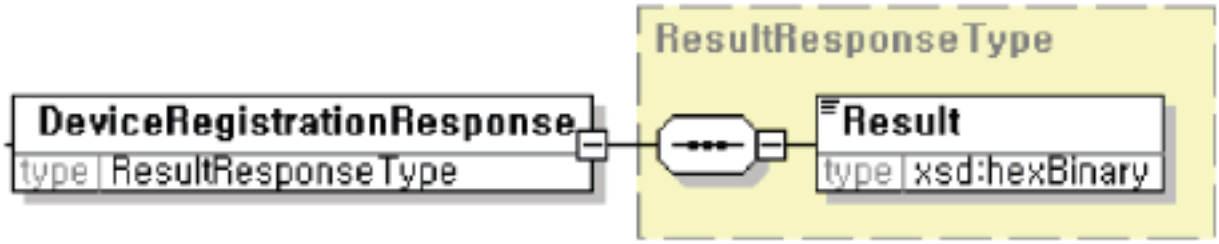


Figure C.32 — Device Registration Response Schema Structure

— The following XML example shows the device registration response message.

Table C.28 — XML Example of Device Registration Response Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <DeviceRegistrationResponse>
                  <Result>0000</Result>
      </DeviceRegistrationResponse>
</DMAP>
```

C.24 SERVICE_REGISTRATION_REQUEST

— XML schema structure of SERVICE_REGISTRATION_REQUEST Message is shown in Figure C.33.

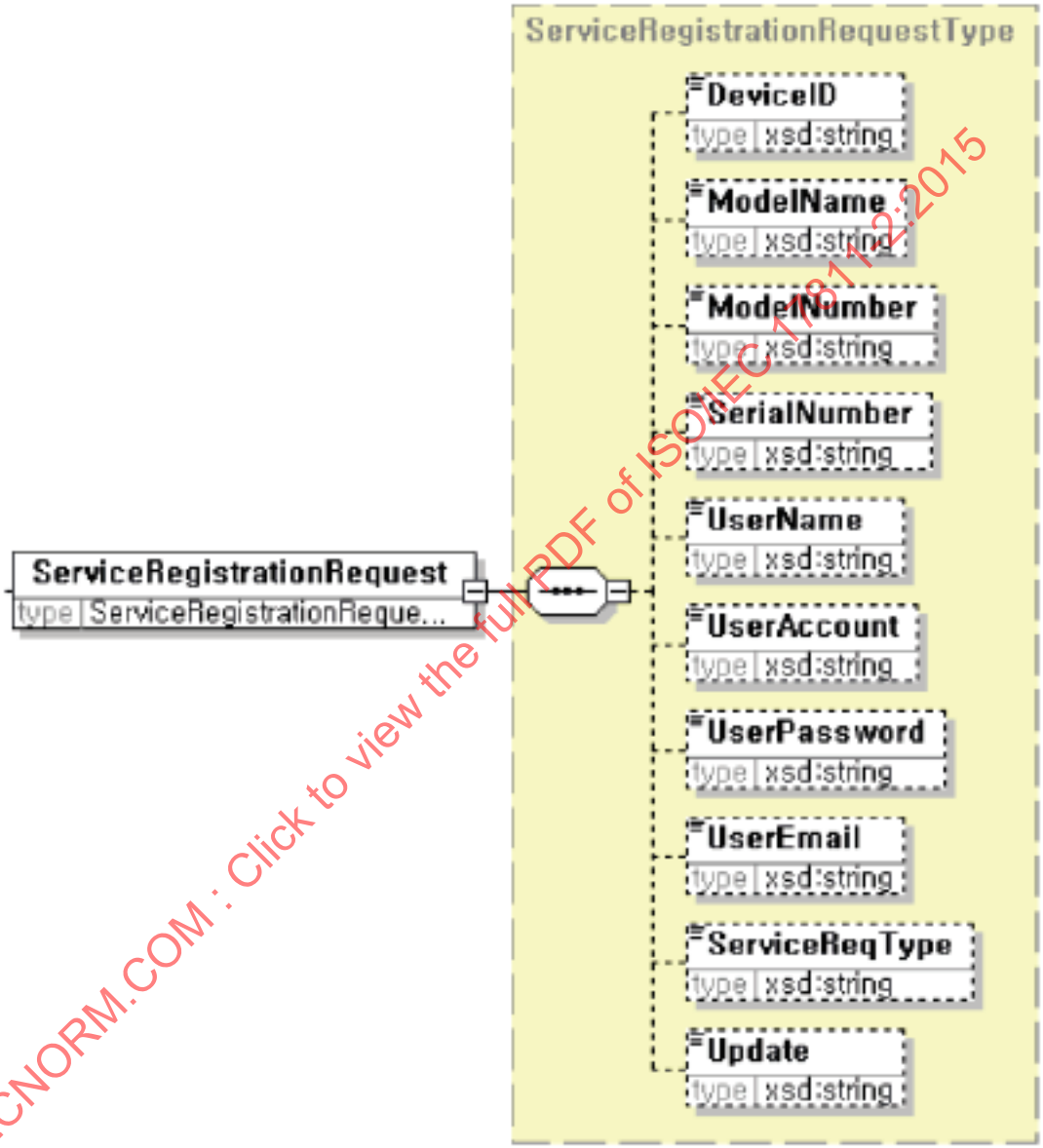


Figure C.33 — Service Registration Resquest Schema Structure

— The following XML example shows the service registration request message.

Table C.29 — XML Example of Service Registration Request Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <ServiceRegistrationRequest>
                  <DeviceID>0123456789AB0101</DeviceID>
                  <ModelName>test</ModelName>
                  <ModelNumber>test</ModelNumber>
                  <SerialNumber>test</SerialNumber>
                  <UserName>test</UserName>
                  <UserAccount>test</UserAccount>
                  <UserPassword>test</UserPassword>
                  <UserEmail>test</UserEmail>
                  <ServiceReqType>new</ServiceReqType>
                  <Update>test</Update>
      </ServiceRegistrationRequest>
</DMAP>
```

C.25 SERVICE_REGISTRATION_RESPONSE

— XML schema structure of SERVICE_REGISTRATION_RESPONSE Message is shown in Figure C.34.

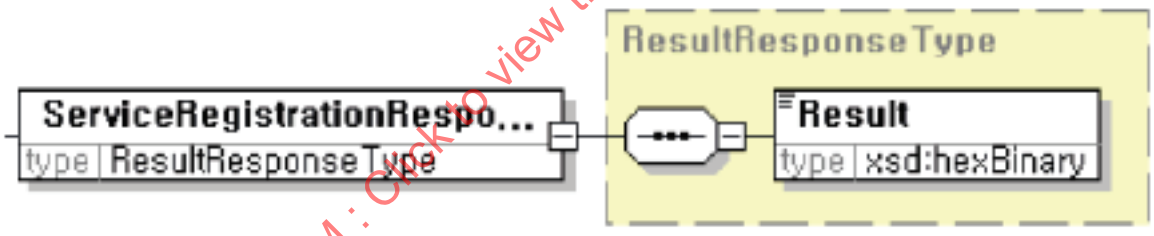


Figure C.34 — Service Registration Response Schema Structure

— The following XML example shows the service registration response message.

Table C.30 — XML Example of Service Registration Response Payload Message

```
<?xml version="1.0" encoding="UTF-8"?>
<DMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <ServiceRegistrationResponse>
                  <Result>0000</Result>
      </ServiceRegistrationResponse>
</DMAP>
```

C.26 XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xsd:element name="DMAP">
                <xsd:complexType>
                        <xsd:choice>
                                <xsd:element name="DeviceRegistrationRequest" type="DeviceRegistrationRequestType"/>
                                <xsd:element name="DeviceRegistrationResponse" type="ResultResponseType"/>
                                <xsd:element name="ServiceRegistrationRequest" type="ServiceRegistrationRequestType"/>
                                <xsd:element name="ServiceRegistrationResponse" type="ResultResponseType"/>
                                <xsd:element name="VersionInfoRequest" type="VersionInfoRequestType"/>
                                <xsd:element name="VersionInfoResponse" type="VersionInfoResponseType"/>
                                <xsd:element name="DiscoveryRequest" type="DiscoveryRequestType"/>
                                <xsd:element name="DiscoveryResponse" type="DiscoveryResponseType"/>
                                <xsd:element name="DeviceAdvertisement" type="AdvertisementType"/>
                                <xsd:element name="DeviceInfoRequest" type="DeviceInfoRequestType"/>
                                <xsd:element name="DeviceInfoResponse" type="DeviceInfoResponseType"/>
                                <xsd:element name="ControlRequest" type="ControlRequestType"/>
                                <xsd:element name="ControlResponse" type="ControlResponseType"/>
                                <xsd:element name="EventNotification" type="EventType"/>
                                <xsd:element name="EventSubscriptionRequest" type="EventSubscriptionRequestType"/>
                                <xsd:element name="EventSubscriptionResponse" type="ResultResponseType"/>
                                <xsd:element name="GetFileInfoRequest" type="GetFileInfoRequestType"/>
                                <xsd:element name="GetFileInfoResponse" type="GetFileInfoResponseType"/>
                                <xsd:element name="GetFileRequest" type="FileRequestType"/>
                                <xsd:element name="GetFileResponse" type="ResultResponseType"/>
                                <xsd:element name="GetFileResult" type="ResultResponseType"/>
                                <xsd:element name="PutFileRequest" type="FileRequestType"/>
```

```
                                <xsd:element name="PutFileResponse" type="ResultResponseType"/>
                                <xsd:element name="PutFileResult" type="ResultResponseType"/>
                                <xsd:element name="ApplyRequest" type="ApplyRequestType"/>
                                <xsd:element name="ApplyResponse" type="ResultResponseType"/>
                                <xsd:element name="ApplyResult" type="ResultResponseType"/>
                        </xsd:choice>
                </xsd:complexType>
        </xsd:element>
        <xsd:complexType name="DeviceRegistrationRequestType">
                <xsd:sequence>
                        <xsd:element name="ManufacturerName" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="ManufactureDate" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="ModelName" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="ModelNumber" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="SerialNumber" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="FirmwareVersion" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="DeviceReqType" type="xsd:string" minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="ServiceRegistrationRequestType">
                <xsd:sequence>
                        <xsd:element name="DeviceID" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="ModelName" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="ModelNumber" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="SerialNumber" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="UserName" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="UserAccount" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="UserPassword" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="UserEmail" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="ServiceReqType" type="xsd:string"
minOccurs="0"/>
                        <xsd:element name="Update" type="xsd:string" minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="VersionInfoRequestType">
                <xsd:sequence>
                        <xsd:element name="ManufacturerName" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="ModelName" type="xsd:string" minOccurs="0"/>
```

```
                        <xsd:element name="ModelNumber" type="xsd:string" minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="VersionInfoResponseType">
                <xsd:sequence>
                        <xsd:element name="Result" type="xsd:hexBinary"/>
                        <xsd:element name="Version" type="VerType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="DiscoveryRequestType">
                <xsd:sequence>
                        <xsd:element name="DiscoveryReqType">
                                <xsd:simpleType>
                                        <xsd:restriction base="xsd:string">
                                                <xsd:enumeration value="All"/>
                                                <xsd:enumeration value="DeviceID"/>
                                                <xsd:enumeration value="DeviceType"/>
                                                <xsd:enumeration value="DeviceName"/>
                                        </xsd:restriction>
                                </xsd:simpleType>
                        </xsd:element>
                        <xsd:choice>
                                <xsd:element name="StrTypeReq" type="xsd:string" minOccurs="0"/>
                                <xsd:element name="HexTypeReq" type="xsd:hexBinary" minOccurs="0"/>
                        </xsd:choice>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="DiscoveryResponseType">
                <xsd:sequence>
                        <xsd:element name="Result" type="xsd:hexBinary"/>
                        <xsd:element name="DeviceList">
                                <xsd:complexType>
                                        <xsd:sequence>
                                                <xsd:element name="BasicInfo"
type="BasicInfoType"/>
                                        </xsd:sequence>
                                        <xsd:attribute name="numofdevice" type="xsd:integer"
use="required"/>
                                </xsd:complexType>
                        </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="BasicInfoType">
                <xsd:sequence>
```

```
                        <xsd:element name="DeviceID" type="xsd:string"/>
                        <xsd:element name="DeviceType" type="xsd:hexBinary"/>
                        <xsd:element name="DeviceName" type="xsd:string"/>
                        <xsd:element name="DeviceSubname" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="DeviceCapabilityList">
                                <xsd:complexType>
                                        <xsd:sequence>
                                                <xsd:element name="DeviceCapability"
type="xsd:hexBinary" maxOccurs="unbounded"/>
                                        </xsd:sequence>
                                        <xsd:attribute name="numofcapability" type="xsd:integer"
use="required"/>
                                </xsd:complexType>
                        </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="AdvertisementType">
                <xsd:sequence>
                        <xsd:element name="DeviceID" type="xsd:string"/>
                        <xsd:element name="DeviceType" type="xsd:hexBinary"/>
                        <xsd:element name="DeviceName" type="xsd:string"/>
                        <xsd:element name="DeviceSubname" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="DeviceCapabilityList">
                                <xsd:complexType>
                                        <xsd:sequence>
                                                <xsd:element name="DeviceCapability"
type="xsd:hexBinary" maxOccurs="unbounded"/>
                                        </xsd:sequence>
                                        <xsd:attribute name="numofcapability" type="xsd:integer"
use="required"/>
                                </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="AdType">
                                <xsd:simpleType>
                                        <xsd:restriction base="xsd:string">
                                                <xsd:enumeration value="Start"/>
                                                <xsd:enumeration value="End"/>
                                        </xsd:restriction>
                                </xsd:simpleType>
                        </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="DeviceInfoRequestType">
```

```
                <xsd:sequence>
                        <xsd:element name=”DeviceInfoReqType”>
                                <xsd:simpleType>
                                        <xsd:restriction base=”xsd:string”>
                                                xsd:enumeration value=”FullDescription”/>
                                                <xsd:enumeration value=”BasicInfo”/>
                                                <xsd:enumeration value=”FunctionList”/>
                                                <xsd:enumeration value=”DeviceProperty”/>
                                                <xsd:enumeration value=”CommonProperty”/>
                                                <xsd:enumeration value=”ConfigProperty”/>
                                                <xsd:enumeration value=”StatusProperty”/>
                                                <xsd:enumeration value=”DeviceSpecificProperty”/>
                                        </xsd:restriction>
                                </xsd:simpleType>
                        </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name=”DeviceInfoResponseType”>
                <xsd:sequence>
                        <xsd:element name=”Result” type=”xsd:hexBinary”/>
                        <xsd:element name=”DeviceInfoList”>
                                <xsd:complexType>
                                        <xsd:choice>
                                                <xsd:element name=”FullDescription”
type=”FullDescriptionType” maxOccurs=”unbounded”/>
                                                <xsd:element name=”BasicInfo” type=”BasicInfoType”
maxOccurs=”unbounded”/>
                                                <xsd:element name=”FunctionList”
maxOccurs=”unbounded”>
                                                        <xsd:complexType>
                                                                <xsd:complexContent>
                                                                        <xsd:extension
base=”FunctionListType”>
                                                                                <xsd:attribute
name=”numoffunction” type=”xsd:integer” use=”required”/>
                                                                        </xsd:extension>
                                                                </xsd:complexContent>
                                                        </xsd:complexType>
                                                </xsd:element>
                                                <xsd:element name=”DeviceProperty”
type=”DevicePropertyType” maxOccurs=”unbounded”/>
                                                <xsd:element name=”CommonProperty”
type=”CommonPropertyType” maxOccurs=”unbounded”/>
```

```
                                                <xsd:element name="ConfigProperty"
type="ConfigPropertyType" maxOccurs="unbounded"/>
                                                <xsd:element name="StatusProperty"
type="StatusPropertyType" maxOccurs="unbounded"/>
                                                <xsd:element name="DeviceSpecificPropertyList" type
="DeviceSpecificPropertyListType" maxOccurs="unbounded"/>
                                        </xsd:choice>
                                        <xsd:attribute name="numofdevice" type="xsd:integer"
use="required"/>
                                </xsd:complexType>
                        </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="FullDescriptionType">
                <xsd:sequence>
                        <xsd:element name="BasicInfo" type="BasicInfoType"/>
                        <xsd:element name="FunctionList">
                                <xsd:complexType>
                                        <xsd:complexContent>
                                                <xsd:extension base="FunctionListType">
                                                        <xsd:attribute name="numoffunction"
type="xsd:integer" use="required"/>
                                                </xsd:extension>
                                        </xsd:complexContent>
                                </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="DeviceProperty">
                                <xsd:complexType>
                                        <xsd:sequence>
                                                <xsd:element name="CommonProperty"
type="CommonPropertyType"/>
                                                <xsd:element name="ConfigProperty"
type="ConfigPropertyType"/>
                                                <xsd:element name="StatusProperty"
type="StatusPropertyType"/>
                                                <xsd:element name="DeviceSpecificPropertyList" type
="DeviceSpecificPropertyListType"/>
                                        </xsd:sequence>
                                </xsd:complexType>
                        </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="DevicePropertyType">
                <xsd:sequence>
```

```
                        <xsd:element name="DeviceID" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="CommonProperty" type="CommonPropertyType"/>
                        <xsd:element name="ConfigProperty" type="ConfigPropertyType"/>
                        <xsd:element name="StatusProperty" type="StatusPropertyType"/>
                        <xsd:element name="DeviceSpecificPropertyList" type="DeviceSpecificPropertyLis
tType"/>
                </xsd:sequence>
        </xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="DeviceID" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="Model" type="ModelType" minOccurs="0"/>
                        <xsd:element name="Manufacture" type="ManufactureType" minOccurs="0"/>
                        <xsd:element name="Version" type="VerType" minOccurs="0"/>
                        <xsd:element name="Location" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="Weight" type="WeightType" minOccurs="0"/>
                        <xsd:element name="PhysicalSize" type="PhysicalSizeType" minOccurs="0"/>
                        <xsd:element name="DisplayList" minOccurs="0">
                                <xsd:complexType>
                                        <xsd:complexContent>
                                                <xsd:extension base="DisplayListType">
                                                        <xsd:attribute name="numofdisplay"
type="xsd:integer" use="required"/>
                                                </xsd:extension>
                                        </xsd:complexContent>
                                </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="MemoryList" minOccurs="0">
                                <xsd:complexType>
                                        <xsd:complexContent>
                                                <xsd:extension base="MemoryListType">
                                                        <xsd:attribute name="numofmemory"
type="xsd:integer" use="required"/>
                                                </xsd:extension>
                                        </xsd:complexContent>
                                </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="MPUList" minOccurs="0">
                                <xsd:complexType>
                                        <xsd:complexContent>
                                                <xsd:extension base="MPUListType">
                                                        <xsd:attribute name="numofMPU"
type="xsd:integer" use="required"/>
                                                </xsd:extension>
                                        </xsd:complexContent>
```

```
                                </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="NetworkList" minOccurs="0">
                                <xsd:complexType>
                                        <xsd:complexContent>
                                                <xsd:extension base="NetworkListType">
                                                        <xsd:attribute name="numofnetwork"
type="xsd:integer" use="required"/>
                                                </xsd:extension>
                                        </xsd:complexContent>
                                </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="StorageList" minOccurs="0">
                                <xsd:complexType>
                                        <xsd:complexContent>
                                                <xsd:extension base="StorageListType">
                                                        <xsd:attribute name="numofstorage"
type="xsd:integer" use="required"/>
                                                </xsd:extension>
                                        </xsd:complexContent>
                                </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="NeighborList" minOccurs="0">
                                <xsd:complexType>
                                        <xsd:complexContent>
                                                <xsd:extension base="NeighborListType">
                                                        <xsd:attribute name="numofneighbor"
type="xsd:integer" use="required"/>
                                                </xsd:extension>
                                        </xsd:complexContent>
                                </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="PowerConsumption" type="PowerConsumptionType" minOc-
curs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="ConfigPropertyType">
                xsd:sequence>
                        <xsd:element name="DeviceID" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="NetworkConfig">
                                <xsd:complexType>
                                        <xsd:sequence>
                                                <xsd:element name="Ethernet" minOccurs="0"
maxOccurs="unbounded">
                                                        <xsd:complexType>
                                                                <xsd:sequence>
```