# TELECOM APPLICATIONS PROFILE SPECIFICATION

## ZIGBEE PROFILE: 0X0107 VERSION 1.00

ZigBee Document 075307r07

April 1, 2010 3:03 am

Sponsored by: ZigBee Alliance

| | |
|---|---|
| Accepted by | This document has not yet been accepted for release by the ZigBee Alliance Board of Directors. |
| Abstract | This document defines the Telecom Applications profile. |
| Keywords | ZigBee, Profile, Telecom Services, Telecom Applications, Application Framework. |

**April 1, 2010**

**This page intentionally blank**

# NOTICE OF USE AND DISCLOSURE

ZigBee Alliance, Inc.
2400 Camino Ramon, Suite 375
San Ramon, CA 94583, USA

**This page intentionally blank**

# CONTACT INFORMATION

Much of the information in this document is preliminary and subject to change. Members of the ZigBee Working Group are encouraged to review and provide inputs for this proposal. For document status updates, please contact:

Claudio Borean
Telecom Italia,
Via G. Reiss Romoli, 274 Turin 10148 - ITALY
E-Mail: claudio.borean@telecomitalia.it
Phone: +39 011 228 5570

You can also submit comments using the ZigBee Alliance reflector. Its web site address is:

www.zigbee.org

The information on this page should be removed when this document is accepted.

**This page intentionally blank**

# PARTICIPANTS

The following is a list of those who were members of the ZigBee Architecture Review Committee (ZARC) leadership when this document was released:

**Skip Ashton**: Chair

**Phil Jamieson**: Vice-Chair

When the document was released, the Telecom Applications Profile Task Group leadership was composed of the following members:

**Claudio Borean**: Chair

**Chun Hui Zhu**: Vice-Chair

**Zhong Yongfeng**: Technical Editor

**Noriyuki Sato**: Secretary

Contributions were made to this document from the following members:

| | | | |
|---|---|---|---|
| Claudio Borean | Niwat Thepvilojanapong | Silviu Chiricescu | Yongjun Liu |
| Eunchang Choi | Shinji Motegi | Chunhui (Allan) Zhu | Yongfeng Zhong |
| Noriyuki Sato | Gabriel Chagaray | Roberta Giannantonio | Betty Zhao |
| Shigeru Fukunaga | Solène Quélard | Luis Miguel Campoy | Deepanshu Gautam |
| Kiyoshi Fukui | Boris Moltchanov | WoongChul Choi | Andrea Ranalli |
| Laura Colazzo | Klaus Kursawe | | |

**This page intentionally blank**

# TABLE OF CONTENTS

# LIST OF FIGURES

This page intentionally blank

# List of Tables

**This page intentionally blank**

# CHAPTER 1 INTRODUCTION

## 1.1 Scope

This profile defines device descriptions and standard practices for applications needed in a telecom market. Installation scenarios range from a single room to a large environment (e.g. entertainment areas). The key application domains included in this initial version are information delivery, location based services, peer-to-peer small data sharing, mobile commerce, mobile gaming, voice over ZigBee and chatting. Other applications will be added in future versions.

## 1.2 Purpose

This specification provides standard interfaces and device definitions to allow interoperability among ZigBee devices produced by various manufacturers of telecom applications market.

**This page intentionally blank**

# CHAPTER 2
# REFERENCES

## 2.1 ZigBee Alliance Documents

The following standards and specifications contain provisions, which through reference in this document constitute provisions of this specification. All the standards and specifications listed are normative references. At the time of publication, the editions indicated were valid. All standards and specifications are subject to revision, and parties to agreements based on this specification are encouraged to investigate the possibility of applying the most recent editions of the standards and specifications indicated below.

[R1] ZigBee Specification, ZigBee document number 053474r17, ZigBee Alliance

[R2] Telecom Applications - Marketing Requirements Document, ZigBee document number 064199r14, ZigBee Alliance.

[R3] Telecom Applications - Technical Requirements Document, ZigBee document number 064715r04, ZigBee Alliance

[R4] ZigBee Cluster Library Specification (ZCL), 075123r02, ZigBee Alliance

[R5] ZigBee document 064321r01, ZigBee Stack Profile

[R6] ZigBee document 064309r02, Commissioning Framework

[R7] ZigBee document 075357r00, Billing Cluster, ZigBee Telecom Application Profile Task Group

[R8] ZigBee document 08010r01, Partition Cluster, ZigBee Telecom Application Profile Task Group

[R9] ZigBee document 075358r01, Payment Cluster, ZigBee Telecom Application Profile Task Group

[R10] ZigBee document 075344r02, Mobile Gaming Cluster, ZigBee Telecom Application Profile Task Group

[R11] ZigBee document 075336r04, Voice Over ZigBee Cluster, ZigBee Telecom Application Profile Task Group

[R12] ZigBee document 075411r01, Chatting Cluster, ZigBee Telecom Application Profile Task Group

[R13] ZigBee document 075183r02, Information Delivery Cluster, ZigBee Telecom Application Profile Task Group

[R14] ZigBee document 075187r01, Data Rate Control Cluster, ZigBee Telecom Application Profile Task Group

[R15] ZigBee document 075212r02, P2P Data sharing cluster, ZigBee Telecom Application Profile Task Group

[R16] ZigBee document 075341r00 Extension of Location Cluster, ZigBee Telecom Application Profile Task Group

[R17] ZigBee document 08006r01, ZigBee-2007 Layer PICs and Stack Profiles, Core Stack Group

[R18] ZigBee document 053520r25, ZigBee Home Automation Profile Specification

[R19] ISO/IEC 7816, Identification Cards, Integrated Circuits cards with contacts

[R20] ZigBee document 094993r03, TA Best Practices, ZigBee Telecom Application

# CHAPTER 3

# DEFINITIONS

## 3.1 Conformance Levels

**Expected**: A key word used to describe the behavior of the hardware or software in the design models assumed by this Draft. Other hardware and software design models may also be implemented.

**May**: A key word indicating a course of action permissible within the limits of the standard (may equals is permitted).

**Shall**: A key word indicating mandatory requirements to be strictly followed in order to conform to the standard; deviations from shall are prohibited (shall equals is required to).

**Should**: A key word indicating that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; that a certain course of action is preferred but not necessarily required; or, that (in the negative form) a certain course of action is deprecated but not prohibited (should equals is recommended that).

## 3.2 ZigBee Definitions

**Attribute**: A data entity which represents a physical quantity or state. This data is communicated to other devices using commands.

**Cluster**: A container for one or more attributes and/or messages in a command structure.

**Cluster identifier**: A reference to the unique enumeration of clusters within a specific application profile. The cluster identifier is a 16-bit number unique within the scope of the application profile and identifies a specific cluster. Cluster identifiers are designated as inputs or outputs in the simple descriptor for use in creating a binding table.

**Device**: A description of a specific device within an application profile. For example, the light sensor device description is a member of the home automation application profile. The device description also has a unique identifier that is exchanged as part of the discovery process.

**Node**: Same as a unit.

**Product**: A product is a unit that is intended to be marketed. It implements application profiles that may be a combination of private, published, and standard.

**Service discovery**: The ability of a device to locate services of interest.

**Unit**: A unit consists of one or more physical objects (e.g., switch, controller, etc.) and their corresponding application profile(s) that share a single 802.15.4 radio. Each unit has a unique 64-bit IEEE address.

**ZigBee coordinator**: An IEEE 802.15.4-2003 PAN coordinator.

**ZigBee end device**: an IEEE 802.15.4-2003 RFD or FFD participating in a ZigBee network, which is neither the ZigBee coordinator nor a ZigBee router.

**ZigBee router**: an IEEE 802.15.4-2003 FFD participating in a ZigBee network, which is not the ZigBee coordinator but may act as an IEEE 802.15.4-2003 coordinator within its personal operating space, that is capable of routing messages between devices and supporting associations.

# CHAPTER 4

# ACRONYMS AND ABBREVIATIONS

## 4.1 Acronyms and Abbreviations

| | |
|---|---|
| AP | Access Point |
| APS | Application Support Sub-layer |
| CBA | Commercial Building Automation |
| EFT | Electronic Funds Transfer |
| EOF | End Of File |
| HA | Home Automation |
| ID | Information delivery |
| IN | Information node |
| m-commerce | Mobile commerce |
| MT | Mobile Terminal |
| PD | Payment Device |
| PHHC | Personal Home and hospital Health Care |
| P2P | Peer-to-Peer |
| POS | Point of Sales |
| RAN | RSSI Anchor Node |
| RLG | RSSI Location Gateway |
| RLN | RSSI Location node |
| SAS | Startup Attribute Set |
| SE | Smart Energy |
| TA | Telecom Applications |
| UD | User Device |

| | |
|---|---|
| VOZ | Voice Over ZigBee |
| ZAP | ZigBee Access Point |
| ZBH | ZigBee Headset |
| ZBM | ZigBee Microphone |
| ZBS | ZigBee Speaker |
| ZCL | ZigBee Cluster Library |
| ZIN | ZigBee Information Node |
| ZIT | ZigBee Information Terminal |
| ZSIM | ZigBee SIM card |

# CHAPTER 5

# PROFILE DESCRIPTION

## 5.1 A ZigBee Telecom Applications Network

TA networks scale from 2 to hundreds of nodes. TA networks involve network equipment and also mobile terminals used by people that may not have any ZigBee expertise. Installation and configuration concepts shall be easy and uniform across multiple OEM vendors.

ZigBee Telecom Applications, as described in [R2] are very heterogeneous, so the requirements deriving from this kind of networks are different from other ZigBee Profiles as described in [R3]. Since the user is typically directly involved in the network using a mobile handset, he expects to see the result of his interaction across the network quickly.

TA networks could include both ZigBee feature set and ZigBee PRO feature set nodes, it is recommended that the ratio of the nodes should not be 50/50, but the majority of the nodes in the network should be based on one stack profile or the other to get consistent performance.

ZigBee TA products may[1] support the ZigBee commissioning cluster. TA may need a way to disable it for some scenarios. A sample scenario for accessing mobile payment services could be the following one:

- Go to service provider kiosk
- Select “registration” - switch to commissioning network
- Join registration network
- Download service SAS
- Go to service location (e.g. ticketing machine)
- Select service at location which loads SASs

1. CCB #1112

- Use service.

A ZigBee TA makes possible networks such as the following:

- Information network
- Small Data sharing network
- Mobile payment/ticketing network
- Chatting networks
- Voice over ZigBee network

Combination of these networks shall be allowed, however for services that require higher data rates than usual ZigBee services like data sharing and VoZ, the usage of the mechanism specified for the Data Rate Control Cluster is recommended. Information network may be intended as a commissioning network in certain services like m-commerce.

## 5.2 ZigBee Stack Profile

Products that conform to this specification shall use stack profile number 0x01 or profile 0x02, as defined in [R1] and in [R17].

## 5.3 Startup Attribute Set (SAS)

In order to insure interoperability, all ZigBee TA devices should implement compatible Startup Attribute Sets (SAS). The device must internally implement these stack settings to insure compatibility and consistent user experience. TA shall use the SAS as specified in the following clauses.

### 5.3.1 Startup Parameters

The startup parameters and their default values are listed in Table 5.1.

**Table 5.1 Startup Parameters**

| Parameter | Value | Comment |
|---|---|---|
| Short Address | 0xFFFF or installer specified. | |
| E PANiD | 0x0000000000000000 or 0x0050C27710000000 | The second EPID is the global commissioning EPID reserved by the ZigBee Alliance. |
| PAN ID | 0xFFFF or installer specified. | |
| Channel Mask | All channels in frequency band. | If needed, the power transmitted by the device on channel 26 can be lowered to comply with FCC regulations. |
| Protocol Version | 0x02 (ZigBee and later) | |
| Stack Profile | 1 (ZigBee) or 2 (ZigBee PRO) | |
| Startup Control | 2 (two) if un-commissioned, so it will join network by association when join command is indicated.<br><br>0 (Zero) if commissioned. Indicates that the device should consider itself a part of the network indicated by the *ExtendedPANId* attribute. In this case it will not perform any explicit join or rejoin operation. | |
| Trust Center Address | 0x0000000000000000 or installer specified. | Please note: Identifying or establishing the Trust Center as a device other than the Coordinator is an optional stack profile feature. Since this is an implementation specific issue, installation tools and setting address is managed by the OEM vendors. |
| Master Key | | not used, high security is not used in this profile. |

**Table 5.1 Startup Parameters (Continued)**

| Parameter | Value | Comment |
|---|---|---|
| Link Key | 0x00000000000000000000000000000001if the Key Establishment Cluster is being used to install a link key<br><br>Installer provided if using preconfigured link keys | |
| Network Key | 0x00000000000000000000000000000001 if no pre-installed key present | |
| Use Insecure Join | TRUE | This setting enables the device to join a network that uses the ZigBee stack profile, e.g. a Home Automation network. It should be typically disabled (by the commissioning tool) in an operational TA network. |

## 5.3.2 Join Parameters

The join parameters and their default values are listed in Table 5.2.

**Table 5.2 Join Parameters**

| Parameter | Value | Comment |
|---|---|---|
| ScanAttempts | | At boot time or when instructed to join a network, the device should complete up to three (3) scan attempts to find a ZigBee Coordinator or Router with which to associate. If it has not been commissioned, this means that when the user presses a button or uses another methodology to join a network, it will scan all of the channels up to three times to find a network that allows joining. If it has already been commissioned, it should scan up to three times to find its original PAN to join. (ZigBee Pro devices should scan for their original extended PAN ID and ZigBee (2007) devices can only scan for their original PAN ID). |
| TimeBetweenScans | 1 second | determines the number of seconds between each unsuccessful scan attempt. |
| RejoinInterval | 60 seconds or shorter | how quickly a device will attempt to rejoin the network if it finds itself disconnected. |
| MaxRejoinInterval | 15 minutes | imposes an upper bound on the RejoinInterval parameter - this must be restarted if device is touched by human user, i.e. by a button press. This parameter is intended to throttle how often a device will scan to find its network in case the network is no longer present and therefore a scan attempt by the device would always fail, i.e., if a device finds itself disconnected, it will try to rejoin the network, scanning all channels if necessary. If the scan fails to find the network, or fails to successfully rejoin, the device will wait for 15 minutes before attempting to rejoin again. |

## 5.3.3 Security Parameters[2]

SecurityTimeoutPeriod: Determined by the stack profile.

TrustCenterNetworkKey: The Trust Center will pick the network key. ZigBee TA devices shall not depend on pre-configured keys to be commissioned or to interoperate.

Trust Center Link Key: 0x5A 0x69 0x67 0x42 0x65 0x65 0x41 0x6C 0x6C 0x69 0x61 0x6E 0x63 0x65 0x30 0x39

2. CCB #1127

The current network key shall be transported using the default TC link key in the case where the joining device is unknown or has no specific authorization associated with it. This allows for the case where alternative pre-configured link keys specifically associated with a device can be used as well. It is not required to use Link keys for communication when a device has joined the network unless explicitly specified by the individual device which clusters require link keys. Only network level security is required when not specified.

The security parameters and their default values are listed in Table 5.3.

**Table 5.3 Security Parameters**

| Parameter | Value | Comment |
|---|---|---|
| SecurityTimeoutPeriod | set by stack profile | |
| DefaultTrustCenterlinkKey | 0x5A 0x69 0x67 0x42 0x65 0x65 0x41 0x6C 0x6C 0x69 0x61 0x6E 0x63 0x65 0x30 0x39 | For ZigBee Telecom Applications devices the network key shall be transported using the default TC link key in the case where the joining device is unknown or has no specific authorization associated with it. |

## 5.3.4 End Device Parameters

The end device parameters and their default values are listed in Table 5.4.

**Table 5.4 End Device Parameters**

| Parameter | Value | Comment |
|---|---|---|
| IndirectPollRate | set by stack profile | This is how often a device will poll its parent for new data. It is recommended that an end device that is designed to receive data should poll its parent every 60 seconds. |

### 5.3.5 Link Status Parameters

The link status parameters and their default values are listed in Table 5.5

**Table 5.5 Link Status Parameters**

| Parameter | Value |
|---|---|
| LinkStatusPeriod | set by stack profile |
| RouterAgeLimit | set by stack profile |
| RepairThreshold | set by stack profile |

### 5.3.6 Concentrator Parameters

The concentrator parameters and their default values are listed in Table 5.6.

**Table 5.6 Concentrator Parameters**

| Parameter | Value | Comment |
|---|---|---|
| ConcentratorFlag | set by stack profile | configures device to be a concentrator. |
| RouterAgeLimit | 12 (twelve) | OEMs that produce a concentrator product will set the max concentrator radius to this value. |
| RepairThreshold | set by stack profile | This is how soon nodes should reply to a concentrator after hearing a route request command. |

## 5.3.7 APS Transport Parameters

The APS transport parameters and their default values are listed in Table 5.7.

**Table 5.7 APS Transport Parameters**

| Parameter | Value | Comment |
|---|---|---|
| MaxFrameRetries | set by stack profile | This determines the maximum number of retries allowed after a transmission failure. |
| AckWaitDuration | set by stack profile | This is the maximum number of seconds to wait for acknowledgement of an APS frame. |

## 5.3.8 APS Fragmentation Parameters

For fragmentation there are application settings from the APS IB that must be defined by the application profile. For Telecom Applications these parameters are to be set as follows in Table 5.8:

**Table 5.8 APS Fragmentation Parameters**

| Parameters | Identifier | Type | Value | Description |
|---|---|---|---|---|
| apsInterframeDelay | 0xc9 | Integer | 50 | Standard delay in milliseconds between sending two blocks of a fragmented transmission (see sub-clause 2.2.8.4.5) |
| apsMaxWindowSize | 0xcd | Integer | 3 | Fragmentation parameter - The maximum number of unacknowledged frames that can be active at once (see sub-clause 2.2.8.4.5). |

In addition the Maximum Incoming Transfer Size Field in the Node descriptor defines the largest ASDU that can be transferred using stack fragmentation. For Telecom Applications this value shall be set to 512 bytes.

### 5.3.9 Binding Parameters

The binding parameters and their default values are listed in Table 5.9.

**Table 5.9 Binding Parameters**

| Parameter | Value | Comment |
|---|---|---|
| EndDeviceBindTimeout | 60 seconds | Timeout value for end device binding. End Device binding is set by the coordinator. |

## 5.4 Device Descriptions

Device descriptions specified in this profile are summarized in Table 5.10 along with their respective Device IDs. The devices are organized according to the end application areas they address. A product that conforms to this specification shall implement at least one of these device descriptions and shall also include the device descriptions corresponding to all applications implemented on the product where a standard device description is specified in this profile. For example, if a product implements both an information node and a point of sale application, then the Information node and Point of Sale device descriptions must both be supported.

This list will be added to in future versions of the profile as new clusters are developed to meet the needs of manufacturers. The reserved values shall not be used until the profile defines them. Manufacturer-specific device descriptions shall reside on a separate endpoint and use a private profile ID.

**Table 5.10 Devices Specified in the TA Profile**

| | Device | Device ID |
|---|---|---|
| **Generic** | ZigBee SIM card (ZSIM) | 0x0000 |
| | ZigBee Mobile Terminal (ZMT) | 0x0001 |
| | Configuration Tool | 0x0005 |
| | Range Extender | 0x0008 |
| | Reserved | 0x0003, 0x0004, 0x0006, 0x0007 0x0009-0x00FF |
| **Information** | ZigBee Access Point (ZAP) | 0x0100 |
| | ZigBee Information node (ZIN) | 0x0101 |
| | ZigBee Information Terminal (ZIT)[a] | 0x0102 |
| | Reserved | 0x0103- 0x01FF |
| | | |
| **Payments** | Point of sale | 0x0200 |
| | Ticketing machine | 0x0201 |
| | Pay- controller | 0x0202 |
| | Billing unit | 0x0203 |
| | Charging unit | 0x0204 |
| | Reserved | 0x0205- 0x02FF |
| **Cards** | ZigBee flash card | 0x0300 |
| | ZigBee PC Smart Card Reader | 0x0301 |
| | Reserved | 0x0302 - 0x3FF |
| **Voice** | ZigBee headset | 0x0400 |
| | ZigBee microphone | 0x0401 |
| | ZigBee Speaker | 0x0402 |
| | Reserved | 0x0403 -0x04FF |
| **Localization[b]** | RSSI Anchor Node (RAN) | 0x0500 |
| | RSSI Location Node (RLN) | 0x0501 |
| | RSSI Location Gateway (RLG) | 0x0502 |
| | Reserved | 0x0503 -0x05FF |

**Table 5.10 Devices Specified in the TA Profile (Continued)**

| | Device | Device ID |
|---|---|---|
| **Chatting**[c] | Chatting Unit | 0x0600 |
| | Chatting station | 0x0601 |
| | Reserved | 0x0602 -0x06FF |
| **Reserved** | Reserved | 0x0700-0xbFFF |

a. CCB #1116

b. CCB #1116

c. CCB #1116

## 5.5 ZigBee Cluster Library (ZCL)

This profile utilizes clusters specified in the ZigBee Cluster Library. The implementation details for each cluster are given in the ZCL specifications. Further specification and clarification is given in this profile where necessary.

The ZCL provides a mechanism for clusters to report changes to the value of various attributes. It also provides commands to configure the reporting parameters. The attributes that a particular cluster is capable of reporting are listed in the ZCL specification for each cluster. A product shall support the reporting mechanism for all attributes specified in the ZCL that this product implements with a given cluster. The minimum reporting interval specified in the ZCL [R4] shall be set to a value greater than or equal to 0x0001. The maximum reporting interval should be set to 0x0000 by default, and if it is set to a non-zero value it shall be set to a value greater than or equal to 0x003C and greater than the value of the minimum reporting interval. These settings will restrict the attributes from being reported more often than once every second if the attribute is changing quickly and at least once every minute if the attribute does not change for a long time. It is recommended that the minimum reporting interval be set to a higher value whenever the application can tolerate it. It is recommended that the maximum reporting interval be set to a much greater value to avoid unnecessary traffic.

## 5.6 Cluster List

The clusters used in this profile, are listed in Table 5.11. The clusters are listed according to the functional domain they belong to in the ZCL. The corresponding cluster identifiers can be found in the ZCL Foundation specification [R4].

The functionality made available by all supported clusters shall be that given in their ZCL specifications except where a device description in this profile includes further specification, clarification or restriction as needed for a particular device.

Most clusters include optional attributes. The application designer must be aware that optional attributes may not be implemented on a particular device. It is the responsibility of a device's application to discover and deal with unsupported attributes on other devices.

It is expected that clusters will continue to be developed in the ZCL that will be useful in this profile. In many cases, new clusters will be organized into new device descriptions that are separate from those currently defined. There may also be situations where it makes sense to add clusters as optional or possibly even mandatory elements of existing device descriptions. Creating new device descriptions is the preferred method of adding new clusters to this specification, because new functionality can be mandated in a new device description without causing compatibility issues with previously defined devices.

Manufacturer-specific clusters may be added to any device description in this profile as long as they follow the specifications given in the ZCL Foundation specification [R4].

**Table 5.11 Clusters Used in the TA Profile**[a]

| Functional Domain | Cluster Name | Cluster IDs |
|---|---|---|
| General | Basic | 0x0000 |
| General | Identify | 0x0003 |
| General | Groups | 0x0004 |
| General | On/Off | 0x0006 |
| General | Commissioning | 0x0015 |
| General | Partition *New* | 0x0016 |
| General | RSSI Location *Extended* | 0x000b |
| General | Alpha-Secure Key Establishment (ASKE) | 0x0017 |
| General | Alpha-Secure Access Control (ASAC) | 0x0018 |
| Protocol Interfaces | ISO7816 Protocol Tunnel *New* | 0x0615 |
| Telecommunication | Information *New* | 0x0900 |
| Telecommunication | Data sharing *New* | 0x0901 |
| Telecommunication | Gaming *New* | 0x0902 |
| Telecommunication | Data Rate control *New* | 0x0903 |

**Table 5.11 Clusters Used in the TA Profile[a] (Continued)**

| Functional Domain | Cluster Name | | Cluster IDs |
|---|---|---|---|
| Telecommunication | Voice over ZigBee | *New* | 0x0904 |
| Telecommunication | Chatting | *New* | 0x0905 |
| Financial | Payment | *New* | 0x0a01 |
| Financial | Billing | *New* | 0x0a02 |

a. CCB #1116

### 5.6.1 Cluster Dependencies

The clusters listed in Table 5.11used in TA profile, even if defined as optional for specific devices, shall follow the dependencies as specified below (when implemented):

- Whenever the Billing cluster is supported, the alpha-secure clusters shall be supported as well;
- Whenever the Payment cluster is supported, the alpha-secure clusters shall be supported as well;
- Specific clusters defined in TA profile like Data sharing may also require the use of Partition cluster.

## 5.7 Commissioning

### 5.7.1 Startup and Joining

Many, if not all of the devices described in this document will require some form of commissioning, even if the user or installer doesn't see it. This is because, for example, an actuating device needs to be bound to some sort of target in order to do useful work. It is worth specifying that the discovery of TA services may be performed using match descriptor requests (and managing the respective match descriptor responses) with profileID equal to TA profile (0x0107).

The expected commissioning practices according to the specific use cases enabled by the TA profile are listed in 5.7.2 and in the following table:

**Table 5.12 Commissioning Practices for TA Use Cases**

| ID# | Use Case | Description |
|---|---|---|
| **1** | *Information delivery free information and Mobile Advertising* | The device should be able to join any info delivery network using common security model (TC link key). In order to maintain compatibility with ZigBee profile interoperability default TC link key need to be used to transmit the network key. The default link keys may be updated using alpha secure clusters in order to operate remote configuration of information nodes. |
| **2** | *Information delivery access limited information* | TC link key may set out-of-band to the mobile terminals after service subscription; in case of access to secured information the proper procedure should be followed according to the Access Control field as configured in the information nodes using Update command of Information cluster. |
| **3** | *Location based services* | The device should be able to join any info delivery network using common security model (TC link key). |
| **4** | *Mobile Office* | Pre-configured link key may be used (out of band e.g. PIN codes): PCSC uses its own security level; alternatively alpha secure clusters may be used for key updates. |
| **5** | *Voice Over ZigBee* | The device should be able to join any VoZ network using common security model (TC link key). |
| **6** | *Data Sharing* | Besides common security model (TC link key) to transmit NWK key Link Keys may be established by devices in case of secure data sharing. |
| **7** | *Mobile Payment* | Defined in next version of the profile. |
| **8** | *Chatting* | The device should be able to join any chatting network using common security model (TC link key). In order to maintain compatibility with ZigBee profile interoperability default TC link key need to be used to transmit the network key. The default link keys may be updated using alpha secure clusters or out-of-band methods for peer-to-peer secret messages transmission. |
| **9** | *Mobile Gaming* | Same as scenario #8 |

## 5.7.2 Best Practices and General Procedures

### 5.7.2.1 Mobile Advertising

#### 5.7.2.1.1 Recommended Setup Procedures for Mobile Advertising

The following steps are recommended for setting Mobile advertising scenario:

**Step 1: Find the service**

- Try to join any network;
- Use ZDO Match descriptor request looking for devices supporting Information cluster;
- If matches are found it means a Mobile adverting or information delivery network is in the area. If not, leave the network and try to join to another network.

**Step 2: Set the communication with ZigBee Information nodes (ZIN) or ZigBee Access Point (ZAP)**

- For the forwarding scenario ZIN must know the ZAP address: if not know ZIN should send ZDO Simple Descriptor request and get the device ID of ZAP or use ZDO Match descriptor request to the GW cluster (in case the ZAP supports gateway functionalities);
- For the redirection scenario MT/ZSIM need to know the ZAP address: MT/ZSIM might send ZDO Simple Descriptor request and get the device ID of ZAP or use ZDO Match descriptor request to the GW cluster (in case the ZAP supports gateway functionalities);
- A MT/ZSIM can enable or disable the capability to notify the ZIN of its presence in the area (i.e. by sending the Push Information Response). In order to notify the ZAP with the reception of an advertisement (i.e. a Push Information Response sent back to the ZIN), the ZIN could send a unicast Push Information command to the ZAP, notifying in the payload the presence of the MT in its advertising area.

**Step 3: Detect the exit of an advertising area**

- The application running on the mobile device should detect that the MT/ZSIM is still in an area covered by any information nodes by periodically sending ZDO Match descriptor requests.

### 5.7.2.2 Location Based Services

#### 5.7.2.2.1 Recommended Setup Procedures for Location Based Services for Centralized Location

**Step 1: Find the service and join to the network**

- Try to join any network;
- Use ZDO Match descriptor request looking for devices supporting RSSI Location cluster;
- If matches are found it means an RSSI based location service is in the area.

**Step 2: Set up the network**[3]

- Once an Anchor Node has joined the network, it needs to know the address of the RSSI Location gateway it needs to talk to, in order to enable centralized location: in order to find it an anchor node should send ZDO Match descriptor request to the GW cluster. After getting the proper gateway address the anchor node might send an Anchor Node Announce in broadcast to notify its presence to the network.
- If the user needs to configure the information related to the position of the RSSI Anchor Node, the RSSI Location Gateway should write the *LocationDescription* attribute of Basic cluster of the RSSI Anchor Node. In this case, when changed, the RSSI Anchor node should send the AnchorNodeAnnounce command out again with the updated coordinates.

**Step 3: Detect the exit of the location based service**

- The application running on the RSSI Location Node[4] should detect if the device is still or is not in an area covered by any anchor node: if after 3 Request Own Location commands the node doesn't receive any response it should leave and try again to join the network. Also, it should periodically send ZDO Match descriptor requests in order to be sure to be always reachable by a gateway (case of centralized location).

### 5.7.2.3 Mobile Office

#### 5.7.2.3.1 Recommended Setup Procedures for Mobile Office

The following steps are recommended for setting Mobile Office scenario.

This scenario provides a star ZigBee network topology. At least two ZigBee nodes are involved: a ZigBee Smartcard (a ZSIM or a ZigBee enabled mobile phone), which performs service discovery and joins ZigBee network, and a ZigBee PCSC Reader, that is the network coordinator.

In order to find the service a ZigBee Smartcard should perform following actions:

- Try to join any network
- Use ZDO Match descriptor request looking for devices supporting ISO7816 Protocol Tunnel cluster;

3. CCB #1116
4. CCB #1116

- If matches are found it means that at least one device supporting ISO7816 Protocol Tunnel cluster is in the area;
- Use ZDO Simple Descriptor Request to identify ZigBee PCSC Readers among devices supporting ISO7816 Protocol Tunnel cluster

If Simple Descriptor is verified for more than one device, ask the User which device he wants to connect to. To simplify user experience, PCSC User Descriptors or similar user-friendly ID should be collected by ZigBee SmartCard and proposed to User during device choice.

It's recommended to store network and PCSC Reader device parameters into ZigBee SmartCard non-volatile memory: these parameters should be used to try joining directly to the last-found network during next connections before performing a full scan. If no matches are found or no device with requested Simple Descriptor is in this network, leave the network and try to join to another network.

#### 5.7.2.3.2 Mobile Office Setup Description

Once joined, a User should virtually insert ZigBee SmartCard in a PCSC Reader connected to a Computer Infrastructure. This step is made enabling the ZigBee SmartCard to send InsertSmartCard command to the Reader. The PCSC Reader node changes its *Status* attribute in “busy”, notifies the Computer Infrastructure about the SmartCard arrival and then it starts exchanging several APDUs commands/responses with ZigBee SmartCard by means of TransferAPDU command, carrying ISO7816 protocol APDUs.

Other ZigBee SmartCards that send InsertSmartCard command while the PCSC Reader is busy should be ignored. A similar result should be reached by a SmartCard read and check of PCSC Reader *Status* attribute before sending InsertSmartCard command: if the PCSC Reader is busy no command will be sent.

The virtual removal of ZigBee Smartcard from PCSC Reader is done by sending ExtractSmartCard command to the Reader. Alternatively the PCSC Reader should check the ZigBee SmartCard presence periodically sending ZDO Match descriptor request when no TransferAPDU command is sent. In this case no response means SmartCard absence[5].

### 5.7.2.4 Chatting

#### 5.7.2.4.1 Recommended Service Discovery Procedure for Chatting

Following steps are recommended for discovery chatting:

- Try to join any network;

5. CCB #1128

- Use ZDO Match descriptor request looking for devices supporting chatting cluster.
- If matches are found it means the node has found a valid chatting network. If not, leave the network and try to join to another network (delay between consecutive tries is application specific).

#### 5.7.2.4.2 Chat Setup Description

Once joined to the network, the user should search for existing chat rooms: the chairmen shall respond with Search Chat Response command to notify the user with a list of chat rooms available. The user can then request to join an existing chat room but if the chairmen maintain no chat room or the user doesn't want to join any of them it may form a new chat room and maintain it itself, or request the server to form a new chat room with the Start Chat Request command. After the chat room is formed, the server may broadcast the Search Chat Response command to notify the created room to the other users in the network.

Multicast is the recommended communication method in some procedures of the chatting and chat management as specified above. Groups cluster should be used for group management. Broadcast may be used also, such as in the case multicast is not supported.

### 5.7.2.5 Voice Over ZigBee

#### 5.7.2.5.1 Recommended Setup Procedures for Voice Over ZigBee

The following steps are recommended for setting Voice over ZigBee scenario:

**Step 1: Find the service**

- Try to join any network.
- Use ZDO Match descriptor request looking for devices supporting VoZ cluster.
- If matches are found it means that at least one VoZ device is in the area. If not, leave the network and try to join to another network.

**Step 2: Set the communication between VoZ nodes (VoZNs)**

- VoZN must know another VoZN address. If not know another VoZN address, it should use ZDO Simple Descriptor Request to identify VoZN among devices and get the device ID of another VoZN. If ZDO Simple Descriptor is identified for more than one, ask the user to select a device to communicate with.
- If VoZN knows the device ID of another VoZN, it could send the Establishment Request command with its codec information supporting a codec of another VoZN. VoZN which received the command respond the Establishment Response command with ACK.

### 5.7.3 Security Settings

**(Re)Comissioning Phase**

For the use of the ASKE cluster, the mobile network securely loads into the device a 32 bit identifier, consisting of a 24 bit unique device identifier and an 8 bit version number of the keying material. The version number can be equal to the KeyingMaterialVersion parameter in the ASKE Security Domain Table. The mobile network SHALL keep a log file of identifiers submitted to devices for future linkability. The mobile network SHALL track the location of a mobile device and redo the commissioning step if the device enters a new geographic region with different keying material.

**Joining the Network**

To join the network, the device contacts the trust server via an out-of-band mechanism, i.e., the mobile network. The device then acquires the EPID and a network address, as well as the trust-center link key. The device then uses the trust center link key to securely acquire the network key via the trust center in the ZigBee network.

**Application Pairing of Devices**

When a mobile device wants to securely communicate with an access point, the access point shall initiate the ASKE key agreement protocol as described in the ASKE cluster. If the access point holds several different versions of keying material, it SHALL derive which keying material to use by the crypto-identifier of the mobile devices, i.e., the last 8 bits thereof.

If the communication needs to be started by the mobile device it shall initiate the ASKE key agreement protocol instead.

#### 5.7.3.1 Security Best Practices

Spatial Partitioning. When the number of participants exceeds a certain threshold, the deployment area SHOULD be divided into a number of networks. The number of networks and their configurations is to be defined by the network provider. In this case, different networks - which should be defined by geographic areas - use independent key material. The mobile network MAY distribute key materials from surrounding networks to the mobile device or the access points in a network to allow for soft roaming. New key material is distributed by the mobile network, which is also responsible for detecting which network the mobile device is in.

Temporal Key Updates. Depending on the mobility and number of mobile nodes, a periodic update of the keying material is recommended. Each static device shall thereby keep at least two version of the keying material, thus allowing for a smooth transition that does not require all mobile devices to be updated at once.

This page intentionally blank

# CHAPTER 6

# NETWORK LAYER SPECIFICATION

## 6.1 NWK Layer Service Specification

Constants, error codes and general alarms. Profile-specific constants are shown in Table 6.1.

**Table 6.1 Constants Specific to the TA Profile**

| Constant | Description | Value |
|---|---|---|
| Values of the *Physical Environment* attribute of the Basic Device Settings attribute set defined in the Basic cluster for use with this profile. | The application can set these values properly. However, default Descriptions and Values could be taken from [R18]. | See [R18] for default values |

**This page intentionally blank**

CHAPTER

# 7

# DEVICE SPECIFICATION

## 7.1 Common Clusters

Support for certain clusters is common to all the devices in this profile. The clusters shown in Table 7.1 shall be supported by all devices in this profile as mandatory or optional according the designation given here. Individual device descriptions may place further restrictions on support of the optional clusters shown here.

**Table 7.1 Clusters Common to All Devices**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| Basic | *None* |
| | |
| **Optional** | |
| Groups | |
| Manufacturer-specific (See 7.1.2 for details) | Manufacturer-specific (See 7.1.2 for details) |
| Commissioning[a] | |
| Identify[b] | |

a. CCB #1112

b. CCB #1116

### 7.1.1 Optional Support for Clusters With Reporting Capability

Some clusters support the ability to report changes to the value of particular attributes. These reports are typically received by the client side of the cluster. Support for client-side reporting is optional.

### 7.1.2 Manufacturer-Specific Clusters

The ZCL provides a range of cluster IDs that are reserved for manufacturer specific clusters. Manufacturer-specific clusters that conform to the requirements given in the ZCL may be added to any device description specified in this profile.

## 7.2 Feature and Function Description

Each device must support a certain set of features and functions. A table is used to specify the mandatory and optional features and functions of each device. This chapter contains a description of what must be supported if the feature or function is supported by the device. The mandatory or optional configuration for each device is described in the up coming chapters.

**Join (End Devices and Routers):**

Go find and join available TA network (see section 5.7)

**Form Network**

For devices that can start a network.

**Allow Others to Join Network (Router and Coordinator only):**

Allows you to add more nodes to an existing network.

**Restore to Factory Fresh Settings:**

Restore the device settings to fresh state. (also performs leave).

**Pair Devices (End Device Bind Request):**

If this feature is supported the device must provide a way for the user to issue an End device Bind Request.

**Enable Identify Mode:**

If this feature is supported the device must provide a way for the user to enable Identify for 60 seconds.

**Group Nodes (Add Group If Identify):**

If this feature is supported the device must provide a way for the user to send an "add Group if identifying Request",

**Create Scene (Store Scene):**

If this feature is supported the device must provide a way for the user to send an "Store Scene Req".

**Service Discovery (Match Descriptor Request):**

If this feature is supported the device must provide a way for device to send a match descriptor request, receive match descriptor responses and utilize them for commissioning the device.

**ZDP Bind Response:**

If this feature is supported the device must be able to receive a ZDP Bind Request and respond correctly with an ZDP Bind Response.

**ZDP Unbind Response:**

If this feature is supported the device must be able to receive a ZDP Unbind Request and respond correctly with an ZDP Unbind Response.

**End Device Annce/Device Annce:**

If this feature is supported the device must Send End Device Annce (ZigBee 2006)/ Send Device annce (ZigBee 2007) upon joining and re-joining a network.

**Service Discovery Response:**

If this feature is supported the device must be able to receive a Match descriptor request, and respond with a match descriptor response correctly.

## 7.3 Generic Devices

### 7.3.1 ZigBee SIM Card (Z-SIM)

The ZigBee SIM card is capable of running different services based on m-commerce, information delivery, data sharing, as mentioned in [R2]. The ZigBee SIM card in this case should be intended as a platform for ZigBee application; if we consider the Z-SIM hardware platform, different application objects could be implemented (so different profiles could be supported). In this case other clusters associated to different application objects (e.g., healthcare "Data collection Unit") could be supported.

#### 7.3.1.1 Supported Clusters

In addition to those specified in Table 7.1, the Z-SIM device shall support the clusters listed in Table 7.2. Both client and server ends of the Basic cluster are mandatory, so that the device can interrogate what other devices are present on the network, and so that other devices can also interrogate it if required. The client side of the Identify cluster is mandatory so that the device can instruct other devices to identify themselves.

At least one of the optional clusters must be implemented. The cluster shall follow the dependencies listed in 5.6.1.

**Table 7.2 Clusters Supported by the Z-SIM**

| **Server Side** | **Client Side** |
|---|---|
| **Mandatory** | |
| *None* | Basic |
| | Identify |
| | ISO7816 Protocol Tunnel |
| **Optional** | |
| Information | Information |
| Payment | Payment |
| | Billing |
| ASKE | ASKE |
| ASAC | ASAC |
| Data sharing | Data sharing |
| Voice over ZigBee | Voice over ZigBee |
| Data rate control | Data rate control |
| Chatting | Chatting |
| ISO7816 Protocol Tunnel | |
| RSSI Location[a] | |

a. CCB #1116

#### 7.3.1.2 Supported Features and Functions

The ZSIM device shall support the features and functions listed below.

**Table 7.3 Features and Functions Supported by the ZigBee SIM Card**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory/ Optional** | M | M | M | O | O | O | O |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory/ Optional** | O | O | M | O | O | M | M |

### 7.3.2 ZigBee Mobile Terminal

The ZigBee Mobile Terminal is capable of running different services based on m-commerce, information delivery, data sharing, VoZ, as mentioned in [R2]. It is a mobile terminal enabled with ZigBee (integrated transceiver but also enabled through the use of ZigBee card such as SDIO or Compact flash or others). The Mobile Terminal (intended as a ZigBee TA application) may be implemented as a Host application on a ZigBee Gateway (in the mobile terminal hardware acting as a gateway itself): the memory issues related to the number of clusters to be implemented (true for 8-bit micros implementations) don't apply for this case.

#### 7.3.2.1 Supported Clusters

In addition to those specified in Table 7.1, the ZigBee Mobile Terminal device shall support the clusters listed in Table 7.4. Both client and server ends of the Basic cluster are mandatory, so that the device can interrogate what other devices are present on the network, and so that other devices can also interrogate it if required. The client side of the Identify cluster is mandatory so that the device can instruct other devices to identify themselves.

At least one of the optional clusters must be implemented. The cluster shall follow the dependencies listed in 5.6.1.

**Table 7.4 Clusters Supported by the ZigBee Mobile Terminal**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| *None* | Basic |
| | Identify |
| **Optional** | |
| Information | Information |
| Payment | Payment |
| | Billing |
| ASKE | ASKE |
| ASAC | ASAC |
| Data sharing | Data sharing |
| Data Rate Control | Data Rate Control |
| Voice over ZigBee | Voice over ZigBee |
| Gaming | Gaming |
| Chatting | Chatting |
| ISO7816 Protocol Tunnel | ISO7816 Protocol Tunnel |
| Partition[a] | Partition |
| RSSI Location[b] | |

a. CCB #1116

b. CCB #1116

### 7.3.2.2 Supported Features and Functions

The ZigBee Mobile Terminal device shall support the features and functions listed below.

**Table 7.5 Features and Functions Supported by the Mobile Terminal**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory/ Optional** | M | M | M | O | O | O | O |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory/ Optional** | O | O | M | O | O | M | M |

## 7.3.3 Configuration Tool

The Configuration Tool device is capable of configuring other devices. This device is intended for configuring newly installed devices and may be used for performance optimization thereafter.

The intention of this specification is to define a generic configuration device type. All other clusters are optionally supported as clients and/or servers for this device type.

### 7.3.3.1 Supported Clusters

In addition to those specified in Table 7.1, the Configuration Tool device shall support all of the mandatory and at least one of the optional clusters listed in Table 7.6.

Both client and server forms of the Basic cluster are mandatory, so that the device can interrogate what other devices are present on the network, and so that other devices can also interrogate it if required. The Identify client cluster is mandatory so that the device can ask other devices to identify themselves.

**Table 7.6 Clusters Supported by the Configuration Tool**

| **Server Side** | **Client Side** |
|---|---|
| **Mandatory** | |
| *None* | Basic |
| | Identify |
| | Groups |
| **Optional** | |
| | Information |
| | Data Rate Control |
| | Data Sharing |
| | Payment |
| | Billing |
| | Voice over ZigBee |
| ASKE | ASKE |
| ASAC | ASAC |
| | RSSI Location[a] |
| | Partition [b] |

a. CCB #1116

b. CCB #1116

### 7.3.3.2 Supported Features and Functions

The Configuration Tool device shall support the features and functions listed below.

**Table 7.7 Features and Functions Supported by the Configuration Tool Device**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory/Optional** | M | M | M | O | O | O | O |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory/Optional** | M | M | O | O | O | M | M |

## 7.3.4 Range Extender

The Range Extender is a simple device which acts as a router for other devices. The Range Extender device shall not be a ZigBee end device. A product that implements the Range Extender devices shall not implement any other devices defined in this profile. This device shall only be used if the product is not intended to have any other application, or if a private application is implemented that has not been addressed by this profile. Range Extender shall be a router both for ZigBee and ZigBee PRO feature set networks even if its feature set does not correspond to one used within the network[6].

### 7.3.4.1 Supported Clusters

The Range Extender device shall support the mandatory common clusters listed in Table 7.1.

6. CCB #1140

#### 7.3.4.2 Supported Features and Functions

The Range Extender device shall support the features and functions listed below.

**Table 7.8 Features and Functions Supported by the Range Extender Device**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory/ Optional** | M | O | M | M | O | O | O |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory/ Optional** | O | O | O | O | O | M | M |

## 7.4 Information Devices

### 7.4.1 ZigBee Access Point[7]

The ZigBee Access Point can be used to deliver some information through ZigBee network to the connected ZigBee Mobile Terminals (see [R2]). In many use cases, ZigBee Access Point will also get some information from the ZigBee Mobile Terminals, and deliver to the application servers. In location services, ZigBee Access Point knows their own location, and exchanges some information between the ZigBee Mobile Terminals to estimate the location of the ZigBee Mobile Terminals.

7. CCB #1116

#### 7.4.1.1 Supported Clusters

In addition to those specified in Table 7.1, the ZigBee Access Point device shall support the clusters listed in Table 7.9.

**Table 7.9 Clusters Supported by the ZigBee AP**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| Information | Information |
| **Optional** | |
| Payment | Payment |
| ASKE | ASKE |
| ASAC | ASAC |
| Billing [a] | Billing |

a. CCB #1116

### 7.4.1.2 Supported Features and Functions

The ZigBee Access Points device shall support the features and functions listed below.

Table 7.10 Features and Functions Supported by the ZigBee AP Device

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory /Optional** | M | M | M | M | O | O | M |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory /Optional** | O | O | O | O | O | M | M |

## 7.4.2 ZigBee Information Nodes[8]

The ZigBee information node is a device that can be used in Information Services. Information nodes may maintain and update information and may communicate to ZigBee TA profile devices like ZigBee Mobile Terminal and ZigBee SIM card. Information maintained in the ZigBee information node can be updated through the operator network by using ZigBee Access Points (or ZigBee gateways using ZigBee Access Points clusters). In location services, ZigBee Information Nodes knows their own location, and exchanges some information between the ZigBee Mobile Terminals to estimate the location of the ZigBee Mobile Terminals.

8. CCB #1116

#### 7.4.2.1 Supported Clusters

In addition to those specified in Table 7.1, the ZigBee Information node device shall support the clusters listed in Table 7.11. The support of Billing cluster for the Information node is recommended.

**Table 7.11 Clusters Supported by the ZigBee Information Nodes**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| Information | *None* |
| **Optional** | |
| | Information |
| Billing | |
| ASKE | ASKE |
| ASAC | ASAC |

#### 7.4.2.2 Supported Features and Functions

The ZigBee Information Node device shall support the features and functions listed below.

**Table 7.12 Features and Functions Supported by the ZigBee Information Node Device**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory/ Optional** | M | O | M | M | O | O | M |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory/ Optional** | O | O | O | O | O | M | M |

## 7.4.3 ZigBee Information Terminal[9]

The ZigBee information Terminal is a device that can be used in Information Services. Information terminal may retrieve information from ZigBee TA profile devices like ZigBee Information Nodes or ZigBee Access Points. The Information Terminal device can be equipped with a Human Machine Interface such as a display or touch screen to allow the user to interact with the network. Information can be delivered to Information Terminal by ZigBee Information Nodes or by ZigBee Access Points using both unsolicited (Push) and solicited (Pull) procedures as defined in the Information cluster (see A.2). In case the ZigBee Information Terminal is not provided with a HMI (i.e. asset tracking tag) it shall support only Push information response.

9. CCB #1116

#### 7.4.3.1 Supported Clusters

In addition to those specified in Table 7.1, the ZigBee Information Terminal device shall support the clusters listed in Table 7.13.

**Table 7.13 Clusters Supported by the ZigBee Information Terminal**

| **Server Side** | **Client Side** |
|---|---|
| **Mandatory** | |
| *None* | Information |
| **Optional** | |
| *None* | *None* |

#### 7.4.3.2 Supported Features and Functions

The ZigBee Information Terminal device shall support the features and functions listed below.

**Table 7.14 Features and Functions Supported by the ZigBee Information Terminal Device**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory /Optional** | M | O | O | M | O | O | O |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory /Optional** | O | O | O | O | O | M | M |

## 7.5 Payments Devices

### 7.5.1 Point of Sale

The ZigBee Point of Sales (ZigBee POS) is a device that can be used for M-Commerce and payments services (e.g. mobile payments). It may communicate to ZigBee TA profile devices like ZigBee Mobile Terminal and ZigBee SIM card in order to operate payment transactions or deliver an electronic product code to complete required payment operations. ZigBee Points of Sales usually have telecommunications interfaces that enable the connection to the operator network towards a charging unit.

#### 7.5.1.1 Supported Clusters

In addition to those specified in Table 7.1, the ZigBee Point of Sale device shall support the clusters listed in Table 7.15.

**Table 7.15 Clusters Supported by the POS**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| Information | |
| Payment | |
| ASKE | ASKE |
| ASAC | ASAC |
| **Optional** | |
| *None* | *None* |

### 7.5.1.2 Supported Features and Functions

The ZigBee Point of Sale device shall support the features and functions listed below.

**Table 7.16 Features and Functions Supported by the ZigBee Point of Sale Device**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory /Optional** | M | M | M | M | O | O | M |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory /Optional** | O | O | O | O | O | M | M |

## 7.5.2 Ticketing Machine

The ZigBee Ticketing Machine is a device that can be used for M-Commerce services (e.g. mobile ticketing). It may communicate to ZigBee TA profile devices like ZigBee Mobile Terminal and ZigBee SIM card in order to operate payment transactions or trigger the terminal to complete required payment operations. ZigBee Ticketing machines usually have logging functionalities to be used in order to check the payments and the number of users accessing the service.

### 7.5.2.1 Supported Clusters

In addition to those specified in Table 7.1, the ZigBee Ticketing Machine device shall support the clusters listed in Table 7.17.

**Table 7.17 Clusters Supported by the Ticketing Machine**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| Information | |
| Payment | |
| ASKE | ASKE |
| ASAC | ASAC |
| **Optional** | |
| *None* | *None* |

### 7.5.2.2 Supported Features and Functions

The ZigBee Ticketing Machine device shall support the features and functions listed below.

**Table 7.18 Features and Functions Supported by the ZigBee Ticketing Machine Device**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory /Optional** | M | M | M | M | O | O | M |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory /Optional** | O | O | O | O | O | M | M |

## 7.5.3 Pay Controller

The ZigBee Pay controller is a device that can be used inside M-Commerce services (e.g. mobile ticketing) in order to perform control functionalities. It may communicate to ZigBee TA profile devices like ZigBee Mobile Terminal and ZigBee SIM card in order to check the transactions or trigger the terminal to complete required payment operations. ZigBee Pay-controller machines may have logging functionalities to be used in order to store data related to the checked users.

### 7.5.3.1 Supported Clusters

In addition to those specified in Table 7.1, the ZigBee Pay controller device shall support the clusters listed in Table 7.19.

**Table 7.19 Clusters Supported by the Pay Controller**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| Information | Information |
| ASKE | ASKE |
| ASAC | ASAC |
| **Optional** | |
| Payment | Payment |

### 7.5.3.2 Supported Features and Functions

The ZigBee Pay controller device shall support the features and functions listed below.

Table 7.20 Features and Functions Supported by the ZigBee Pay Controller Device

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory/ Optional** | M | M | M | M | O | O | M |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory/ Optional** | O | O | O | O | O | M | M |

## 7.5.4 Billing Unit

The ZigBee Billing unit is a device (or a ZigBee Gateway performing TA functionalities) that can be used inside billing services (e.g. zone billing) in order to perform billing functionalities. It may communicate to ZigBee TA profile devices like ZigBee Access Points, ZigBee information nodes, ZigBee Mobile Terminal and ZigBee SIM card in order to compute the bill to the corresponding services or connection time.

### 7.5.4.1 Supported Clusters

In addition to those specified in Table 7.1, the ZigBee Billing unit device shall support the clusters listed in Table 7.21.

**Table 7.21 Clusters Supported by the Billing Unit**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| Information | Information |
| ASKE | ASKE |
| ASAC | ASAC |
| Billing | Billing |
| **Optional** | |
| *None* | *None* |

### 7.5.4.2 Supported Features and Functions

The ZigBee Billing unit shall support the features and functions listed below.

**Table 7.22 Features and Functions Supported by the ZigBee Billing Unit**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory/ Optional** | M | M | M | M | O | O | M |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory/ Optional** | O | O | O | O | O | M | M |

### 7.5.5 Charging Unit

It supports the same clusters as the Billing unit but the Charging unit device operates charging functionalities (i.e. a financial transactions). Besides a dedicated device ID for the Charging unit allows other devices such as Mobile terminal or Z-SIM to run a service discovery (SimpleDescriptorRequest) based directly on the device ID itself.

## 7.6 Memory Devices

### 7.6.1 ZigBee Flash Card

The ZigBee Flash Card is an integration of a ZigBee module and a flash card (e.g. SD/ MMC/CF/MS card, etc.). ZigBee Flash card maybe a memory card (e.g. Multimedia Card) and it could be used as a remote memory and accessed by other Telecom Applications Profile Devices (like Z-SIM, ZigBee Mobile Terminal or Z-VDT) by using, for instance, the Small data sharing functionalities as described in [R2].

#### 7.6.1.1 Supported Clusters

In addition to those specified in Table 7.1, the ZigBee flash cards device shall support the clusters listed in Table 7.23. The ZigBee flash card devices uses the On/Off cluster in order to operate the memory lock/unlock; the On/Off commands can be generated by ZigBee TA devices (i.e. Z-SIM or Mobile Terminal) that require the control of the flash card.

**Table 7.23 Clusters Supported by the ZigBee Flash Cards**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| On/Off | *None* |
| **Optional** | |
| ASKE | ASKE |
| ASAC | ASAC |
| Data Sharing | Data Sharing |
| Data Rate Control | Data Rate Control |
| Information | |
| Partition[a] | Partition |

a. CCB #1116

### 7.6.1.2 Supported Features and Functions

The ZigBee Flash card device shall support the features and functions listed below.

**Table 7.24 Features and Functions Supported by the ZigBee Flash Card**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory/ Optional** | M | M | M | M | O | O | M |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory/ Optional** | O | O | O | O | O | M | M |

## 7.6.2 ZigBee PC Smart Card Reader

The ZigBee PC Smart Card Reader (PCSC) is used to establish a connection between the PC and a remote smart card using ZigBee (e.g. a ZSIM card might be used as a remote smart card for mobile office scenarios).

#### 7.6.2.1 Supported Clusters

In addition to those specified in Table 7.1, the ZigBee PC Smart card reader shall support the clusters listed in Table 7.25.

**Table 7.25 Clusters Supported by the ZigBee PCSC**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| ISO7816 Protocol Tunnel | *None* |
| **Optional** | |
| ASKE | ASKE |
| ASAC | ASAC |

#### 7.6.2.2 Supported Features and Functions

The ZigBeePC Smart Card Reader device shall support the features and functions listed below.

**Table 7.26 Features and Functions Supported by the ZigBee PC Smart Card Reader**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory/ Optional** | M | O | O | O | O | O | M |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory/ Optional** | O | O | O | O | O | M | M |

## 7.7 Voice Devices

### 7.7.1 ZigBee Headset

The ZigBee headsets are used for delivering voice from/to ZigBee mobile terminals. It consists of a voice codec module, a microphone, a mini speaker and a ZigBee module.

#### 7.7.1.1 Supported Clusters

In addition to those specified in Table 7.1, the ZigBee Headset shall support the clusters listed in Table 7.27.

**Table 7.27 Clusters Supported by the ZigBee Headset**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| Voice over ZigBee | Voice over ZigBee |
| **Optional** | |
| ASKE | ASKE |
| ASAC | ASAC |

#### 7.7.1.2 Supported Features and Functions

The ZigBee headset device shall support the features and functions listed below.

**Table 7.28 Features and Functions Supported by the ZigBee Headset**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory /Optional** | M | M | M | M | O | O | O |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory /Optional** | O | O | O | O | O | M | M |

## 7.7.2 ZigBee Microphone

The ZigBee microphones are used for delivering voice to ZigBee mobile terminals or speaker. It consists of a microphone and a ZigBee module. A ZigBee microphone can be used as a standalone or as a built-in system.

### 7.7.2.1 Supported Clusters

In addition to those specified in Table 7.1, the ZigBee microphone shall support the clusters listed in Table 7.29.

**Table 7.29 Clusters supported by the ZigBee Microphone**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| *None* | Voice over ZigBee |
| **Optional** | |
| ASKE | ASKE |
| ASAC | ASAC |

### 7.7.2.2 Supported Features and Functions

The ZigBee microphone device shall support the features and functions listed below.

**Table 7.30 Features and Functions Supported by the ZigBee Microphone**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory /Optional** | M | M | M | M | O | O | O |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory /Optional** | O | O | O | O | O | M | M |

### 7.7.3 ZigBee Speaker

The ZigBee speaker is used for receiving voice using ZigBee (from a ZigBee microphone or mobile terminal). It consists of a speaker and a ZigBee module. A ZigBee speaker may be for instance installed in home appliances like a refrigerator or a washer.

#### 7.7.3.1 Supported Clusters

In addition to those specified in Table 7.1, the ZigBee speaker shall support the clusters listed in Table 7.31.

**Table 7.31 Clusters Supported by the ZigBee Speaker**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| Voice over ZigBee | *None* |
| **Optional** | |
| ASKE | ASKE |
| ASAC | ASAC |

#### 7.7.3.2 Supported Features and Functions

The ZigBee speaker device shall support the features and functions listed below.

**Table 7.32 Features and Functions Supported by the ZigBee Speaker**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory /Optional** | M | M | M | M | O | O | O |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory /Optional** | O | O | O | O | O | M | M |

## 7.8 Localization Devices[10]

### 7.8.1 RSSI Anchor Node

The RSSI Anchor node is a device that can be used as reference to estimate the position of a mobile node (RSSI Location device) in the network. This device shall support the client side of RSSI location cluster; since the Anchor Node doesn't typically support a HMI, it should support only the following commands and it is not required implement the ZCL Foundation commands:

- Commands generated from Anchor Node:
  - Anchor Node Announce
  - RSSI Response

10. CCB #1116

- Commands received by Anchor Node:
    - RSSI Ping
    - RSSI Request

### 7.8.1.1 Supported Clusters

In addition to those specified in Table 7.1, the RSSI Anchor Node shall support the clusters listed in Table 7.33.

**Table 7.33 Clusters Supported by the RSSI Anchor Node**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| *None* | RSSI Location |
| **Optional** | |
| *None* | *None* |

### 7.8.1.2 Supported Features and Functions

The RSSI Anchor Node device shall support the features and functions listed below.

**Table 7.34 Features and Functions Supported by the RSSI Anchor Node**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory /Optional** | M | O | O | O | O | O | O |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory /Optional** | O | O | O | O | O | M | M |

## 7.8.2 RSSI Location Node

The RSSI Location node is a device moving in an area which position can be estimated using RSSI location mechanism. This device shall support the server side of RSSI location cluster and can use centralized location procedures or distributed location methods as described in ZCL RSSI Location Cluster.

### 7.8.2.1 Supported Clusters

In addition to those specified in Table 7.1, the RSSI Location Node shall support the clusters listed in Table 7.35.

**Table 7.35 Clusters Supported by the RSSI Location Node**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| RSSI Location | *None* |
| **Optional** | |
| *None* | *None* |

### 7.8.2.2 Supported Features and Functions

The RSSI Location Node device shall support the features and functions listed below.

**Table 7.36 Features and Functions Supported by the RSSI Location Node**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory/ Optional** | M | O | O | O | O | O | O |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory/ Optional** | O | O | O | O | O | M | M |

## 7.8.3 RSSI Location Gateway

The RSSI Location Gateway is a ZigBee gateway device that collects RSSI measurements from RSSI Location Nodes and may send this information to a centralized server that could estimate the position of the RSSI Location node using RSSI localization algorithms. This device shall support the client side of RSSI location cluster and shall use centralized location procedures or distributed location methods as described in ZCL RSSI Location Cluster.

### 7.8.3.1 Supported Clusters

In addition to those specified in Table 7.1, the RSSI Location Gateway shall support the clusters listed in Table 7.37.

**Table 7.37 Clusters Supported by the RSSI Location Gateway**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| *None* | RSSI Location |
| **Optional** | |
| *None* | *None* |

### 7.8.3.2 Supported Features and Functions

The RSSI Location Gateway device shall support the features and functions listed below.

**Table 7.38 Features and Functions Supported by the RSSI Location Node**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory/ Optional** | M | M | M | O | O | O | O |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory/ Optional** | O | O | O | O | O | M | M |

## 7.9 Chatting Devices[11]

### 7.9.1 Chatting Unit

The Chatting Unit is a device that can be used to operate chat with other users. The Chatting Unit is usually provided with a HMI and can be a game console, a Personal Computer, or any another terminal usable for chatting via ZigBee network. This device shall support the client and server side of chatting cluster.

#### 7.9.1.1 Supported Clusters

In addition to those specified in Table 7.1, the shall support the clusters listed in Table 7.39.

**Table 7.39 Clusters Supported by the Chatting Unit**

| **Server Side** | **Client Side** |
|---|---|
| **Mandatory** | |
| Chatting | Chatting |
| **Optional** | |
| *None* | *None* |

11. CCB #1116

#### 7.9.1.2 Supported Features and Functions

The Chatting Unit device shall support the features and functions listed below.

Table 7.40 Features and Functions Supported by the Chatting Unit

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory /Optional** | M | O | O | O | O | O | O |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory /Optional** | O | O | O | O | O | M | M |

### 7.9.2 Chatting Station

The Chatting Station is a device that can be used to manage the chatting operated among Chatting Units and it is commonly used to enable the centralized scenario of chatting service (see A.10). It shall support the server side of Chatting cluster in order to manage the operations between the Chatting Units.

### 7.9.2.1 Supported Clusters

In addition to those specified in Table 7.1, the Chatting Station shall support the clusters listed in Table 7.41.

**Table 7.41 Clusters Supported by the Chatting Station**

| Server Side | Client Side |
|---|---|
| **Mandatory** | |
| Chatting | *None* |
| **Optional** | |
| *None* | *None* |

### 7.9.2.2 Supported Features and Functions

The Chatting Station shall support the features and functions listed below.

**Table 7.42 Features and Functions Supported by the Chatting Station**

| **Device Type/ Feature or function** | Join (end devices and routers only) | Form Network (coordinator only) | Allow Others to Join Network (routers and coordinators only) | Restore to Factory Fresh Settings | Pair Devices - (End Device Bind Request) | Bind Manager - (End Device Bind Response - Coordinator only) | Enable Identify Mode |
|---|---|---|---|---|---|---|---|
| **Mandatory/ Optional** | M | M | M | O | O | O | O |
| **Device Type/ Feature or function** | Group Nodes (send out an Add group If Identify) | Create Scene (Store Scene) | Service discovery (Match Descriptor Request) | ZDP Bind Response | ZDP Unbind Response | End Device Annce/ device annce | Service Discovery response (Match Descriptor Response) |
| **Mandatory/ Optional** | O | O | O | O | O | M | M |

# ANNEX A

# CANDIDATE MATERIAL FOR ZCL TO BE USED IN TA

## A.1 Partition Cluster

### A.1.1 Scope and Purpose

This section specifies a single cluster, the Partition Cluster, which provides commands and attributes for enabling partitioning of large frame to be carried from other clusters of ZigBee devices. This cluster is designed to provide a standardized interface for the applications to manage extended size frame format, up to 100KB long. The Partition Cluster can be used in different application scenarios that requires extended frame for services provided by particular clusters.

This section should be used in conjunction with the ZigBee Cluster Library, Foundation Specification (see [R4]]), which gives an overview of the library and specifies the frame formats and general commands used therein.

### A.1.2 Introduction

The cluster specified in this may be used in different application domains. The Partition Cluster provides the attributes and commands required for enabling and managing the transmission of extended frames over a ZigBee network.

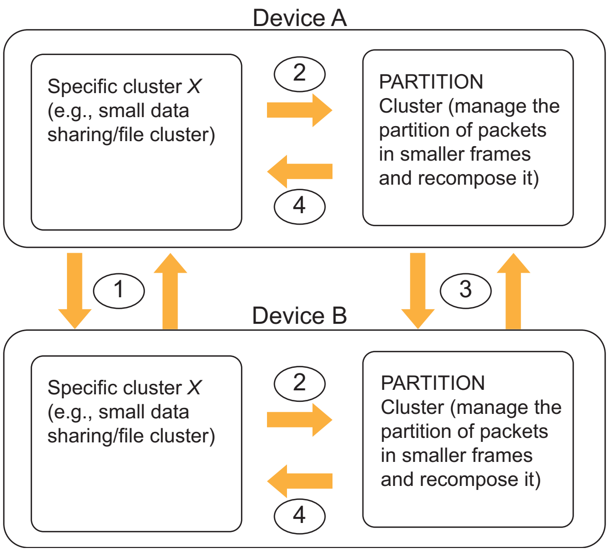

Device A
Specific cluster X (e.g., small data sharing/file cluster)
2
4
PARTITION Cluster (manage the partition of packets in smaller frames and recompose it)
1
3
Device B
Specific cluster X (e.g., small data sharing/file cluster)
2
4
PARTITION Cluster (manage the partition of packets in smaller frames and recompose it)

**Figure A.1** Typical Usage of the Partition Cluster

The typical usage of Partition Cluster is shown in Figure A.1 and can be represented by the following phases:

**1** Cluster based Discovery (e.g. performing Match_Desc_Req) can be operated to the specific cluster X that needs to transfer information to a matching cluster (e.g. File Cluster); moreover cluster based discovery should be used in order to check the support of the Partition Cluster by a recipient device;

**2** If the application entity requires transmission of large frames (e.g. an application willing to use data sharing/file cluster, generic tunnel cluster) the specific application entity shall subscribe to the Partition Cluster; registration or subscription phase is described in A.1.5.

**3** The Partition Cluster will perform and manage the “fragmentation” and send the rebuilt frame to the registered specific cluster;

**4** The Partition Cluster will forward the recomposed packet to the specific clusters that registered to the Partitioning Cluster (e.g. Cluster X).

The application object implementing and using the Partition Cluster should have enough memory to manage the incoming frames; the Partition Cluster is designed for devices like Mobile Phones or other gateways that have extended computing capabilities in comparison with typical ZigBee devices.

Since the Partition Cluster performs a handshake phase between the devices using Partition Cluster (reading and writing the proper defined attributes) as described with more details in A.1.5, both client and server should be used in order to guarantee a full bidirectional link in the communication (see Figure A.2).

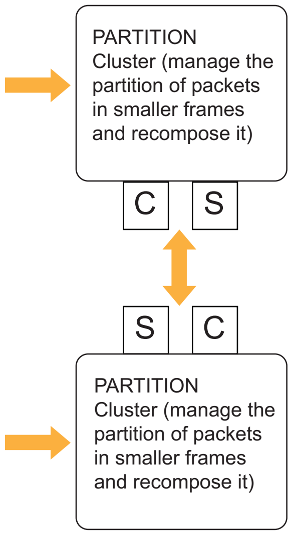

PARTITION Cluster (manage the partition of packets in smaller frames and recompose it)
C
S
S
C
PARTITION Cluster (manage the partition of packets in smaller frames and recompose it)

**Figure A.2** Client and Server in Partition Cluster

A simple way to enable the use of the Partition Cluster should be to define a specific API that would support the sending/receive functionalities through the use of Partition Cluster. Partition should be considered like a specific tunnel cluster: commands exposed to the application objects (general API to be used by the application) should be the following ones:

- TransferFrameUsingPartitionCluster (send/receive) → the max size for the carried data is typically 25KB<x<100KB as from discussed requirements. This command may pass a handler to the sequence of bytes corresponding to the ZCL message of the specific cluster using the Partition Cluster. In order to operate using the Partition Cluster the application may want to manage the transmission and reception of large frames running the handshake phase described in A.1.5.

  Rather than pushing the large frame to the application, the Partition Cluster may only inform the application that a packet has arrived (very short packet that can be fed through the stack). The application will then read the frame from the Partitioning Cluster. The detailed mechanism to perform this operation is out of scope of this specification

- Read/Write handshake commands

Partition Cluster related commands should be sent transparently between the ZigBee application objects managing the fragmentation to guarantee the reconstruction of the received frame; these commands are described in the following sections:

- *Transfer partitioned frame* (max dimension<max size carried by the ZCL standard frame ~80B)
- Multiple ACKs

In the Partition Cluster attributes, a list of registered clusters should be inserted in order to manage possible sharing and re-use of it by multiple clusters.

## A.1.3 Server

### A.1.3.1 Dependencies

None.

### A.1.3.2 Attributes

The attributes are used in the Partition Cluster summarized in Table A.1.

**Table A.1 Attributes of the Partition Cluster**

| Identifier | Name | Type | Range | Access | Default | Mandatory/ Optional |
|---|---|---|---|---|---|---|
| 0x0000 | *MaximumIncomingTransferSize* | Unsigned 16- bit integer | 0x0000-0xffff | ReadOnly | 0x0500 | M |
| 0x0001 | *MaximumOutgoingTransferSize* | Unsigned 16-bit integer | 0x0000-0xffff | ReadOnly | 0x0500 | M |
| 0x0002 | *PartionedFrame Size* | Unsigned 8-bit integer | 0x00-0xff | Read/ Write | 0x50 | M |
| 0x0003 | *LargeFrameSize* | Unsigned 16-bit integer | 0x0000-0xffff | Read/ Write | 0x0500 | M |
| 0x0004 | *NumberOfACKFrame* | Unsigned 8-bit integer | 0x00-0xff | Read/ Write | 0x64 | M |
| 0x0005 | *NACK Timeout* | Unsigned 16-bit integer | 0x0000-0xffff | Read | *apsAckWaitDuration + Interframe Delay * NumberOfACKFrames* | M |
| 0x0006 | *Interframe Delay* | Unsigned 8-bit Integer | Default-0xff | Read/ Write | *apsInterFrameDelay* | M |

Table A.1 Attributes of the Partition Cluster (Continued)

| Identifier | Name | Type | Range | Access | Default | Mandatory/ Optional |
|---|---|---|---|---|---|---|
| 0x0007 | *NumberOfSend Retries* | Unsigned 8-bit integer | 0x00-0xff | ReadOnly | 0x03 | M |
| 0x0008 | *SenderTimeout* | Unsigned 16-bit integer | Default-0xffff | ReadOnly | 2*apsAck WaitDurati on + Interframe Delay * *NumberOfA CKFrame*s | M |
| 0x0009 | *ReceiverTimeout* | Unsigned 16-bit integer | Default-0xffff | ReadOnly | *apsAckWait Duration+ Interframe Delay +NumberOf SendRetries * NACKTime out* | M |
| 0x000a-0xffff | Reserved | | | | | |

### A.1.3.2.1 *MaximumIncomingTransferSize* Attribute

The *MaximumIncomingTransferSize* attribute specifies the maximum size, as multiple of *PartionedFrameSize*, of the application service data unit (ASDU), that can be transferred to this node in one single message transfer. The ASDU referred to is the ZCL frame, including header and payload, of any command received by a Partition Cluster on the same endpoint.

### A.1.3.2.2 *MaximumOutgoingTransferSize* Attribute

The *MaximumOutgoingTransferSize* attribute specifies the maximum size, as multiple of *PartionedFrameSize*, of the application service data unit (ASDU), that can be transferred from this node in one single message transfer. The ASDU referred to is the ZCL frame, including header and payload, of any command received by a Partition Cluster on the same endpoint.

### A.1.3.2.3 *PartionedFrameSize* Attribute

The *PartitionedFrameSize* attribute specifies the size in bytes of a partitioned frame transferred using TransferPartitionedFrame command. The default value for this attribute is equal to 80 bytes (0x50) because a "large frame" to be transferred using the Partition Cluster shall be partitioned into smaller *PartitionedFrameSize* frame size.

#### A.1.3.2.4 *LargeFrameSize* Attribute

The *LargeFrameSize* attribute specifies the size, in multiple of *PartionedFrameSize*, of a large frame to be partitioned using the Partition Cluster into *PartionedFrameSize* bytes carried by TransferPartitionedFrame commands. The default value of this attribute should be set equal to 0x0500 (so that, given the default *PartitionedFrameSize* attribute equal to 80bytes the default large frame would be 100KB). The length in byte of the large frame to be partitioned is equal to *PartitionedFrameSize*LargeFrameSize*. In case the frame to be partitioned is not multiple of *PartitionedFrameSize*LargeFrameSize*, the last TransferPartionedFrame command shall be padded with zeros in order to fit in *PartitionedFrameSize* length of the last *TransferPartionedFrame* command.

#### A.1.3.2.5 *NumberOfACKFrame* Attribute

The *NumberOfACKFrame* attribute specifies the number of partitioned frames to be received before sending a multiple acknowledge command. The proper setting of this attribute guarantee the reduction of acknowledge packet to be transmitted over the network. If *NumberOfAckFrame* attribute is set to 0x00, it indicates an non-ACK transmission. In this case, the sender would ignore the sender timeout and send the blocks continuously with InterframeDelay interval between each partitioned frame. In this case the receiver shall not return the MultipleACK after receiving the block, and the *ReceiverTimeout* and *NACKTimeout* attributes (set to the receiver) shall be also ignored.

#### A.1.3.2.6 *NACKTimeout* Attribute

*NACKTimeout* attribute specifies the maximum time, expressed in milliseconds, the receiver entity should wait after having received the last *NumberOfACKFrame* partitioned frames, before sending a MultipleACK command to the sender. The receiver shall transmit immediately if it receives all the partitioned frames correctly.

#### A.1.3.2.7 *InterFrameDelay* Attribute

The *InterFrameDelay* attribute specifies the delay in milliseconds between successive transmissions of TransferPartionedFrame commands. Default value for this attributes is given by the apsInterFrameDelay. 0x00 is not a valid value for this attribute. If the device doesn't support APS fragmentation but supports the Partition Cluster, this value shall be set to 10ms.

#### A.1.3.2.8 *NumberOfSendRetries* Attribute

The *NumberOfSendRetries* specifies the maximum number of retries the sender should perform in case no MultipleACK have been received in SenderTimeout time period. This attribute should be reset to the default value when a MultipleACK command is received.

#### A.1.3.2.9 *SenderTimeout* Attribute

The *SenderTimeout* attribute specifies is the time that the sender should wait for the MultipleACK before sending a number of *NumberOfACKFrame* of TransferPartitionedFrame commands again. This attribute should be reset to the default value when a MultipleACK command is received and started with the first block sent to the receiver.

#### A.1.3.2.10 *ReceiverTimeout* Attribute

The *ReceiverTimeout* attribute specifies the maximum time the receiver need to wait for a TransferPartitionedFrame command after the reception the first frame of the large frame to be transferred. If there will be no frames received after *ReceiverTimeout*, the receiver will exit the Partition procedure.

### A.1.3.3 Commands Received

The received command IDs for the Partition Cluster are listed in Table A.2.

**Table A.2 Server Received Command IDs for the Partition Cluster**

| Command Identifier Field Value | Description | Mandatory/Optional |
|---|---|---|
| 0x00 | TransferPartitionedFrame | M |
| 0x01 | ReadHandshakeParam | M |
| 0x02 | WriteHandshakeParam | M |
| 0x03 - 0xff | Reserved | |

#### A.1.3.3.1 TransferPartitionedFrame Command

The TransferPartitionedFrame command is used to send a partitioned frame to another Partition Cluster. It shall be originated by the sender device and sent to the recipient device which is expected to answer with a MultipleACK (as defined in A.1.3.4.1). When the sender composes and sends to the receiver the first TransferPartitionedFrame command, a timer on the sender is started; this timer shall be used to check if the sender received a MultipleACK before SenderTimeout time period. The sender may wait for a MultipleACK after every *NumberOfACKFrame* blocks transmission. In that case the value *NumberOfACKFrame* should be set in a handshake phase. The sender will consider a successful transmission of a *NumberOfACKFrame* number of blocks if no *NACKId*s are carried by the MultipleACK command payload.

The TransferPartitionedFrame command shall be formatted as illustrated in Figure A.3.

| Octets: | Variable | 1 octet | 1-2 octet | Variable |
|---|---|---|---|---|
| Data Types | - | 8-bit bitmap | Unsigned 8/16-bit integer | Octet string |
| Field Name | ZCL Header | *Fragmentation Options* | *PartitionIndicator* | *PartitionedFrame* |

**Figure A.3** Format of the TransferPartitionedFrame Command

The *Fragmentation Options* field shall be formatted as illustrated in the following figure.

| b0: 1 bit | b1: 1 bit | b2-b7: 6 bit |
|---|---|---|
| First block | Indicator length | Reserved |

**Figure A.4** Format of the FragmentationOptions Field

First Block field *b0=1* indicates that the TransferPartitionedFrame command carries the first block of the overall transfer while b0=0 indicates that the TransferPartitionedFrame command doesn't carry a first block. Indicator length field specifies if the PartitionIndicator field is 1 or 2-bytes long: b1=0 indicates that the PartitionIndicator is 1-byte long, b1 = 1 indicates that the PartitionIndicator is 2-bytes long.

PartitionIndicator field specifies the overall number of blocks for the 1st partitioned frame (fragment), and the block index for the other fragments starting from 0x01 or 0x0001 (respectively for b1=0 or b1 = 1).

The address mechanism used for the TransferPartitionedFrame command should not use broadcasting and it should not use multicasting.

### A.1.3.3.1.1 Effect on Receipt

The receiver will start receiving TransferPartitionedFrame commands and start the *NACKTimeout* and *ReceiverTimeout* timers after the reception of the first frame related to the transaction registered by the handshake phase (WriteHandshakeParam command); if *NumberOfACKFrames* have been received, the Partition Cluster of the receiver will send a MultipleACK command with no *NACKId*. The block indexes of expected TransferPartitionedFrame commands that have not been received in *NACKTimeout* (*NACKIds*) will be inserted in the MultipleACK command returned to the sender. If there are no frames received after *ReceiverTimeout*, the receiver will exit the partition procedure. In case the receiver receives a number equal to *NumberOfACKFrame* partitioned frames it shall send the MultipleACK command without waiting for a *NACKTimeout* time.

The receiver will also reset the *ReceiverTimeout* timer after reception of a TransferPartitionedFrame command.

#### A.1.3.3.2 ReadHandshakeParam Command

The ReadHandshakeParam command is used in order to read the appropriate set of parameters for each transaction to be performed by the Partition Cluster. The Partitioned ClusterID field identifies the specific cluster referred to the large frame that is going to be partitioned by the Partition Cluster itself. The transaction number of the specific frame to be partitioned shall be carried directly in the ZCL header.

| **Octets:** | **Variable** | **2** | **2** | **…** | **2** |
|---|---|---|---|---|---|
| Data Types | - | ClusterID | AttributeID | … | AttributeID |
| Field Name | ZCL Header | Partitioned ClusterID | Attribute identifier 1 | … | Attribute identifier n |

**Figure A.5** Format of the ReadHandshakeParam Frame

#### A.1.3.3.3 WriteHandshakeParam Command

The WriteHandshakeParam command is used during the handshake phase in order to write the appropriate parameters for each transaction to be performed by the Partition Cluster. The Partitioned ClusterID field identifies the specific cluster referred to the frames that is going to be partitioned by the Partition Cluster itself. The transaction number of the specific frame to be partitioned shall be carried in the ZCL header. See section 2.4.3 of [R4] for write attribute record format. By using the WriteHandshakeParam command report it is possible to write Partition Cluster attributes related to the specific large frame to be transferred using partitioning.

| **Octets:** | **Variable** | **2** | **2** | **…** | **2** |
|---|---|---|---|---|---|
| Data Types | - | ClusterID | See [R4] | … | See [R4] |
| Field Name | ZCL Header | Partitioned ClusterID | Write Attribute Record 1 | | Write Attribute Record *n* |

**Figure A.6** Format of the WriteHandshakeParam Frame

| **Octets: 2** | **1** | **Variable** |
|---|---|---|
| Attribute identifier | Attribute Data Type | Attribute data |

**Figure A.7** Format of theWrite Attribute Record Field

### A.1.3.4 Commands Generated

The generated command IDs for the server Partition Cluster are listed in Table A.3.

**Table A.3 Generated Command IDs for the Partition Cluster**

| Command Identifier Field Value | Description | Mandatory/ Optional |
|---|---|---|
| 0x00 | MultipleACK | M |
| 0x01 | ReadHandshakeParamResponse | M |
| 0x02 - 0xff | Reserved | |

#### A.1.3.4.1 MultipleACK Command

The receiver shall return the MultipleACK command when receiving a number equal to *NumberOfACKFrame* TransferPartitionedFrame commands (partitioned frames) or when NACKTimeout expires. The MultipleACK command will carry no *NACKId* in the payload if *NumberOfACKFrame* TransferPartitionedFrame commands are received. The sender may wait for a MultipleACK command after every *NumberOfACKFrame* blocks transmission. The MultipleACK command shall be formatted as illustrated in Figure A.8.

| Octets: | 1 | 1-2 | 1-2 | 1-2 | 1-2 |
|---|---|---|---|---|---|
| Data Types | 8-bit bitmap | Unsigned 8/ 16-bit integer | Unsigned 8/16-bit integer | | Unsigned 8/16-bit integer |
| Field Name | ACK Options | FirstFrameID | *NACKId* | … | *NACKId* |

**Figure A.8** Format of the MultipleACK Command

The ACKOptions payload fields shall be formatted as illustrated in the following figure.

| b0: 1 bit | b1-b7: 7 bit |
|---|---|
| NACKId length | Reserved |

**Figure A.9** Format of the ACK Options Field

NACKId length specifies if the NACKId corresponding to the PartitionIndicator (NACKIds carried in this command are the values of the PartitionIndicator field in Figure A.3), and the FirstFrameID are 1 or 2 bytes long: b0=0 indicates that the

NACKIds and the FirstFrameID are 1-byte long, b0 = 1 indicates that the NACKIds and the FirstFrameID are 2-bytes long.

FirstFrameID field indicates the first partition frame (block) index of the current overall *NumberOfACKFrame* blocks the MultipleACK refers to. It is used in order to identify the set of *NumberOfACKFrame* the MultipleACK command refers to.

NACKId fields represent the ID of partitioned frame that have not been received yet after *NACKTimeout*.

#### A.1.3.4.1.1 Effect on Receipt

After sending a number of TransferPartitionedFrame commands equal to *NumberOfACKFrame* (Number of acknowledged frames) the sender will wait for a MultipleACK: a successful transmission is indicated by a MultipleACK command with no NACKId fields carried. The sender shall stop sending the next *NumberOfACKFrame* blocks until it receives a MultipleACK command reporting a successful transmission.

When the sender successfully sends the current *NumberOfACKFrame* blocks and receives a MultipleACK command with no NACKId fields, the Partition Cluster should proceed to send the next *NumberOfACKFrame* set of blocks of the large frame to be transmitted, until all the set of blocks have been sent out. The partition parameters such as *NumberOfACKFrame* may be tuned after sending out the current *NumberOfACKFrame*, set of blocks (e.g. the value of *NumberOfACKFrame* may be decreased after retransmissions of many TransferPartitionedFrame commands of a previous transaction).

In case the receiver does need to send out several MultipleACKs to the sender, it should not send out a next one until completing the reception of all blocks indicated in the NACKId fields of the previous MultipleACK. The sender should receive MultipleACK command by sender timeout (this timeout specifies how long to wait for a MultipleACK); if no MultipleACK command is received the sender will retransmit the TransferPartitionedFrame commands up to a maximum number of retries (in order to optimize the protocol the sender may reduce also the *NumberOfACKFrame* value by using the writing command defined in the handshake phase); if the sender doesn't receive any MultipleACK after maximum number of retries it will exit the partition procedure and the TransferFrameUsingPartitionCluster response will notify the error in the partition procedure; otherwise, if MultipleACK is received carrying some NACK IDs, the sender will reset the sender timeout and the max number of retries and resend the no acknowledged TransferPartitionedFrame commands up to max number of retries until a MultipleACK with no NACK is received (success in the partition transaction) or the SenderTimeout expires (in case no MultipleACK commands are received) or max number of retries reached (in case MultipleACK commands are received but still with NACKIds).

The *SenderTimeout* is equal to *2*apscAckWaitDuration* + *InterframeDelay* * *NumberOfACKFrame*s.

### A.1.3.4.2 ReadHandshakeParamResponse Command

The ReadHandshakeParamResponse command is used to respond to the corresponding ReadHandshakeParam command in order to communicate the appropriate set of parameters configured for each transaction to be performed by the Partition Cluster. The Partitioned ClusterID field identifies the specific cluster referred to the large frame that is going to be partitioned by the Partition Cluster itself. The transaction number of the specific frame to be partitioned shall be carried directly in the ZCL header. The *Read* Attribute status record field is the same as defined for the ZCL (see 2.4.2.1 of [R4]).

| **Octets:** | **Variable** | **2** | **Variable** | **…** | **Variable** |
|---|---|---|---|---|---|
| Data Types | - | ClusterID | See [R4] | … | See [R4] |
| Field Name | ZCL Header | Partitioned ClusterID | *Read* attribute status record 1 | … | *Read* attribute status record n |

**Figure A.10** Format of the ReadHandshakeParamResponse Frame

| **Octets: 2** | **2** | **0/1** | **0/Variable** |
|---|---|---|---|
| Attribute identifier | Status | Attribute Data Type | Attribute data |

**Figure A.11** Format of the Read Attribute Status Record Field

## A.1.4 Client

### A.1.4.1 Attributes

None.

### A.1.4.2 Commands Received

The client receives the cluster-specific commands detailed in sub-clause A.1.3.4, as required by application profiles.

### A.1.4.3 Commands Generated

The client generates the cluster-specific commands detailed in sub-clause A.1.3.3 as required by application profiles.

## A.1.5 General Use of Partition Cluster

The Partition Cluster may be used by multiple clusters defined in a single application object. In order to perform the recognition of multiple partitioned frames associated to a specific cluster and reconstruct a partitioned large frame the Partition Cluster shall maintain an internal table similar to the one presented in the Table A.4: each large frame to be partitioned can be identified by the ClusterID and the ZCL transaction sequence number.

The specific clusters using the Partition Cluster to transfer large frame shall subscribe to the registration table writing an entry for each frame to be partitioned with the Partition Cluster attributes fields specified in A.1.3.2. This entry shall be cancelled by the Partition Cluster when the frame is correctly transferred or the partitioning procedure exited with errors. The entries of this table should be inserted during the handshake phase, i.e. in the sender when the WriteHandshakeParam command is generated and in the receiver when the WriteHandshakeParam command is received.

The partitioned frames generated from the partitioning of a large frame shall use the ZCL transaction sequence number inserted in the ZCL header of the large frame for each small partitioned frame (transferred using the TransferPartitionedFrame commands) in order to identify the proper fragment if multiple partitions are running on the same endpoint with large frames carrying the same ClusterIDs.

**Table A.4 Registration Table of Clusters Using the Partition Cluster**

| ClusterID | Transaction Sequence Number | Partition Cluster Attributes |
|---|---|---|
| Registered cluster ID | Transaction sequence number (of the packet to be partitioned) through the Partition Cluster | Attributes that are written using the WriteHandshakeParam command |

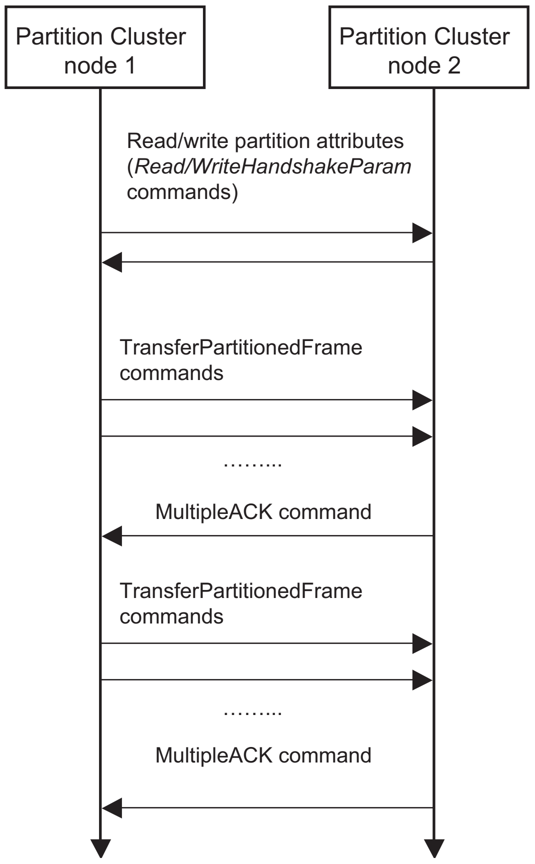

Partition Cluster node 1
Partition Cluster node 2
Read/write partition attributes (*Read/WriteHandshakeParam* commands)
TransferPartitionedFrame commands
……...
MultipleACK command
TransferPartitionedFrame commands
……...
MultipleACK command

**Figure A.12** Example of Partition Cluster Use

## A.2 Information Cluster

### A.2.1 Scope and Purpose

This document specifies a single cluster, the Information cluster, which provides commands and attributes for information delivery service on ZigBee networks and also specifies three types of special nodes on which this cluster works. One of the nodes, Information Node (IN) is a node which provides information contents in both pull-based and push-based information delivery to a mobile terminal. The contents may have links to other contents and thus they may be organized in a structure. Mobile Terminal (MT) is the one of special nodes and is used by an end-user who looks into information from the information node. The other node is Access Point (AP) which updates contents stored in Information nodes and has a role of gateway connected to the operator network. It is also assumed to be a

ZigBee coordinator which forms a network with Information nodes and Mobile Terminals. Access point may have a function of Information Node.

This document should be used in conjunction with the ZigBee Cluster Library, Foundation Specification [R4] which gives an overview of the library and specifies the frame formats and general commands used therein.

Information Delivery Service in this document is considered ‘Pull-based delivery’ and ‘Push-based delivery’. Both methods are provided by a single cluster, the information cluster. Figure A.14 shows typical usage of the cluster. This cluster may use Partition Cluster.

#### A.2.1.1 Data Structure of Contents Data

Typical data structure of contents data is as illustrated in Figure A.13. Each content data has its Content ID, which is used when the client cluster requests content to the server cluster.

A content data includes the ‘title strings’, ‘actual content’, ‘number of child contents’ and ‘children’s content IDs’.

To let each content data have its children content data makes it enables to organize list-structure or tree-structure and obtain a hierarchical content data structure, and a user who uses information delivery can request information along to child information links.

To obtain the first contents ID, there are methods, reading server attribute ‘Root ID’, using ID sent by out-of-band like via GPRS network and using ID provided by another telecom application clusters (ex. Payment or gaming).

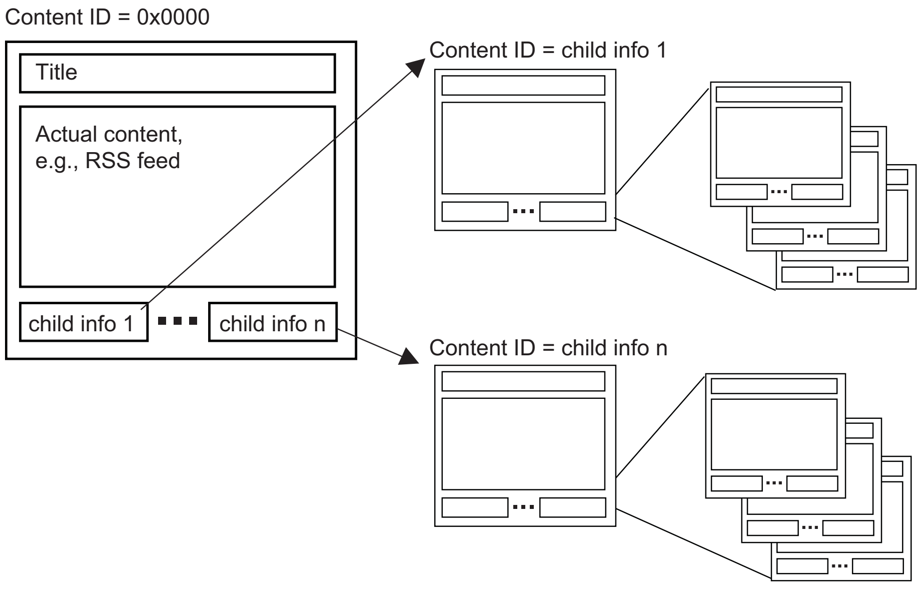

Content ID = 0x0000
Title
Actual content, e.g., RSS feed
child info 1
child info n
Content ID = child info 1
Content ID = child info n

**Figure A.13** Typical Content Data Structure

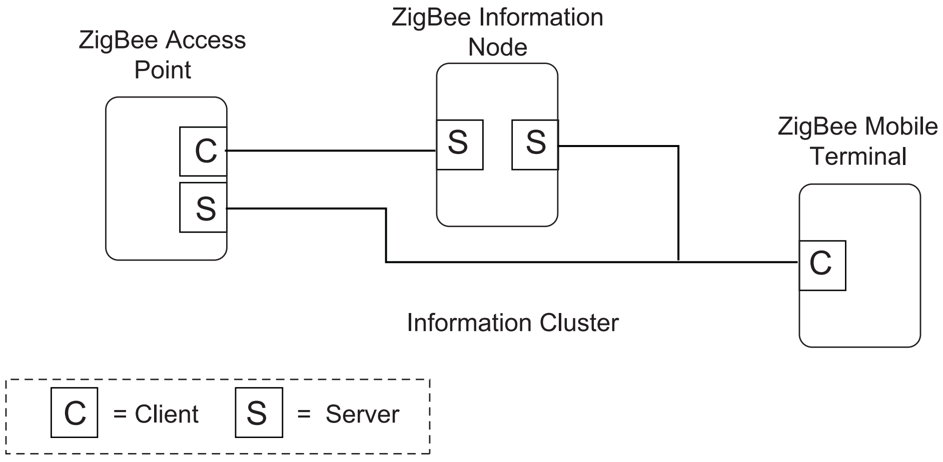
## A.2.2 Cluster List

A cluster specified in this document is listed in Table A.5.

Server cluster is expected to be implemented in the Information Node. Client cluster (including functions related to contents provisioning) is expected to be implemented in the Mobile Terminal. A part of commands of client cluster, update command and configuration commands are expected to be implemented in the Access Point. The Access Point may have functionality of the Information node and it has server cluster in that case. Some user specific content is provided with processing user-side information, defined as a preference. If a preference needs to be processed not in the Information Node but in an Access Point or in a server beyond the Access Point as a gateway, information indicates the Access Point's ID so that the Mobile Terminal can switch to access it (like illustrated in Figure A.15) or the Information Node acts as proxy and access the Access Point with client function to forward preference, commands to the Access Point and contents to the Mobile Terminal (like illustrated in Figure A.14).

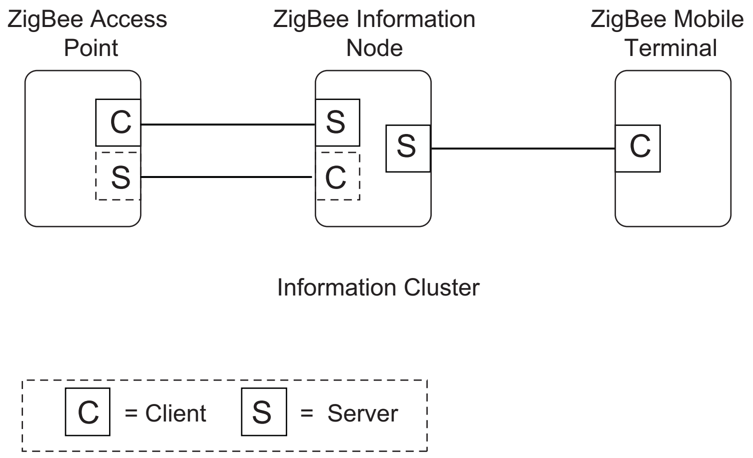
**Table A.5 Clusters Specified for Information Delivery**

| Cluster Name | Description |
|---|---|
| Information cluster | Attributes and commands for providing Information service to a ZigBee device. |



*Note: Device names are examples for illustration only*

**Figure A.14** Typical Usage of the Information Cluster



*Note: Device names are examples for illustration only*
*Note 2: Dashed boxes are for the cas IN works as proxy for MT*

**Figure A.15** Typical Usage of the Information Cluster, With Proxy Function

## A.2.3 Overview

This cluster provides attributes and commands for Information Delivery Service.

## A.2.4 Server

The Information Node (IN) has a server cluster which provides information delivery service. A client cluster in Mobile Terminal (MT) requests information and IN responds with requested contents on pull-based delivery. Besides, cluster can provides push-based delivery so the server cluster in the IN sends contents to client cluster in the MT (if properly configured).

Content may have links to the other contents. A link is called as child information in this document and it is represented as a ContentID. Contents can be organized in tree-structure.

Content may be one of three explicitly specified types: octet strings, character strings or RSS feed, so that the browser in the MT can understand easily what content it should access.

Cluster also provides such function that the client cluster in the AP can update contents and delete them in the IN.

Preference is used for carrying user-side information to let the IN provide user specific contents based on the user-side information. Contents may be modified along with that information on the IN. An example scenario of Information cluster is illustrated in Figure A.16.

A preference may be processed not in an IN but in an AP or in a server beyond the AP as a gateway. In that case, the IN needs to have client function to forward preference, commands and contents as proxy for MT (Forwarding scenario) or the IN needs to inform the MT to switch its access from the IN to the AP (Redirection scenario). The Cluster supports both scenarios. If the preference, commands and contents are forwarded by the IN between the MT and the AP, they may be just relayed transparently through the IN, or they may be processed by the IN. The IN may process the preference before forwarding it, and may process the stored preference together with the contents to create the customized contents after receiving the response from the AP, then sending the contents to the MT.

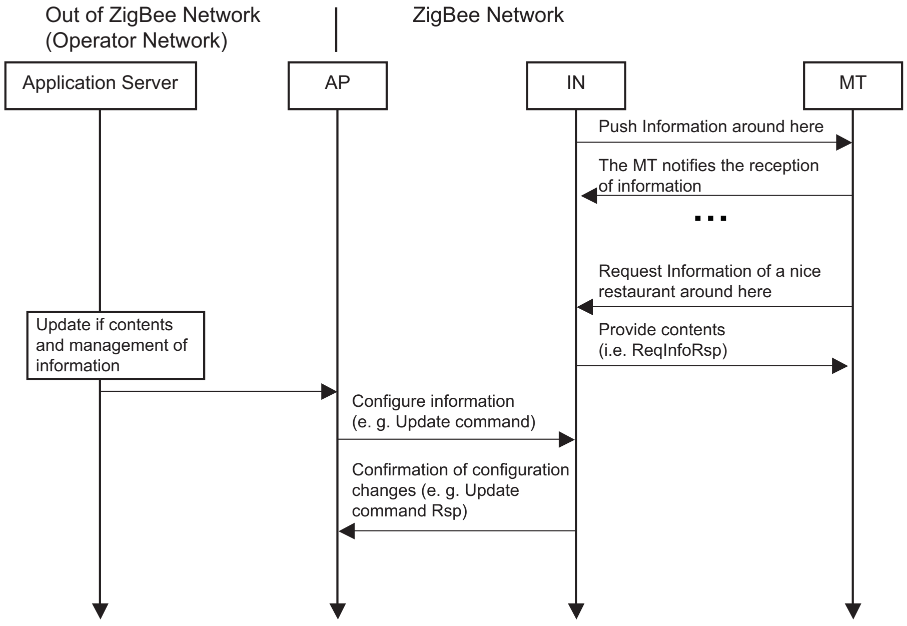

Out of ZigBee Network
(Operator Network)
ZigBee Network
Application Server
AP
IN
MT
Push Information around here
The MT notifies the reception of information
...
Request Information of a nice restaurant around here
Update if contents and management of information
Provide contents (i.e. ReqInfoRsp)
Configure information (e. g. Update command)
Confirmation of configuration changes (e. g. Update command Rsp)

**Figure A.16** An Example Sequence

Pull-based service is expected to work as follows:

**1** It provides decentralized contents distributed by Update command from the central node (the AP) (e.g. tree-structure contents distribution, specific permission to peep the contents to the authorized user).

**2** Advanced application program provides service in conjunction with the other functions like a Location cluster, or the preference data from the MT. (e.g. direction service based on the location information of user, push service matching individual attribute - invitation of a test drive of new car to men in thirties which hobby is driving or etc.)

**3** Hybrid service of the item 1 and 2.

The preference format is application dependent and is used by the service like the item b. Of course the application uses the preference shall have the ability to parse it. If the application doesn't have the ability, it shall report it that. The cluster provides a status code to inform it. For example, let's assume the IN provides service like item a. and the AP connected a network out of the ZigBee and an application server is deployed there. First the MT access to the IN to get general contents for the all of users. The IN can provide simple contents to the MT. Second, the MT request a content - good restaurant to the IN, which content is indicated to redirect to the AP. The MT switch its access to the AP. The AP requests preference to the MT to get user-side information, favorite food in the example. The AP sends the preference to the application server and it replies with the food restaurant which the user likes. Thus the AP can provide the content modified along with user-side information - the restaurant which provides he likes - to the MT. The example illustrated in Figure A.17.

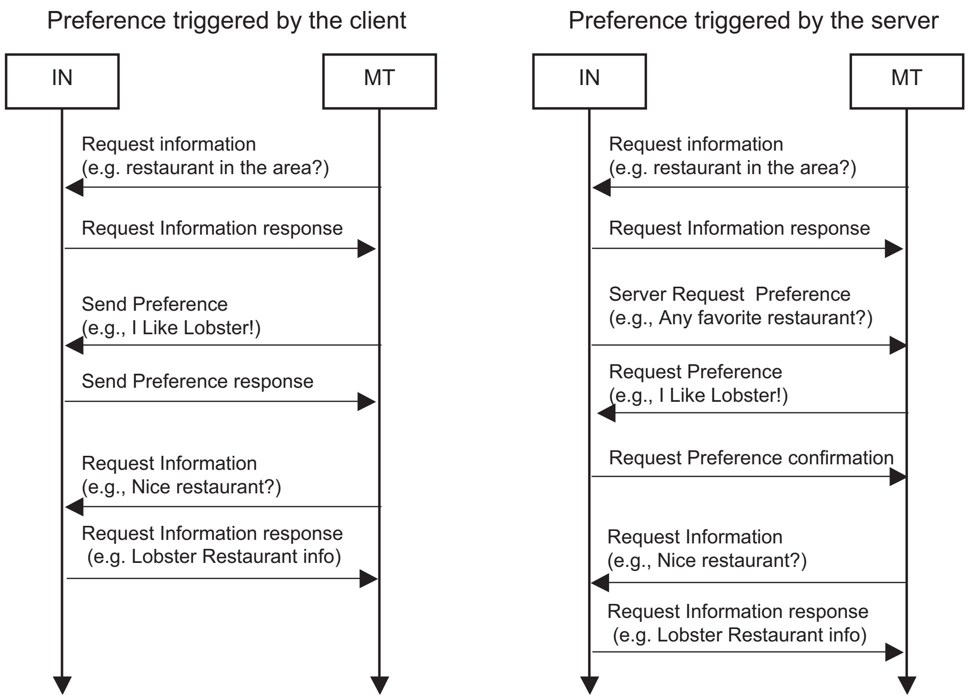

Preference triggered by the client
IN
MT
Request information
(e.g. restaurant in the area?)
Request Information response
Send Preference
(e.g., I Like Lobster!)
Send Preference response
Request Information
(e.g., Nice restaurant?)
Request Information response
(e.g. Lobster Restaurant info)
Preference triggered by the server
IN
MT
Request information
(e.g. restaurant in the area?)
Request Information response
Server Request Preference
(e.g., Any favorite restaurant?)
Request Preference
(e.g., I Like Lobster!)
Request Preference confirmation
Request Information
(e.g., Nice restaurant?)
Request Information response
(e.g. Lobster Restaurant info)

**Figure A.17** Preference Scenarios (Triggered by the Client or by the Server)

### A.2.4.1 Dependencies

None.

### A.2.4.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table A.6.

**Table A.6** ***Information Cluster*** **Attribute Sets**

| Attribute Set Identifier | Description |
|---|---|
| 0x000 | Node Information |
| 0x001 | Contents Information |
| 0x002 - 0xfff | Reserved |

### A.2.4.2.1 *Node Information* Attribute Set

The *Node Information* attribute set contains the attributes summarized in Table A.7.

**Table A.7** ***Node Information*** **Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0000 | *Node Description* | Character Strings | N/A | ReadOnly | M |
| 0x0001 | *Delivery Enable* | Boolean | 0x00 - 0x01 | ReadOnly | M |
| 0x0002 | *Push Information timer* | Unsigned 32-bit integer | N/A | ReadOnly | O |
| 0x0003 | *Enable secure configuration* | Boolean | 0x00 - 0x01 | ReadOnly | M |
| 0x0004 - 0x00ff | *Reserved* | N/A | N/A | N/A | N/A |

#### A.2.4.2.1.1 *Node Description* Attribute

This *Node Description* Attribute holds strings which indicate what Information Delivery service is available so that an end-user can select and distinguish this service.

#### A.2.4.2.1.2 *Delivery Enable* Attribute

The *Delivery Enable* attribute is Boolean and indicates whether the cluster is able to communicate with the other nodes. It is a read only attribute but it can be changed by using the Configure Delivery Enable command. If it is set to TRUE (0x01), the Information cluster is able to manage the following commands:

Request Information, Push Information Response, Send Preference and Request Preference Response.

#### A.2.4.2.1.3 *Push Information Timer* Attribute

The *Push Information Timer* is an Unsigned 32-bit integer and indicates whether the cluster is able to send Push Information command and the time between those commands. It is a read only attribute but it can be changed by using the Configure Push Information timer command. If this attribute is set to 0, then the automatic Push Information is disabled, otherwise the value is considered as an interval (in milliseconds) that elapses between Push Information commands. If this attribute is set to 0, it's still possible for the device to push information triggered by an event such as button being pushed.

#### A.2.4.2.1.4 *Enable Secure Configuration* Attribute

The *Enable Secure Configuration* attribute is a Boolean and indicates whether an application layer security is required in order to process the configuration commands: Update, Delete, Configure Delivery Enable, Configure Set Root ID, Configure Node Description, Configure Push Information timer. If this attribute is set to TRUE, then server side of the cluster need to use application link keys for processing those commands. If FALSE, then all the commands can be processed without using link keys.

### A.2.4.2.2 *Contents Information* Attribute Set

The *Node Information* attribute set contains the attributes summarized in Table A.8.

**Table A.8** ***Contents Information*** **Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory / Optional |
|---|---|---|---|---|---|
| 0x0010 | *NumberOfContents* | Unsigned 16-bit Integer | 0x0000-0xFFFF | ReadOnly | O |
| 0x0011 | *ContentRootID* | Unsigned 16-bit Integer | 0x0000-0xFFFF | ReadOnly | O |
| 0x0012 - 0x001f | *Reserved* | N/A | N/A | N/A | N/A |

#### A.2.4.2.2.1 *NumberOfContents* Attribute

This attribute holds the total number of contents which this server node has. It should reflect the result of updating command by AP. This attribute holds the total number of contents which this server node has. If the number is more than 0xffff, this attribute shall be set to 0xffff.

#### A.2.4.2.2.2 *ContentRootID* Attribute

This attribute holds root Content ID of octet strings, character string and RSSFeed Contents. *ContentRootID* is a start pointer so that user can access variety contents. If this attribute doesn't exist, there are no contents. 0xffff for this attribute means it is not specified yet.

## A.2.4.3 Commands Received

The received command IDs for the information cluster are listed in Table A.9. Please notice that at least one of the commands shall be implemented though they are defined as optional.

**Table A.9 Received Command IDs for the Information Cluster**

| Command Identifier Field Value | Description | Mandatory/ Optional | Command Type |
|---|---|---|---|
| 0x00 | Request Information | M | *Operation* |
| 0x01 | Push Information Response | M | *Operation* |
| 0x02 | Send Preference | O | *Operation* |
| 0x03 | Request Preference Response | O | *Operation* |
| 0x04 | Update | O | *Configuration* |
| 0x05 | Delete | O | *Configuration* |
| 0x06 | Configure Node Description | O | *Configuration* |
| 0x07 | Configure Delivery Enable | O | *Configuration* |
| 0x08 | Configure Push Information timer | O | *Configuration* |
| 0x09 | Configure Set Root ID | O | *Configuration* |
| 0x0a - 0xff | Reserved | N/A | N/A |

#### A.2.4.3.1 Request Information Command

This is a command requesting information as a list, as a content of text strings and as an RSS feed from mobile terminal to the Information Node or to the Access Point. An Information Node (or an Access Point) that receives this command shall reply by Request Information Response command with requested information to the sender of this command. It specifies how to indicate content by the Inquiry Type and also specifies what data type of content is requested by the Data Type ID. For example, in pull scenario, MT gets contents list, sending this command (e.g. Inquiry ID = ‘Request by depth’) and receiving Request Information Response Command with the list of titles. By another Request Information Command indicating contents ID, MT can get an individual content.

#### A.2.4.3.1.1 Frame Format

The Request Information command shall be formatted as illustrated in Figure A.18

| Octets: | Variable | 1 | 1 | Variable |
|---|---|---|---|---|
| Data Types | - | 8-bit enumeration | 8-bit bitmap | See A.2.4.3.1.2 |
| Field Name | ZCL Header | Inquiry ID | Data Type ID | Request Information Payload |

**Figure A.18** Payload Format of Request Information Command

Inquiry ID shall be set as one of IDs listed in Table A.10.

Data Type ID indicates what type of contents the response command requires. It shall be formatted by combination of bitmasks described in Table A.11. A bit for ‘Title’ indicates the request requires ‘Title strings’ and it can be combined other type of contents. Flagging the ‘Title’ bit indicates a request for a title to be attached and the other bits used for filter. If ‘Title’ bit, ‘Octet’ bit and ‘RSS’ bit are flagged, that means request is “Octet content attached title and RSS content attached title are required.” In the case that only the ‘Title’ bit is flagged, the request means “Just titles are required.” Please notice that all the contents shall maintain a title in the local database.

**Table A.10 Inquiry ID**

| Inquiry ID | Description | Optional/ Mandatory |
|---|---|---|
| 0x00 | Request a content by a content ID | M |
| 0x01 | Request contents by multiple IDs | O |
| 0x02 | Request all | O |
| 0x03 | Request by depth | O |
| 0x04 - 0xff | Reserved | - |

**Table A.11 Data Type IDs**

| Data Type ID | Bit Mask | Description |
|---|---|---|
| 0x01 | 0000 0001 | Title |
| 0x02 | 0000 0010 | Octet String |
| 0x04 | 0000 0100 | Character String |
| 0x08 | 0000 1000 | RSS Feed |
| 0x1X - 0xfX | - | Reserved |

### A.2.4.3.1.2 Request Information Payload

Request Information Payload changes along with Inquiry ID listed in Table A.10. Payload formats for each Inquiry ID are described following sections.

### A.2.4.3.1.3 Inquiry ID

#### A.2.4.3.1.3.1 Format for Request a Content by a Content ID

The command with this ID requests a single content by a Content ID. Format is illustrated in Figure A.19.

A server shall respond Request Information Response command with a content indicated by the content ID.

| Octets | 2 |
|---|---|
| Data Types | Unsigned 16-bit integer |
| Field Name | Content ID |

**Figure A.19** Payload Format for the Request a Content by a Content ID Command

#### A.2.4.3.1.3.2 Format for Request Contents by Multiple IDs

The command with this ID requests several contents by indicating several content IDs. It shall be formatted as illustrated in Figure A.2.

A server shall respond Request Information Response command with contents indicated by content IDs.

| 2 | 2 | … | 2 |
|---|---|---|---|
| Content ID 1 | Content ID 2 | … | Content ID n |

**Figure A.20** Request Information Payload for the Request Contents by Multiple IDs Command

### A.2.4.3.1.3.3 Format for Request All

The command with this ID requests all contents. No payload format is specified and it should be empty.

A server shall respond Request Information Response command with all contents.

### A.2.4.3.1.3.4 Format for Request by Depth

Upon receipt of the command with this ID, server shall reply Request Information Response command with concatenated contents indicated by Start ID and Depth. Request Information Payload format for this ID is specified in Figure A.21.

Start ID field holds content ID for starting point to retrieve structured contents.

Depth field holds how many levels to request from Start ID tracing child information. If a depth equals to 0x00, the requested content should be single content of Start ID itself.

Server shall provide concatenated contents, which needs a prevention of duplication induced by the loop of links. (For example, if the content has a child content which child ID refers its parent (A → B, B → A), there is a loop. If the requester indicates 2 for the depth and requests content "A", searching child information would be like as A → B → A. However only content A and B should be carried in this case).

| **Octets:** | **2** | **1** |
|---|---|---|
| Data Types | Unsigned 16-bit integer | Unsigned 8-bit integer |
| Field Name | Start ID | Depth |

**Figure A.21** Request Information Payload for the Request by Depth Command

### A.2.4.3.2 Push Information Response Command

This command is used by the client to notify the reception of the data carried by the Push Information command, and it is used by the server to confirm if it is correctly stored or not into MT. This command shall not be used if the Push Information Command is sent by broadcast. It is to prevent explosion of response.

Payload format shall be as illustrated in Figure A.22.

| **Octet: Variable** | **2** | **1** | **…** | **2** | **1** |
|---|---|---|---|---|---|
| ZCL Header | Notification 1 | | … | Notification n | |
| | Content ID 1 | Status Feedback 1 | | Content ID n | Status Feedback n |

**Figure A.22** Payload Format of the Push Information Response Command

Notification field has two sub-fields, content ID and Status Feedback. Content ID indicates what content the notification has the status for. Status Feedback indicates the status of the reception of the content.

Possible message for Status Feedback are SUCCESS, FAILURE, MALFORMED_COMMAND, UNSUP_CLUSTER_COMMAND, INVALID_FIELD, INSUFFICIENT_SPACE, HARDWARE_FAILURE and SOFTWARE_FAILURE already specified in the enumeration lists of ZCL [R4].

### A.2.4.3.3 Send Preference Command

This command carries a preference, that is specific information of interest for the user, from the client to the server. Upon receipt of this command on the server, the server application may modify or change user specific contents along with preference information. The type of data put into the preference is based on the Preference Type field. Payload format for this command shall be as illustrated in Figure A.23.

| **Octet: 3** | **2** | **Variable** |
|---|---|---|
| ZCL Header | Preference Type | Preference Payload, see Table A.12 |

**Figure A.23** Payload Format for the Send Preference Command

The Preference Type determines the format of the preference Payload. All devices must support Preference Type of 0x0000.

**Table A.12 Preference Type**

| Preference Type | Description |
|---|---|
| 0x0000 | Preference is Multiple Content ID |
| 0x0001 | Preference is multiple octet strings |
| 0x0002 - 0x7fff | Reserved |
| 0x8000 - 0xfffb | Used for vendor specific format |
| 0xfffc - 0xffff | Reserved |

| Octets | 1 | 2 | … | 2 |
|---|---|---|---|---|
| Data Types | Unsigned 8-bit integer | Unsigned 16-bit integer | … | Unsigned 16-bit integer |
| Field Name | Count | Content ID 1 | | Content ID N (based on count) |

**Figure A.24** Payload Format for the Preference is Multiple Content ID Command (0x0000)

| Octets | 1 | Variable (1-256) | … | Variable (1-256) |
|---|---|---|---|---|
| Data Types | Unsigned 8-bit integer | Octet String (ZCL data type 0x41) | … | Octet String (ZCL data type 0x41) |
| Field Name | Count | Preference Data 1 | | Preference Data N (based on count) |

**Figure A.25** Payload Format for the Preference is Multiple Octet Strings Command (0x0001)

As described in Figure A.17 there are two scenarios for the preference:

**1** Preference triggered by server side (Information Node or Access Point):

- IN ← (Request Information) ← MT
- IN → (Request Information Response) → MT
- IN → (Server Request Preference) → MT
- IN ← (Request Preference Response) ← MT
- IN → (Request Preference Confirmation) → MT
- IN ← (Request Information) ← MT
- IN → (Request Information Response) → MT

**2** Preference triggered by client side (e.g. Mobile terminal):

- IN ← (Request Information) ← MT
- IN → (Request Information Response) → MT
- IN ← (Send Preference) ← MT
- IN → (Send Preference Response) → MT
- IN ← (Request Information) ← MT
- IN → (Request Information Response) → MT

#### A.2.4.3.4 Request Preference Response Command

This command carries a preference as a response of Server Request Preference command on pull-basis. Format shall be as illustrated in Figure A.26.

| Octet: 3 | 1 | 2 | Variable |
|---|---|---|---|
| ZCL Header | Status Feedback | Preference Type | Preference Payload, see Table A.12 |

**Figure A.26** Payload Format of the Request Preference Response Command

Status Feedback carries a message as a response to previous Server Request Preference command. Possible messages are SUCCESS, FAILURE, NOT_FOUND, MALFORMED_COMMAND, UNSUP_CLUSTER_COMMAND, INVALID_FIELD, HARDWARE_FAILURE and SOFTWARE_FAILURE already specified in enumeration lists of ZCL [R1]. Besides, REQUEST_DENIED is included to these messages for this cluster specification.

#### A.2.4.3.5 Update Command

Server cluster in the IN which receives this command from the AP shall updates contents by the one which the command carried except that there is an error in the IN. Update command also indicates various control to the contents by the control fields. Control fields affect to all of contents carried by the Update command, so contents required to be indicated different control should be carried by another Update command.

Payload format is as illustrated in Figure A.27.

| Octet | Variable | 1 | 1 | Variable |
|---|---|---|---|---|
| Data Type | Variable | 8-bit enumeration | 8-bit bitmap | Payload Format for 'Multiple Content' |
| Field Name | ZCL Header | Access Control Field | Option Field | Contents Data |

**Figure A.27** Payload Format for the Update Command

### A.2.4.3.5.1 Access Control Field

The Access Control Field is an 8-bit enumeration and is used to indicate security level for the validation to access the contents which are carried by the Update command. All of contents carried by the Update command shall be affected by this control field. The enumeration values are listed up in Figure A.28.

- **Free to access:** All of the clients are permitted to access the contents without special validation.
- **Link key establishment based:** The client to access to the IN shall be required to establish link key establishment to achieve the contents. Contents shall be encrypted by the link key.
- **Billing based:** The client to access the contents is required to finish the Billing cluster procedure.
- **Vendor-specific:** No special method is defined in this document. The application defines it (out-of-box, out-of-band, etc.)

| Access Control Mode Value | Description |
|---|---|
| 0x00 | Free to access |
| 0x01 | Link key establishment based |
| 0x02 | Billing based |
| 0x03 - 0xfe | Reserved |
| 0xff | Vendor Specific |

**Figure A.28** Value of the Access Control Field

### A.2.4.3.5.2 Option Field

Option Field is used for advanced indication while updating contents. Forwarding flag, Redirection flag, Overwrite update flag are defined in the current version. The 'Forwarding' flag or the 'Redirection' flag are used to indicate 'content' so that the commands of request and response related to the indicated 'content' shall be forwarded or redirected to the AP. If both 'Forward' flag and 'Redirection' flag are 1, the server cluster shall reply the INVALID_FIELD by the Update Response command.

The format is as illustrated in Figure A.29.

| Bits: 1 | 1 | 1 | 5 |
|---|---|---|---|
| Forward | Redirection | Overwrite update | Reserved |

**Figure A.29** Format for the Redirection Control Field

**Forward flag:** The Information Node is required to forward messages from the MT to the AP and message from the AP to the MT with acting as proxy. All the requests from the MT for the contents updated with this flag are forwarded to the AP. A Preference from the MT is also sent to AP if the IN has it. The AP answers the response command to the IN with requested contents and they are forwarded to the MT similarly. Figure A.30 shows an example usage of forwarding.

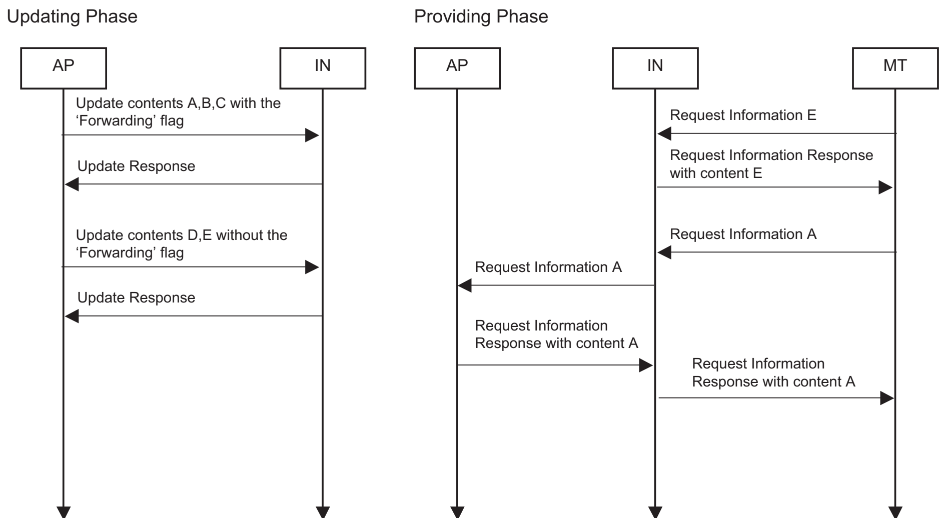

Updating Phase
Providing Phase
AP
IN
AP
IN
MT
Update contents A,B,C with the 'Forwarding' flag
Update Response
Update contents D,E without the 'Forwarding' flag
Update Response
Request Information E
Request Information Response with content E
Request Information A
Request Information A
Request Information Response with content A
Request Information Response with content A

**Figure A.30** An Example Sequence of Forwarding Case

**Redirection flag:** A client requested the contents indicated by this flag shall receive Request Information Response command with Status feedback 'INDICATION_REDIRECTION_TO_AP'. The client is required to switch to access from the Information Node to the Access Point. This flag makes MT enable to switch access to AP automatically without user's operation (Like that user access the child content). For example, let IN have general site-dependent information and content depends on user-side information generated by a server beyond the operator network which AP is connected. Figure A.31 shows an example usage of forwarding.

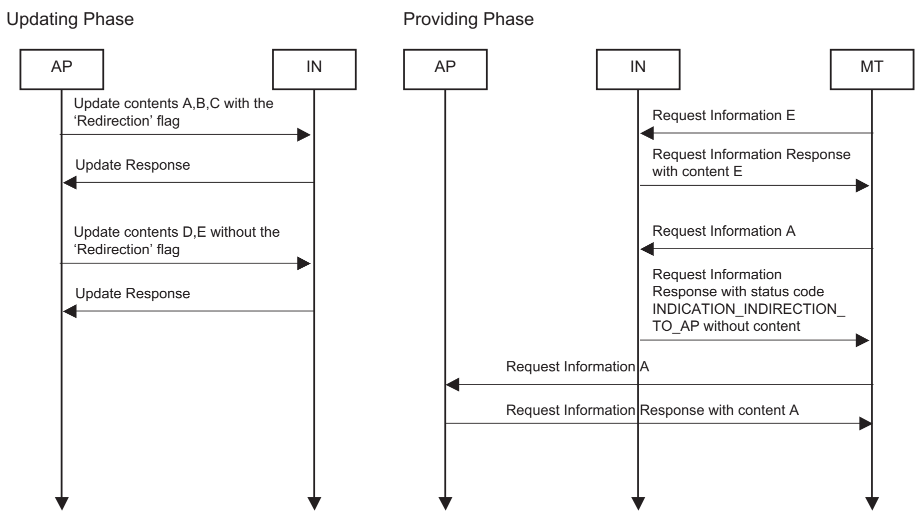

Updating Phase
Providing Phase
AP
IN
AP
IN
MT
Update contents A,B,C with the 'Redirection' flag
Update Response
Update contents D,E without the 'Redirection' flag
Update Response
Request Information E
Request Information Response with content E
Request Information A
Request Information Response with status code INDICATION_INDIRECTION_TO_AP without content
Request Information A
Request Information Response with content A

**Figure A.31** An Example Sequence of Redirecting Case

**Overwrite:** For the case that the IN has already contents corresponding to the one required to Update, the command indicates if it can be overwritten or not. If it is 0b1, the contents carried by the Update command overwrites the contents which the IN has. If it is 0b0, overwriting is not permitted. In that case, the error 'FAILURE' on Update Response command is issued by the IN and the command is ignored if the IN has corresponding contents.

### A.2.4.3.6 Delete Command

Server cluster in the IN which receives this command from the AP shall delete contents by the one which the command carried except that there is an error in the IN. Delete command also indicates various control to the contents by the control fields. Control fields affect to all of contents carried by the Delete command, so contents required to be indicated different control should be carried by another Delete command.

Payload format is as illustrated in Figure A.32.

| Octet | Variable | 1 | 2 | … | 2 |
|---|---|---|---|---|---|
| Data Type | Variable | 8-bit bitmap | Unsigned 16-bit Integer | … | Unsigned 16-bit Integer |
| Field Name | ZCL Header | Deletion Option | ContentID 1 | … | ContentID n |

**Figure A.32** Payload Format for the Delete Command

#### A.2.4.3.6.1 Deletion Option Field

Deletion Option field is used to enables various deletion functions. The format is as illustrated in Figure A.33

| Bits: 1 | 2-8 |
|---|---|
| Recursive | Reserved |

**Figure A.33** Format for the Deletion Option Field

**Recursive:** If it is 0b1, all the sub tree starting for thee content carried by the Delete command are deleted. If it is 0b0, only the content carried by the Delete command is deleted. How the children are linked to the rest of the tree is out of scope of this document, it's application dependent.

### A.2.4.3.7 Configure Node Description[12]

Payload format for the Configure Node Description command shall be as illustrated in Figure A.34.

Upon recipient of this command, the server cluster shall change its *Node Description* attribute to the value of the "Description" field in this command. The use of this specific command guarantees that *Node Description* attribute can be reconfigured only when *Delivery Enable* attribute is set to TRUE. Upon reception of this command the recipient will reply with Default Response with status field equal to SUCCESS (if the requester set the Disable Default response bit of ZCL header to 0). The Configure Node Description command will be acknowledged with Default Response with status field equal to NOT_AUTHORIZED in case the recipient entity requires a secure link for configuration (i.e. *Enable Secure Configuration* attribute set to TRUE) while the sender didn't use the proper link key for sending the configuration commands.

| Octets: Variable | Variable |
|---|---|
| Data Type | Character string |
| ZCL header | Description |

**Figure A.34** Payload Format for the Configure Node Description Command

### A.2.4.3.8 Configure Delivery Enable

Payload format for the Configure Delivery Enable command shall be as illustrated in Figure A.35.

Upon recipient of this command, the server side of the Information cluster should set the value present in the Enable Flag field into the *Delivery Enable* attribute.

12. CCB #1115

Note that if *Enable Secure Configuration* attribute is set to TRUE (0x01 as for ZCL definition of Boolean in 075123r02), this command should be handled only whether the entity sending this message will have previously set up its link key with the recipient entity.

If the ‘Enable flag’ is set to FALSE (0x00), the server cluster shall stop the information delivery service. However the device supporting this server cluster (i.e. Information node) shall still accept those commands needed to configure the node: Update, Delete, Configure Node Description, Configure Delivery Enable, Configure Push Information timer and Configure Set Root ID.

All cluster-specific commands aside from “configuration” commands will be replied by their respective responses with status code REQUEST_DENIED if the *Delivery Enable* attribute is disabled.

The Configure Delivery Enable command will be acknowledged with Default Response with status field equal to NOT_AUTHORIZED in case the recipient entity requires a secure link for configuration (i.e. *Enable Secure Configuration* attribute set to TRUE) while the sender didn’t use the proper link key for sending the configuration commands.’

| **Octets: Variable** | **1** |
|---|---|
| ZCL header | Enable flag |

**Figure A.35** Payload Format for the Configure Delivery Enable Command

#### A.2.4.3.9 Configure Push Information Timer

Payload format for the Configure Push Information Timer command shall be as illustrated in Figure A.36.

Upon recipient of this command, the server side of the Information cluster shall change its *Push Information Timer* attribute to the value of the Timer field carried in this command.

The Configure Push Information Timer command will be acknowledged with Default Response with status field equal to NOT_AUTHORIZED in case the recipient entity requires a secure link for configuration (i.e. *Enable Secure Configuration* attribute set to TRUE) while the sender didn’t use the proper link key for sending the configuration commands.

| **Octets: Variable** | **4** |
|---|---|
| ZCL header | Timer |

**Figure A.36** Payload Format for the Configure Push Information Timer Command

#### A.2.4.3.10 Configure Set Root ID

Payload format for the Configuration Set Root ID command shall be as illustrated in Figure A.37.

Upon receipt of this command, the server side of Information cluster shall change its *Root ID* attribute to the value of the Root ID field in this command.

The Configure Set Root ID command will be acknowledged with Default Response with status field equal to NOT_AUTHORIZED in case the recipient entity requires a secure link for configuration (i.e. *Enable Secure Configuration* attribute set to TRUE) while the sender didn't use the proper link key for sending the configuration commands.

| **Octets: Variable** | **2** |
|---|---|
| ZCL header | Root ID |

**Figure A.37** Payload Format for the Configure Set Root ID Command

### A.2.4.4 Commands Generated

The generated command IDs for the Information cluster are listed in Table A.13. Please notice that at least one of the following commands shall be implemented.

**Table A.13 Generated Command IDs for the Information Cluster**

| **Command Identifier Field Value** | **Description** | **Mandatory / Optional** | **Command Type** |
|---|---|---|---|
| 0x00 | Request Information Response | M | Operation |
| 0x01 | Push Information | M | Operation |
| 0x02 | Send Preference Response | O | Operation |
| 0x03 | Server Request Preference | O | Operation |
| 0x04 | Request Preference Confirmation | O | Operation |
| 0x05 | Update Response | O | Configuration |
| 0x06 | Delete Response | O | Configuration |
| 0x06 - 0xff | Reserved | N/A | N/A |

#### A.2.4.4.1 Request Information Response Command

This command is a response command according to a Request Information command which a client requests and carries requested information or carries status feedback if error occurs. Payload format for this command shall be as illustrated in Figure A.38.

| Octet: Variable | 1 | 1 | Variable | … | 1 | Variable |
|---|---|---|---|---|---|---|
| ZCL Header | Number | Status Feedback 1 | Single Content or ContentID | … | Status Feedback n | Single Content or ContentID |

**Figure A.38** Payload Format of the Request Information Response Command

Number indicates how many single contents are carried by this command. Pairs of 'Status Feedback' and 'Single Content' appear corresponding to this number.

Single content is the actual content which format is specified in sub-clause A.2.6.1.

Status Feedback carries a message as a response to the previous Request Information command sent by the client. Possible messages are SUCCESS, FAILURE, NOT_FOUND, MALFORMED_COMMAND, UNSUP_CLUSTER_COMMAND, INVALID_FIELD, HARDWARE_FAILURE and SOFTWARE_FAILURE already specified in enumeration lists of ZCL [R1] Besides, INDICATION_REDIRECTION_TO_AP, REQUEST_DENIED, PREFERENCE_IGNORED and MULTIPLE_REQUEST_NOT_ALLOWED are included to these messages for this cluster specification. All the Status enumerations are listed up in Table A.14.

If a content is not available due to some reason (an error in many cases), the Single Content field should be replaced by the "Content ID" in order to report which content requested has an error. (i.e. all the status codes except SUCCESS and PREFERENCE_IGNORED).

#### A.2.4.4.2 Push Information Command

This is a command putting information especially from an information node (or an access point) to a mobile terminal on push basis. A content sent to mobile terminal (e.g. a list of contents, a content described in octets strings, character strings, a title of content or RSS feed) is carried by this command.

| Octet: Variable | Variable |
|---|---|
| ZCL Header | Contents Data |

**Figure A.39** Payload Format of the Push Information Command

Format for Contents Data shall be as described in sub-clause A.2.6.

#### A.2.4.4.3 Send Preference Response Command

This command is used by the server to notify whether the data carried by Send Preference command generated by the client is accepted correctly or not.

Payload format shall be as illustrated in Figure A.40.

Status Feedback carries a message as a response to the previous command Send Preference command from the client. Possible values are: SUCCESS, ZCL_PREFERENCE_DENIED, ZCL_PREFERENCE_IGNORED. If all the Preference Data are correctly processed, it is enough to respond with a unique Status Feedback equals to SUCCESS.

Also, if the server device does not support the Preference Type carried by the command, a unique Status Feedback value will be set to ZCL_PREFERENCE_IGNORED (0x74).

| **Octet: 3** | **1** | | **1** |
|---|---|---|---|
| ZCL Header | Status Feedback 1 | … | Status Feedback n |

**Figure A.40** Payload Format for the Send Preference Response Command and the Request Preference Confirmation Command

### A.2.4.4.4 Server Request Preference Command

This command requests a Preference as user-side information in the MT on pull-basis.

Upon receipt of this command at client cluster in the MT, the client is required to respond by Request Preference Response command.

This command needs no payload format specified but just ZCL header as illustrated in Figure A.41.

| **Octets: Variable** |
|---|
| ZCL Header |

**Figure A.41** Payload Format of the Server Request Preference Command

### A.2.4.4.5 Request Preference Confirmation Command

This command is used by the server to notify whether the data carried by Request Preference Response command generated by the client is accepted correctly or not.

Payload format shall be as illustrated in Figure A.40.

Status Feedback carries a message as a response to the previous command Request Preference Response command from the client. Possible values are: SUCCESS, ZCL_PREFERENCE_DENIED, ZCL_PREFERENCE_IGNORED. If all the Preference Data are correctly processed, it is enough to respond with a unique Status Feedback equals to SUCCESS.

Also, if the server device does not support the Preference Type carried by the command, a unique Status Feedback value will be set to ZCL_PREFERENCE_IGNORED (0x74).

#### A.2.4.4.6 Update Response Command

This command is used to notify any result of an Update command received by IN.

Payload format for this command shall be as illustrated in Figure A.42.

Notification field has two sub-fields, content ID and Status Feedback. Content ID indicates what content the notification has the status for. Status Feedback indicates the status of the reception of the content.

| **Octet: Variable** | **2** | **1** | **…** | **2** | **1** |
|---|---|---|---|---|---|
| ZCL Header | Notification 1 | | … | Notification n | |
| | Content ID 1 | Status Feedback 1 | | Content ID n | Status Feedback n |

**Figure A.42** Payload Format of the Update Response and Delete Response Command

Status Feedback carries a message as a response to the previous command Update command from the client. Possible messages are SUCCESS, FAILURE, MALFORMED_COMMAND, UNSUP_CLUSTER_COMMAND, INVALID_FIELD, INSUFFICIENT_SPACE, HARDWARE_FAILURE and SOFTWARE_FAILURE already specified in enumeration lists of ZCL [R1]. Besides, REQUEST_DENIED is included into these messages for this cluster specification. All the status enumerations are listed up in Table A.14.

#### A.2.4.4.7 Delete Response Command

This command is used to notify any result of a Delete command received by IN.

Payload format for this command shall be as illustrated in Figure A.42.

Notification field has two sub-fields, content ID and Status Feedback. Content ID indicates what content the notification has the status for. Status Feedback indicates the status of the reception of the content.

## A.2.5 Client

### A.2.5.1 Commands Received

The client receives the cluster-specific commands detailed in sub-clause A.2.4.4

### A.2.5.2 Commands Generated

The client generates the cluster-specific commands detailed in A.2.4.3, as required by application.

## A.2.6 Payload Formats for Contents Data

This section describes about payload format for contents data as used in the commands defied for the Information cluster.

### A.2.6.1 Payload Format for Multiple Contents

Payload format for the contents shall be as illustrated in Figure A.43

| **Octet** | **1** | **Variable** | **…** | **Variable** |
|---|---|---|---|---|
| Data Type | Unsigned 8-bit Integer | Format for Single Content (defined in this section) | | Format for Single Content (defined in this section) |
| Field Name | Number | Single Content 1 | … | Single Content n |

**Figure A.43** Payload Format for Multiple Contents

Number field holds a number of single contents. The payload format for the single content is specified in Figure A.44.

| **Octet** | **2** | **1** | **Variable** | **Variable** | **1** | **2/0** | **…** | **2/0** |
|---|---|---|---|---|---|---|---|---|
| Data type | Unsigne d 16-bit Integer | 8-bit Bitmap | Long Character String (defined in this cluster section) | Payload Format for Content (defined in this cluster section) | Unsigned 8-bit Integer | Unsigne d 16-bit Integer | …. | Unsigne d 16-bit Integer |
| Field Name | Content ID | Data Type ID | Title String | Content String | Number of children | Content ID 1 | … | Content ID n |

**Figure A.44** Format for Single Content

Content ID corresponds to the content; There is no rule provided by this document. It is expected to be defined by the service provider.

Data Type indicates the supported data types of content (it could be title and/or long octet, long character string or RSS). If a combination of type is supported by a Single Content, the order of data types shall be the one described in Figure A.43. If a bit field of Title in Data Type ID is 0b1, Title String field will be inserted in the Single Content frame. If another bit than the Title field is 0b1, Content Strings field and following fields appear.

Title String appears in the Single Content frame only if a Title bit in Data Type ID field is flagged. It represents title string in 'character string' data type; 'long character string' data type already includes 2 bytes count field.

Content String holds actual content data described in data type. It is inserted in the frame only if another bit than the Title field is 0b1 in the Data Type ID field.

Number of Children indicates how many links to child-contents this content has. If there is no child for this content this field shall be set to 0.

Content ID n holds List of child-contents ID.

## A.2.6.2 Contents Data Types

### A.2.6.2.1 Title String

| Octet: 2 | Variable |
|---|---|
| Count | Title |

**Figure A.45** Format for Title String

### A.2.6.2.2 Long Octet Strings

Extended count field to two bytes. Count represents how many octets the Octet Data's length is.

| Octet: 2 | Variable |
|---|---|
| Count | Octet Data |

**Figure A.46** Format for Long Octet Strings

### A.2.6.2.3 Long Character Strings

Extended count field to two bytes. Count represents how many characters the Character Data's length is. It should not be in Bytes if the character set is not 8-bit code (e.g. 2-bytes code).

| Octet: 2 | Variable |
|---|---|
| Count | Character Data |

**Figure A.47** Format for Long Character Strings

#### A.2.6.2.4 RSS Feed

Length field represents length in bytes not in character count. What character set is used should be defined in RSS feed data. In many cases, it would be ASCII compatible coding - like a UTF-8.

| Octet: 2 | Variable |
|---|---|
| Length | RSS Feed Data |

**Figure A.48** Format for RSS Feed

### A.2.6.3 Status Codes for the Information Cluster

Where an information cluster command contains a status field, the actual value of the enumerated status values are listed in Table A.14. Because this table copied from ZCL status code enumeration and is inserted the Information cluster-specific status codes at the start point, it may differ from the original if ZCL is updated. However, there is no problem because this table is used only for information cluster and is only used by its specific commands.

**Table A.14 Enumerated Status Values Used in the ZCL**

| Enumerated Status | Value | Description |
|---|---|---|
| SUCCESS | 0x00 | Operation was successful |
| FAILURE | 0x01 | Operation was not successful |
| - | 0x02 - 0x6f | Reserved |
| REQUEST_DENIED | 0x70 | Request was denied due to lack of permission |
| MULTIPLE_REQUEST_NOT_ ALLOWED | 0x71 | Request of multiple contents is not supported |
| INDICATION_REDIRECTION _TO_AP | 0x72 | Server Indicates to change the access to the AP for this content |
| PREFERENCE_DENIED | 0x73 | The preference was not accepted. Because it was not understandable or invalid |
| PREFERENCE_IGNORED | 0x74 | Inform that preference was not used to modify the content which is provided by this command |
| MALFORMED_COMMAND | 0x80 | The command appears to contain the wrong fields, as detected either by the presence of one or more invalid field entries or by there being missing fields. Command not carried out. Implementer has discretion as to whether to return this error or INVALID_FIELD |

**Table A.14 Enumerated Status Values Used in the ZCL (Continued)**

| Enumerated Status | Value | Description |
|---|---|---|
| UNSUP_CLUSTER_COMMAND | 0x81 | The specified general ZCL command is not supported on the device. Command not carried out |
| INVALID_FIELD | 0x85 | At least one field of the command contains an incorrect value, according to the specification the device is implemented to. Command not carried out |
| INSUFFICIENT_SPACE | 0x89 | An attempt to create an entry in a table failed due to an insufficient amount of free space available |
| NOT_FOUND | 0x8b | The requested information (e.g. table entry) could not be found |
| - | 0x8d - 0xbf | Reserved |
| HARDWARE_FAILURE | 0xc0 | An operation was unsuccessful due to a hardware failure |
| SOFTWARE_FAILURE | 0xc1 | An operation was unsuccessful due to a software failure |
| - | 0xc3 - 0xff | Reserved |

## A.3 Billing Cluster

### A.3.1 Scope and Purpose

This document specifies a single cluster, the Billing cluster, which provides commands and attributes for enabling billing for ZigBee devices. This cluster is to provide a standardized interface for the devices to charge ZigBee users for services in different application scenarios (e.g. Telecom Applications services) that requires billing for provided services.

This document should be used in conjunction with the ZigBee Cluster Library, Foundation Specification (see [R4]), which gives an overview of the library and specifies the frame formats and general commands used therein.

The cluster specified in this document is typically used for telecom applications, but may be used in any other application domains. This cluster may use Partition Cluster.

## A.3.2 Cluster List

The Billing cluster provides the attributes and commands required for enabling billing over a ZigBee network.

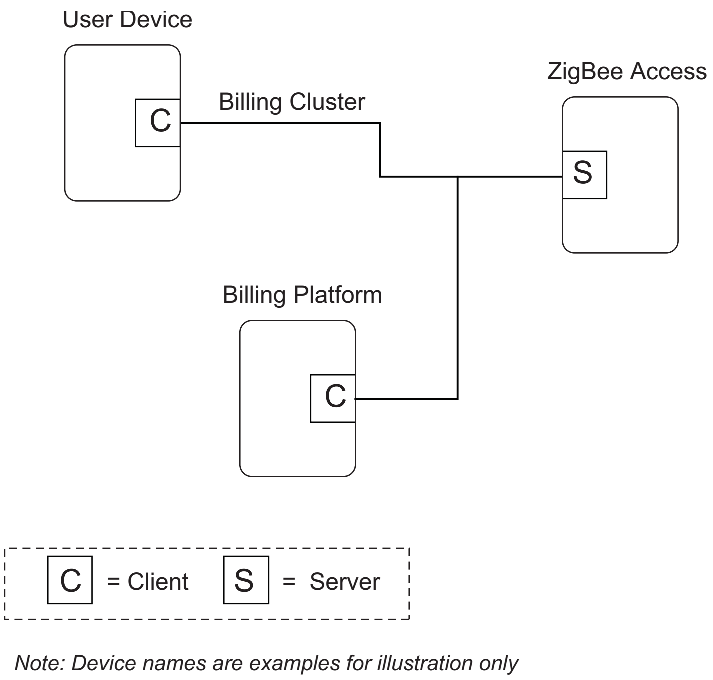


**Figure A.49** Typical Usage of the Payment Cluster

The User Device may be a ZSIM or a Mobile Terminal or any other devices used for services requiring billing, whereas the Billing Platform may be an Access Point, Information node, Service Provider or Billing unit.

## A.3.3 Overview

This cluster provides attributes and commands to enable billing of users for provided services through the use of a billing platform.

Please notice that the optional attributes or commands could be stated as mandatory inside the device description of a particular profile specification.

## A.3.4 Server

### A.3.4.1 Dependencies

This cluster may require the use of Key-establishment cluster.

### A.3.4.2 Attributes Sets

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table A.15.

**Table A.15 Billing Cluster Attributes Sets**

| Attribute Set Identifier | Description |
|---|---|
| 0x000 | User Information |
| 0x001 | Service Information |
| 0x002 | Bill Record |
| 0x003 - 0xfff | Reserved |

### A.3.4.3 *User Information* Attribute Set

The *User Information* attribute set contains the attributes summarized in Table A.16.

**Table A.16 Attributes of the *User Information* Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0000 | *UserID* | Octet string (should be aligned with UserID of other clusters) | | ReadOnly | M |
| 0x0001- 0x000f | Reserved | | | | |

#### A.3.4.3.1 *UserID* Attribute

The *UserID* attribute specifies a unique identification code for the user who has used a specific service.

### A.3.4.4 *Service Information* Attribute Set

The *Service Information* attribute set contains the attributes summarized in Table A.17.

**Table A.17 Attributes of the *Service Information* Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0010 | *ServiceID* | Unsigned 16-bit integer | 0x0000-0xffff | ReadOnly | M |
| 0x0011 | *ServiceProviderID* | Unsigned 16-bit integer | 0x0000-0xffff | ReadOnly | M |
| 0x0012 | *SessionInterval* | Unsigned 16-bit integer | 0x0000-0xffff | ReadOnly | M |
| 0x0013-0x001f | *Reserved* | | | | |

#### A.3.4.4.1 *ServiceID* Attribute

The *ServiceID* attribute contains the unique identifier of the service (e.g. airport scheduling, stock quotations, movie, voice conversation, etc). This attribute is defined by the specific application scenario considered for the billing.

#### A.3.4.4.2 *ServiceProviderID* Attribute

The *ServiceProviderID* attribute indicates the unique identifier of the Service Provider (e.g. Bank).

#### A.3.4.4.3 *SessionInterval* Attribute

The *SessionInterval* attribute indicates the period of transmission of Billing Record information. This value will be continuously decreased after the reception of a Start Service Session command until reached zero or until restored to the initial value after reception of Session Keep Alive command in order to maintain the session up. If *SessionInterval* attribute goes to zero the device that performs the calculation will notify the Billing platform with a Billing Record and, if a Bill Status Notification with Not Allowed status is received from the billing platform for certain user, it will forbid the billed device the access to the service for that user.

### A.3.4.5 Bill Record Set

The *Bill Record* attribute set contains the attributes summarized in Table A.18.

**Table A.18 Attributes of the *Bill Record* Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory / Optional |
|---|---|---|---|---|---|
| 0x0020 | *Timestamp* | 15 octets string (ISO8601) or equivalent | - | ReadOnly | M |
| 0x0021 | *Duration* | Unsigned 16-bit integer | - | Read/ Write | O |
| 0x0022-0x002f | Reserved | | | | |

#### A.3.4.5.1 *Timestamp* Attribute

The *Timestamp* attribute indicates date and time in which the charged service event occurs.

The ISO8601 or equivalent representation is used.

#### A.3.4.5.2 *Duration* Attribute

The *Duration* attribute indicates the duration of a charged session (time elapsed between the reception of Start and Stop Billing Sessions (or Billing Status Notification with a Not Allowed status) for a single service transaction or, in case of device failure, the time elapsed between the start of the billing session and the reception of last Keep Alive Session command). It needs to be calculated before sending the final Bill Record.

### A.3.4.6 Commands Received

The received command IDs for the Billing cluster are listed in Table A.19.

**Table A.19 Server Received Command IDs for the Billing Cluster**

| Command Identifier Field Value | Description | Mandatory/ Optional |
|---|---|---|
| 0x00 | Subscribe | O |
| 0x01 | Unsubscribe | O |
| 0x02 | Start Service Session | O |
| 0x03 | Stop Service Session | O |

Table A.19 Server Received Command IDs for the Billing Cluster (Continued)

| Command Identifier Field Value | Description | Mandatory/ Optional |
|---|---|---|
| 0x04 | Bill Status Notification | O |
| 0x05 | Session Keep Alive | O |
| 0x06 - 0xff | Reserved | |

### A.3.4.6.1 Subscribe Command

The Subscribe command is used to request a subscription of certain user for certain periodic service on charge. It shall be originated by the user device and sent to a device connected to the billing platform (through ZigBee or other means), which is expected to answer with a ZCL Generic Response (as defined in ZCL).

The Subscribe command shall be formatted as illustrated in Figure A.50.

| Octets: | Variable | Variable | 2 | 2 |
|---|---|---|---|---|
| Data Type | - | Octet string | Unsigned 16-bit integer | Unsigned 16-bit integer |
| Field Name | ZCL Header | UserID | ServiceID | ServiceProviderID |

**Figure A.50** Format of the Subscribe Command

## A.3.4.7 Unsubscribe Command

The Unsubscribe command is used to remove a subscription of certain user for certain service on charge. It shall be originated by the user device and sent to a device connected to the billing platform (through ZigBee or other means), which is expected to answer with a ZCL Generic Response (as defined in [R4]).

The Unsubscribe command shall be formatted as illustrated in Figure A.51. Data Types are the same as the ones used for the subscribe command.

| Octets: | Variable | 2 | 2 |
|---|---|---|---|
| ZCL Header | UserID | ServiceID | ServiceProviderID |

**Figure A.51** Format of the Unsubscribe Command

### A.3.4.7.1 Start Billing Session Command

The Start Billing Session command is used for a device to notify the billing platform it is going to run a service application on charge (e.g. TA Information services). It shall be originated by the user device and sent to a device enabling the billed service (e.g. a TA Access Point). The recipient device is expected to answer with a ZCL Generic Response (as defined in [R4]). The Start Billing Session command shall be formatted as illustrated in Figure A.52. Data Types are the same as the ones used for the subscribe command.

| **Octets:** | **Variable** | **2** | **2** |
|---|---|---|---|
| ZCL Header | UserID | ServiceID | ServiceProviderID |

**Figure A.52** Format of the Start Billing Session Command

### A.3.4.7.2 Stop Billing Session Command

The Stop Billing Session command is used for a device to notify the billing platform it is going to stop billing a service application on charge (e.g. TA Information services). It shall be originated by the user device and sent to a device enabling the billed service (e.g. a TA Access Point), which is expected to answer with a ZCL Generic Response (as defined in [R4]). The Stop Billing Session command shall be formatted as illustrated in Figure A.53. Data Types are the same as the ones used for the subscribe command.

| **Octets:** | **Variable** | **2** | **2** |
|---|---|---|---|
| ZCL Header | UserID | ServiceID | ServiceProviderID |

**Figure A.53** Format of the Stop Billing Session Command

### A.3.4.7.3 Bill Status Notification Command

The Bill Status Notification command is used by the billing platform to inform service provider device e.g. information node or access point of the billing state of the user. It shall be originated by a billing platform and it shall be sent towards a service provider device (e.g. information node or access points) to inform it about the user billing status and stop providing the service in case the user has reached the credit limit. This command is expected to be answered with a ZCL Generic Response (as defined in [R4]).

The Bill Status Notification command shall be formatted as illustrated in Figure A.54.

| **Octets:** | **Variable** | **Variable** | **1** |
|---|---|---|---|
| Data Type | - | Octet string | 8-bit enumeration |
| Field Name | ZCL Header | UserID | Status |

**Figure A.54** Format of the Bill Status Notification Command

The Status byte is described in Figure A.55.

| Byte | Status |
|---|---|
| 0x00 | Not Allowed |
| 0x01 | Allowed |
| 0x02-0xFF | Reserved |

**Figure A.55** Format of the Status Byte

#### A.3.4.7.4 Session Keep Alive Command

The Session Keep Alive command is send by a user device under billing to a device offering the service (e.g. TA Access Points) to confirm it is still working and in order to maintain the session alive. It shall be originated by the user device and it shall be sent to a device that provides the service. This command is expected to be answered with a ZCL Generic Response (as defined in [R4]. After reception of this command the recipient application restores *SessionInterval* attribute to the original value so the session can be continued.

The Session Keep Alive command shall be formatted as illustrated in Figure A.56.

| Octets: | Variable | 2 | 2 |
|---|---|---|---|
| ZCL Header | UserID | ServiceID | ServiceProviderID |

**Figure A.56** Format of the Session Keep Alive Command

### A.3.4.8 Commands Generated

The generated command IDs for the server Billing cluster are listed in Table A.20

**Table A.20 Generated Command IDs for the Billing Cluster**

| Command Identifier Field Value | Description | Mandatory/Optional |
|---|---|---|
| 0x00 | Check Bill Status | M |
| 0x01 | Send Bill Record | M |
| 0x02 - 0xff | Reserved | |

#### A.3.4.8.1 Check Bill Status Command

The Check Bill Status command is used by a device supporting service billing cluster (e.g. TA access point) in order to retrieve the billing status of a specific user from the billing platform (e.g. from a Billing platform). This command is expected to be answered with a Bill Status Notification command described above.

The Check Bill Status command shall be formatted as illustrated in Figure A.57.

| Octets: | Variable | 2 | 2 |
|---|---|---|---|
| ZCL Header | UserID | ServiceID | ServiceProviderID |

**Figure A.57** Format of the Check Bill Status Command

#### A.3.4.8.2 Send Bill Record command

The Send Bill Record command is used by a device e.g. information node or access point to inform the billing platform that certain charging event is occurred. It shall be originated by a device controlling the billing events e.g. information node or access point and it shall be sent towards a billing platform on behalf of the device used the service. This command is expected to be answered with a Bill Status Notification command.

The Send Bill Record command shall be formatted as illustrated in Figure A.58.

| Octets: | Variable | Variable (typically 8) | 2 | 2 | Variable (typically 15) | 2 |
|---|---|---|---|---|---|---|
| Data Type | - | Octet string | Unsigned 16-bit integer | Unsigned 16-bit integer | Octet string | Unsigne d 16-bit integer |
| Field Name | ZCL Header | UserID | ServiceID | ServicePr oviderID | Timestamp | Duration |

**Figure A.58** Format of the Send Bill Record Command

## A.3.5 Client

### A.3.5.1 Commands Received

The client receives the cluster-specific commands detailed in sub-clause A.3.4.8, as required by application profiles.

### A.3.5.2 Commands Generated

The client generates the cluster-specific commands detailed in sub-clause A.3.4.6, as required by application profiles.

## A.3.6 New Data Type

No new data types are required by the Billing cluster.

# A.4 Payment Cluster

## A.4.1 Scope and Purpose

This document specifies a single cluster, the Payment cluster, which provides commands and attributes for payment scenarios including ZigBee devices. This cluster is to provide a standardized interface for the devices to pay goods, buy tickets, access payment services, etc.

This document should be used in conjunction with the ZigBee Cluster Library, Foundation Specification (see [R4]), which gives an overview of the library and specifies the frame formats and general commands used therein.

## A.4.2 Introduction

The cluster specified in this document is typically used for telecom applications, but may be used in any other application domains.

## A.4.3 Payment Scenarios

Two application independent scenarios can be identified in payment application domain. The first scenario is the one in which the EFT is operated by the UD. In this case the total process involves four phases:

1. The commercial transaction starts when the UD selects a good/service to buy/use.
2. The PD registers the starting process by generating a unique Serial Number and providing with it the UD together with the price information and a timestamp.
3. The UD executes the EFT with the received info and provides the PD with the financial info about the commercial transaction on the good identified by the Serial Number.
4. The PD registers the received financial info and informs the UD that the commercial transaction has ended.

An example of a payment process belonging to this case and using the profile detailed further in this document can be found in Figure A.66.

The second scenario is the one in which the financial transaction is operated by the PD. In this case the total process involves three phases:

1. The commercial transaction starts when the UD selects a good/service to buy/use.

**2** The PD registers the starting process by generating a unique Serial Number and executing the EFT using the Serial Number together with the price information and a timestamp.

**3** The PD informs the UD that the commercial transaction has ended providing it with the receipt of the commercial transaction

Please notice that in this case the phase 2 can be deferred and can be executed after the phase 3, for instance when EFTs are grouped in one transaction.

An example of a payment process belonging to this case and using the profile detailed further in this document can be found in Figure A.67.

## A.4.4 Cluster List

The Payment cluster is defined in the financial domain. An example of usage of this cluster is reported in Figure A.59.

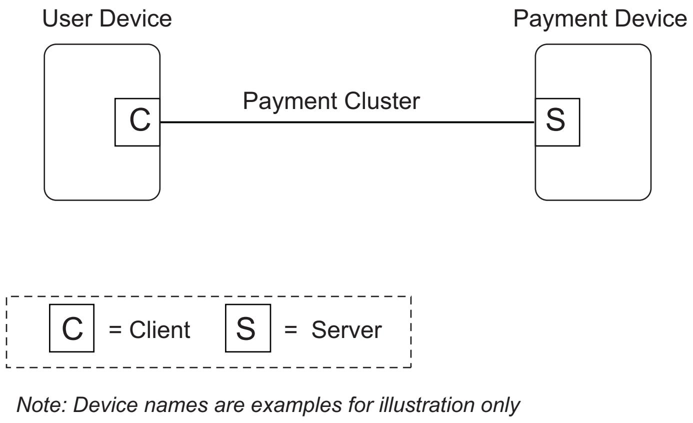


**Figure A.59** Typical Usage of the Payment Cluster

The User Device may be a ZSIM or a Mobile Terminal, whereas the Payment Device may be a POS, Ticketing Machine, etc.

## A.4.5 Overview

This cluster provides attributes and commands to enable payments of goods, tickets and services.

## A.4.6 Server

A payment device (e.g ticketing machine, POS) has a server cluster that provides payment services.

The client cluster in the user device (e.g. ZSIM, Mobile Terminal) may accept to pay for a specific good and make the actual payment transaction offline, after receiving through ZigBee the receipt from the payment device. In scenarios in which the offline payment transaction is done by the payment device, the client cluster in the user device may ask the server to buy a specific good and deliver the receipt along with the transaction status.

### A.4.6.1 Dependencies

Information cluster and security mechanisms for device mutual authentication.

### A.4.6.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in. Table A.21.

**Table A.21 Payment Attributes-Sets**

| Attribute Set Identifier | Description |
|---|---|
| 0x000 | User Information |
| 0x001 | Service Information |
| 0x002 | Price |
| 0x003 | Receipt |
| 0x004 - 0xfff | Reserved |

#### A.4.6.2.1 *User Information* Attribute Set

The *Device Information* attribute set contains the attributes summarized in Table A.22.

**Table A.22 Attributes of the *User Information* Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0000 | *UserID* | Octet string (typically 8octects) | - | ReadOnly | M |
| 0x0001 | *UserType* | Unsigned 16-bit Integer | - | ReadOnly | O |
| 0x0002-0x000f | Reserved | - | - | - | - |

##### A.4.6.2.1.1 *UserID* Attribute

The *UserID* attribute specifies a unique identification code for the user who has subscribed a specific service.

##### A.4.6.2.1.2 *UserType* Attribute

The *UserType* attribute specifies the type of user, for whom specific tariffs may be applied (e.g. students, elderly people, users who benefit of special discounts).

#### A.4.6.2.2 *Service Information* Attribute Set

The *Service Information* attribute set contains the attributes summarized in Table A.23.

**Table A.23 Attributes of the *Service Information* Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0010 | *ServiceID* | Unsigned 16-bit Integer | - | ReadOnly | M |
| 0x0011 | *ServiceProviderID* | Unsigned 16-bit Integer | - | ReadOnly | M |
| 0x0012 | *TotemID* | Unsigned 16-bit Integer | - | ReadOnly | O |
| 0x0013-0x001f | Reserved | | | | |

#### A.4.6.2.2.1 *ServiceID* Attribute

The ServiceID attribute contains the unique identifier of the service (e.g. shopping, parking, bus ticketing). This attribute is defined by the specific application scenario considered for the mobile commerce.

#### A.4.6.2.2.2 *ServiceProviderID* Attribute

The *ServiceProviderID* attribute indicates the unique identifier of the Service Provider (e.g. transport operator).

#### A.4.6.2.2.3 *TotemID* Attribute

The *TotemID* attribute indicates the unique identifier of the totem (i.e. POS, ticketing machine).

### A.4.6.2.3 *Price* Attribute Set

The *Price* attribute set contains the attributes summarized in Table A.24.

**Table A.24 Attributes of the *Price* Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0020 | *Currency* | Unsigned 32-bit Integer | - | ReadOnly | M |
| 0x0021 | *Price Trailing Digit* | Unsigned 8-bit Integer | - | ReadOnly | M |
| 0x0022 | *Price* | Unsigned 32-bit Integer | - | ReadOnly | M |
| 0x0023-0x002f | Reserved | | | | |

Price format has been aligned to the definition currently used in Smart Energy.

#### A.4.6.2.4 *Receipt* Attribute Set

The *Receipt* attribute set contains the attributes summarized in Table A.25.

**Table A.25 Attributes of the *Receipt* Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0030 | *GoodID* | Octet String | - | ReadOnly | M |
| 0x0031 | *SerialNumber* | Octet String (6 Byte) | - | ReadOnly | M |
| 0x0032 | *Timestamp* | Octet String (6 Byte) | - | ReadOnly | M |
| 0x0033 | *TransID* | Unsigned 16-bit Integer | - | Read/Write | O |
| 0x0034 | *TransStatus* | 8-bit enumeration | - | Read/Write | O |
| 0x0035 | *Status* | 8-bit enumeration | - | ReadOnly | M |
| 0x0036-0x003f | Reserved | | | | |

##### A.4.6.2.4.1 *GoodID* Attribute

The *GoodID* attribute identifies the type of good that has been bought (e.g. urban single bus ticket).

If the Payment Cluster is used in conjunction with the Information cluster the *GoodID* attribute is built just by insert the two bytes of the *ContentID* in the octet string type.

##### A.4.6.2.4.2 *SerialNumber* Attribute

The *SerialNumber* attribute indicates a unique identifier of the good, related to a specific *GoodID*.

##### A.4.6.2.4.3 *Timestamp* Attribute

The *Timestamp* attribute indicates date and time in which the good has been bought.

The 6 byte SBCD (Simplified Binary Coded Decimal or BCD 8421) format is used.

##### A.4.6.2.4.4 *TransID* Attribute

The *TransID* attribute indicates the unique identifier of the financial transaction, that can be performed by the user device or the totem, depending on the service scenario.

#### A.4.6.2.4.5 *TransStatus* Attribute

The *TransStatus* attribute indicates the status resulting from the financial transaction. The attribute shall contain one of the values described in Table A.26

**Table A.26 Allowed Values for the *TransStatus* Attribute**

| ***TransStatus* Attribute Value** | **Description** |
|---|---|
| 0x00 | Ok |
| 0x01 | Not Permitted |
| 0xff | General Failure |
| 0x02 - 0xfe | Reserved |

#### A.4.6.2.4.6 *Status* Attribute

The *Status* attribute indicates the status resulting from the commercial transaction. The attribute shall contain one of the values described in Table A.27. If its value is different from 0x00 the commercial transaction shall be considered aborted. See Figure A.68 for an instance of this last case.

**Table A.27 Allowed Values for the *Status* Attribute**

| ***TransStatus* Attribute Value** | **Description** |
|---|---|
| 0x00 | Ok |
| 0x01 | Incorrect Data |
| 0xff | General failure |
| 0x02 - 0xfe | Reserved |

### A.4.6.3 Commands Received

The received commands IDs for the Payment cluster are listed in Table A.28. Please notice that the optional/ fields could be stated as mandatory inside the device description of a particular profile specification.

**Table A.28 Received Command IDs for the Payment Cluster**

| Command Identifier Field Value | Description | Mandatory/Optional |
|---|---|---|
| 0x00 | BuyRequest | O |
| 0x01 | AcceptPayment | O |
| 0x02 | PaymentConfirm | O |
| 0x03 - 0xff | Reserved | |

#### A.4.6.3.1 Buy Request Command

The Buy Request command is used by a device to request a previously selected good to another device. It shall be originated by the user device and sent to the payment device, which then makes the offline payment transaction and sends a Buy Confirm including payment receipt and transaction related data.

The Buy Request command shall be formatted as illustrated in Figure A.60.

| Octets: | Variable | Variable | 2 | 2 | Variable |
|---|---|---|---|---|---|
| Data Type | - | Octet string | Unsigned 16-bit Integer | Unsigned 16-bit Integer | Octet string |
| Field Name | ZCL Header | UserID | UserType | ServiceID | GoodID |

**Figure A.60** Format of the Buy Request Command

#### A.4.6.3.2 Accept Payment Command

The Accept Payment command is used by a device to accept the payment of a specific good, which has been previously selected. It shall be originated by the user device, that will make the payment transaction, and sent to the payment device, which is supposed to answer with a Receipt Delivery.

The Accept Payment command shall be formatted as illustrated in Figure A.61

| Octets: | Variable | Variable | 2 | 2 | Variable |
|---|---|---|---|---|---|
| Data Type | - | Octet string | Unsigned 16-bit Integer | Unsigned 16-bit Integer | Octet string |
| Field Name | ZCL Header | UserID | UserType | ServiceID | GoodID |

**Figure A.61** Format of the Accept Payment Command

### A.4.6.3.3 Payment Confirm Command

The Payment Confirm command is used for a device to confirm the conclusion of the transaction related to a specific good to another device, indicating the id of the operator network transaction and the status. It shall be originated by the user device and sent to the payment device, which is supposed to answer with a transaction end, in the case of a service scenario in which the user device initiates the transaction towards the operator network.

The Payment Confirm command shall be formatted as illustrated in Figure A.62

| Octets: | Variable | Variable (typically 6) | 2 | 1 |
|---|---|---|---|---|
| Data Type | - | Octet string | Unsigned 16-bit Integer | 8-bit enumeration |
| Field Name | ZCL Header | SerialNumber | TransID | TransStatus |

**Figure A.62** Format of the Payment Confirm Command

## A.4.6.4 Commands Generated

The generated command IDs for the Payment cluster are listed in Table A.29

**Table A.29 Generated Command IDs for the Payment Cluster**

| Command Identifier Field Value | Description | Mandatory/ Optional |
|---|---|---|
| 0x00 | BuyConfirm | M |
| 0x01 | ReceiptDelivery | M |
| 0x02 | TransactionEnd | M |
| 0x03 - 0xff | Reserved | |

#### A.4.6.4.1 Buy Confirm Command

The Buy Confirm command is used by a device to confirm the payment and deliver the receipt for the requested good, along with the transaction related data. It shall be originated by the payment device and sent to the user device, which has previously asked for a buy request.

The Buy Confirm command shall be formatted as illustrated in Figure A.63.

| **Octets:** | **Variable** | **Variable (typically 6)** | **9** | **Variable (typically 6)** | **2** | **1** |
|---|---|---|---|---|---|---|
| Data Type | - | Octet string | See A.4.6.2.3 | Octet string | Unsigned 16-bit Integer | 8-bit enumeration |
| Field Name | ZCL Header | SerialNumber | Price set | Timestamp | TransID | TransStatus |

**Figure A.63** Format of the Buy Confirm Command

#### A.4.6.4.2 Receipt Delivery Command

The Receipt Delivery command is used by a device to deliver the receipt for the requested good, ticket or service. It shall be originated by the payment device and sent to the user device, which has previously accepted the payment.

The Receipt Delivery command shall be formatted as illustrated in Figure A.64.

| **Octets:** | **Variable** | **Variable (typically 6)** | **9** | **Variable (typically 6)** |
|---|---|---|---|---|
| Data Type | - | Octet string | See A.4.6.2.3 | Octet string |
| Field Name | ZCL Header | SerialNumber | Price set | Timestamp |

**Figure A.64** Format of the Receipt Delivery Command

#### A.4.6.4.3 Transaction End Command

The Transaction End command is used for a device to acknowledge the transaction end reception. It shall be originated by the payment device and sent to the user device, in the case of a service scenario in which the user device initiates the transaction towards the operator network or whenever the transaction is aborted. In this case the *Status* attribute indicates the failure reason.

The Transaction End command shall be formatted as illustrated in Figure A.65.

| **Octets:** | **Variable** | **Variable (typically 6)** | **1** |
|---|---|---|---|
| Data Type | - | Octet string | 8-bit enumeration |
| Field Name | ZCL Header | SerialNumber | Status |

**Figure A.65** Format of the Transaction End Command

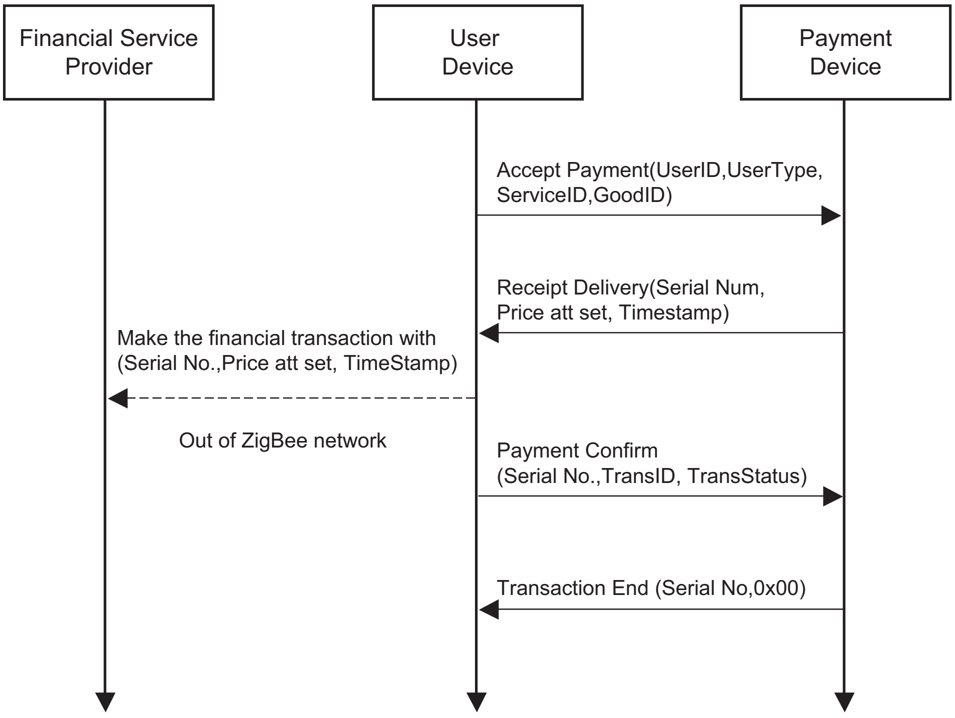

Financial Service Provider
User Device
Payment Device
Accept Payment(UserID,UserType, ServiceID,GoodID)
Receipt Delivery(Serial Num, Price att set, Timestamp)
Make the financial transaction with (Serial No.,Price att set, TimeStamp)
Out of ZigBee network
Payment Confirm (Serial No.,TransID, TransStatus)
Transaction End (Serial No,0x00)

**Figure A.66** Example of Scenario in Which the EFT is Made by the User Device

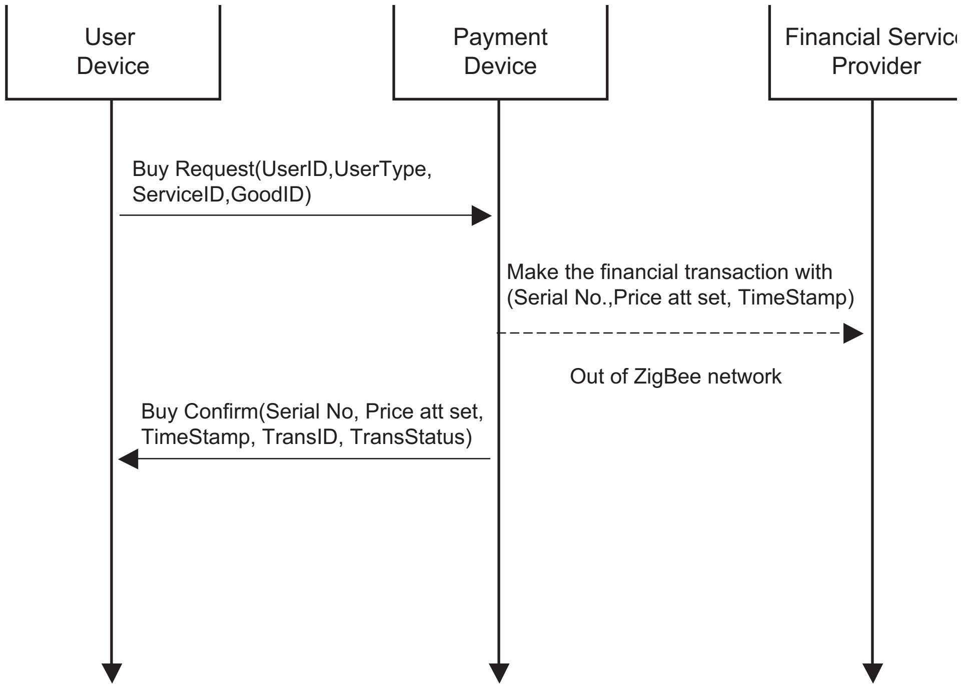

User Device
Payment Device
Financial Servic
Provider
Buy Request(UserID,UserType, ServiceID,GoodID)
Make the financial transaction with (Serial No.,Price att set, TimeStamp)
Out of ZigBee network
Buy Confirm(Serial No, Price att set, TimeStamp, TransID, TransStatus)

**Figure A.67** Example of Scenario in Which the EFT is Made by the Payment Device

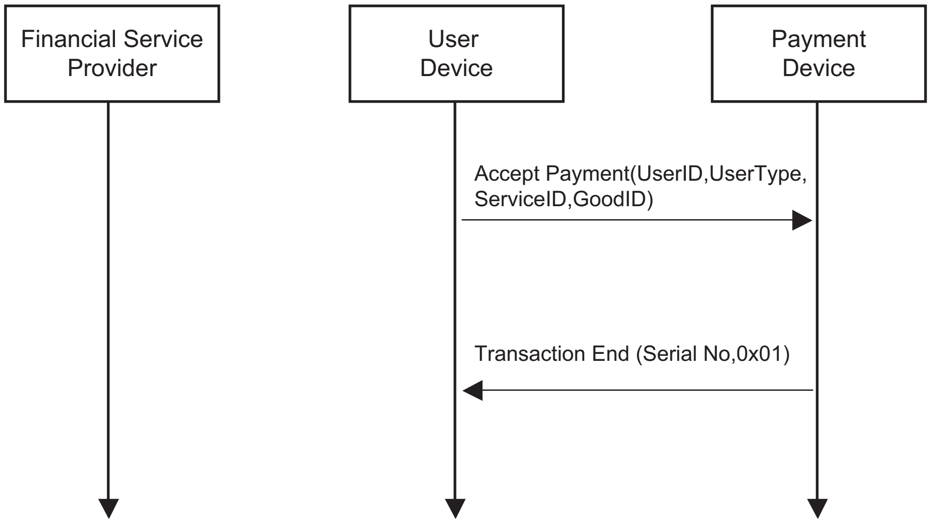

Financial Service Provider
User Device
Payment Device
Accept Payment(UserID,UserType, ServiceID,GoodID)
Transaction End (Serial No,0x01)

**Figure A.68** Example of Scenario in which the User Device Sent Incorrect Data (i.e. GoodID not Recognized by the Payment Device)

### A.4.7 Client

#### A.4.7.1 Commands Received

The client receives the cluster-specific commands detailed in sub-clause A.4.6.4 as required by application profiles

#### A.4.7.2 Commands Generated

The client generates the cluster-specific commands detailed in sub-clause A.4.6.3 as required by application profiles.

## A.5 Data Sharing Cluster

### A.5.1 Scope and Purpose

This document specifies a single cluster, the data sharing cluster, which provides commands and attributes for small data sharing among ZigBee devices. This cluster is to provide a standardized interface for the devices to share data files such as address book, small pictures, etc.

This document should be used in conjunction with the ZigBee Cluster Library, Foundation Specification (see [R4]), which gives an overview of the library and specifies the frame formats and general commands used therein.

### A.5.2 Introduction

The cluster specified in this document is typically used for telecom applications, but may be used in any other application domains. This cluster may use Partition Cluster.

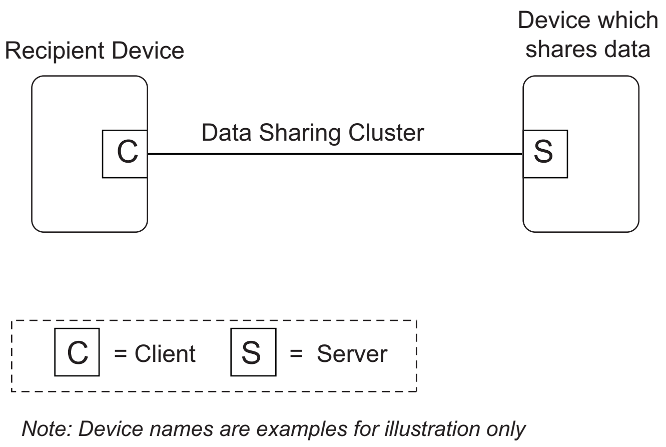


**Figure A.69** Typical Usage of the P2P Data Sharing Cluster

## A.5.3 Overview

This cluster provides attributes and commands for devices to share their data. Another device then is able to read a file from the sharing data server device, or the sharing data server device is able to write a file to other devices.

## A.5.4 Server

The server stores the data to be shared. It may respond to the request from other devices and transmit the data to them, or it may actively request other devices to transmit the data to them.

### A.5.4.1 Dependencies

This cluster does not depend on any other existing clusters. However, in order to successfully fulfill the data sharing, Partition Cluster may also need to be implemented in the same device where the data sharing cluster is implemented.

### A.5.4.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table A.30.

**Table A.30 *P2P Data Sharing* Attribute Sets**

| Attribute Set Identifier | Description |
|---|---|
| 0x000 | Device Information |
| 0x001 - 0xfff | Reserved |

### A.5.4.2.1 *Device Information* Attribute Set

The *Device Information* attribute set contains the attributes summarized in Table A.31.

**Table A.31 Attributes of the *Device Information* Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0000 | *DeviceName* | Character String | | ReadOnly | M |
| 0x0001 | *DeviceDescription* | Character String | | ReadOnly | O |
| 0x0002- 0x000f | Reserved | | | | |

#### A.5.4.2.1.1 *DeviceName* Attribute

The *DeviceName* attribute specifies the name of the device in character string.

#### A.5.4.2.1.2 *DeviceDescription* Attribute

The *DeviceDescription* attribute specifies the descriptions of the device, such as application device type.

## A.5.5 Commands Received

The received command IDs for the P2P data sharing cluster are listed in Table A.32

**Table A.32 Received Command IDs for the P2 Data Sharing Cluster**

| Command Identifier Field Value | Description | Mandatory/Optional |
|---|---|---|
| 0x00 | Read File Request | M |
| 0x01 | Read Record Request | O |
| 0x02 | Write File Response | O |
| 0x03 - 0xff | Reserved | |

### A.5.5.1 Read File Request Command

The Read File Request command is used for a device to request for a file by stream access method from another device. In this case, the file shall be transmitted in octet stream. It shall be originated by the file transmission destination device and sent to the transmission source device.

The Read File Request command shall be formatted as illustrated in Figure A.70.

| Octets: | Variable | 2 | 0/4 | 0/4 |
|---|---|---|---|---|
| Data Type | - | Unsigned 16-bit integer | Unsigned 32-bit integer | Unsigned 32-bit integer |
| Field Name | ZCL Header | File Index | File Start Position | Requested Octet Count |

**Figure A.70** Format of the Read File Request Command

The File Index field indicates the index of the file stored inside the server. If this field is 0x0000, it indicates to get the root directory of the counterpart. The File Start Position field indicates the number of octets from the beginning of the requested file at which reading shall commence. If its value is zero, the reading shall commence from the first octet of the file. The Requested Octet Count field indicates the number of octets that shall be read from the file starting at the File Start Position. If its value is 0xffffffff, the reading shall commence from the File Start Position to the end of the file. The File Start Position field and the Requested Octet Count field shall both exist, or neither of them shall exist. In the case that they do not exist, it is the same as setting File Start Position to zero and Requested Octet Count to 0xffffffff, indicating to read the whole file.

### A.5.5.2 Read Record Request Command

The Read Record Request command is used for a device to request for a file by record access method from another device. In this case, the file shall be transmitted in records. It shall be originated by the file transmission destination device and sent to the transmission source device.

The Read Record Request command shall be formatted as illustrated in Figure A.71

| **Octets:** | **Variable** | **2** | **0/2** | **0/2** |
|---|---|---|---|---|
| Data Type | - | Unsigned 16-bit integer | Unsigned 16-bit integer | Unsigned 16-bit integer |
| Field Name | ZCL Header | File Index | File Start Record | Requested Record Count |

**Figure A.71** Format of the Read Record Request Command

The File Index field indicates the index of the file stored inside the server. If this field is 0x0000, it indicates to get the root directory of the counterpart. The File Start Record field indicates the number of records from the beginning of the requested file at which reading shall commence. If its value is zero, the reading shall commence from the first record of the file. The Requested Record Count field indicates the number of records that shall be read from the file starting at the File Start Record. If its value is 0xffff, the reading shall commence from the File Start Record to the end of the file. The File Start Record field and the Requested Record Count field shall both exist, or neither of them shall exist. In the case that they do not exist, it is the same as setting File Start Record to zero and Requested Record Count to 0xffff, indicating to read the whole file.

### A.5.5.3 Write File Response Command

The Write File Response command is used for a device to return back the response to the Write File Request command. It shall be originated by the file transmission destination device and sent to the transmission source device.

The Write File Response command shall be formatted as illustrated in Figure A.72.

| **Octets:** | **Variable** | **1** | **0/2** |
|---|---|---|---|
| Data Type | - | 8-bit enumeration | Unsigned 16-bit integer |
| Field Name | ZCL Header | Status | File Index |

**Figure A.72** Format of the Write File Response Command

The Status field indicates the status of the Write File Request procedure. Its values shall be those as specified in [R4]. If this field is not SUCCESS, the File Index field shall not exist. The File Index field indicates the receiver assigned index of the file to be written.

## A.5.6 Commands Generated

The generated command IDs for the P2P Data Sharing cluster are listed in Table A.33

**Table A.33 Generated Command IDs for the Data Sharing Cluster**

| Command Identifier Field Value | Description | Mandatory/Optional |
|---|---|---|
| 0x00 | Write File Request | O |
| 0x01 | Modify File Request | O |
| 0x02 | Modify Record Request | O |
| 0x03 | File Transmission | M |
| 0x04 | Record Transmission | O |
| 0x05 - 0xff | Reserved | |

### A.5.6.1 Write File Request Command

The Write File Request command is used for a device to actively request for a file transmission to other devices. It shall be originated by the file transmission source device and sent to the transmission destination device. After transmitting this command, the device should wait for the Write File Response command to decide what to do next.

The Write File Request command shall be formatted as illustrated in Figure A.73.

| Octets: | Variable | 1 | 2/4 |
|---|---|---|---|
| Data Type | - | 8-bit bitmap | Unsigned 16/32-bit integer |
| Field Name | ZCL Header | Write Option | File Size |

**Figure A.73** Format of the Write File Request Command

The Write Option field indicates the options of writing file operation. The least significant bit, i.e. bit 0 indicates which type of file the operation shall be applied for. 0b0 indicates to write an octet file, and 0b1 indicates to write a record file. Other bits in the Write Option field are reserved. The File Size field indicates the size of the file which is to be written. If the Write Option field indicates to write an octet file, the File Size field shall be 4 octets field. And the value 0xffffffff indicates the unknown file size. If the Write Option field indicates to write a record file, the File Size field shall be 2 octets field. And the value 0xffff indicates the unknown file size. Receiver may prepare the buffer for the file beforehand.

| **Octets:** | **Variable** | **2** | **4** | **4** |
|---|---|---|---|---|
| Data Type | - | Unsigned 16-bit Integer | Unsigned 32-bit Integer | Unsigned 32-bit Integer |
| Field Name | ZCL Header | File Index | File Start Position | Octet Count |

**Figure A.74** Format of the Modify File Request Command

The File Index field indicates the index of the file stored inside the server. The File Start Position field indicates the number of octets from the beginning of the requested file at which the data shall start being written. If its value is zero, the writing shall commence from the first octet of the file, i.e. overwriting the whole file. If its value is 0xffffffff, it indicates the end of the file, i.e. appending to file operation. The Octet Count field indicates the number of octet to be modified in the destined file.

### A.5.6.2 Modify Record Request Command

The Modify Record Request command is used for a device to request to modify a record file stored in another device. It shall be originated by the record file transmission source device and sent to the transmission destination device. In order to get the response from the receiver, the Disable default response sub-field in the ZCL header frame control field should be set to zero.

The Modify Record Request command shall be formatted as illustrated in Figure A.75.

| **Octets:** | **Variable** | **2** | **2** | **2** |
|---|---|---|---|---|
| Data Type | - | Unsigned 16-bit Integer | Unsigned 16-bit Integer | Unsigned 16-bit Integer |
| Field Name | ZCL Header | File Index | File Start Record | Record Count |

**Figure A.75** Format of the Modify Record Request Command

The File Index field indicates the index of the file stored inside the server. The File Start Record field indicates the number of records from the beginning of the requested file at which the record data shall start being written. If its value is zero, the writing shall commence from the first record of the file, i.e. overwriting the whole record file. If its value is 0xffff, it indicates the end of the file, i.e. appending to record file operation. The Record Count field indicates the number of record to be modified in the destined record file.

### A.5.6.3 File Transmission Command

The File Transmission command is used for a device to transmit sharing data to other devices. It shall be originated by the file transmission source device and sent to the transmission destination device. The File Transmission command shall be formatted as illustrated in Figure A.76.

| **Octets:** | **Variable** | **1** | **0/2** | **0/4** | **0/4** | **0/1** | **…** | **0/1** |
|---|---|---|---|---|---|---|---|---|
| Data Type | - | 8-bit bitmap | Unsigned 16-bit integer | Unsigned 32-bit integer | Unsign ed 32-bit integer | 8-bit data | - | 8-bit data |
| Field Name | ZCL Header | Transmit Options | File Index | File Start Position | File Length | Octet 1 | … | Octet n |
| | | | | | | File Data | | |

**Figure A.76** Format of the File Transmission Command

The Transmit Options field indicates the transmission options and status, including read or write, directory or not, end of file and transmission status. The format of Transmit Options field shall be illustrated in Figure A.77. The least significant bit, i.e. the bit 0 specifies whether the transmission is for read file request or not. If the bit field is set to one, it indicates the transmission is in response to the request of reading a file, or else it indicates the transmission is in response to the agreement of writing or modifying a file. Bit 1 specifies whether the file to be transmitted is a directory or not. If the bit field is set to one, it indicates the file to be transmitted is a directory, or else it indicates the file to be transmitted is not a directory. Bit 2 specifies whether the file data include the last octet of the file. If the bit field is set to one, it indicates the file data include the last octet of the file, or else it indicates the file data do not include the last octet of the file. In the cases of modifying a file, the EOF bit field indicates the way to modify the file. If the bit field is set to one, to include the last octet of the file means to replace all remaining octets from the File Start Position with the octets carried in File Data field. If the bit field is set to zero, not to include the last octet of the file means to replace the octets from the File Start Position with the equal number of octets carried in File Data field. Bit 3 and bit 4 are reserved. Bits 5 to 7 specify the status of the operation. Value 0b000 indicates SUCCESS status. Value 0b001 indicates FAILURE status. Other values are reserved. If this field is not SUCCESS, the File Index field, the File Start Position field, the File Length field and the File Data field shall not exist, and then this command is to carry back the transmission status from the receiver. The File Index field indicates the index of the file, i.e. the same value as the field in Read File Request command, Write File Response command or Modify File Request command in different cases. The File Start Position field indicates the position of transmitting file data at the original file, i.e. the number of octets from the beginning of the original file. The File Length field indicates how many octets the file contains. The File Data field contains the content of the file, i.e. the octets of the sharing data. In the cases of writing or modifying a file, the number of octets contained in the File Data should be the same as the value indicated in the File Size field of corresponding Write File Request command or Octet Count field of corresponding Modify File Request command. Default Response command with INVALID_FIELD status may be returned by the client after receiving the File Transmission command with incorrect File Data length. This command may use Partition Cluster.

| b0 | b1 | b2 | b3-b4 | b5-b7 |
|---|---|---|---|---|
| Read | Directory | EOF | Reserved | Transmission Status |

**Figure A.77** Format of the TransmitOptions Field

### A.5.6.4 Record Transmission Command

The Record Transmission command is used for a device to transmit sharing record data to other devices. It shall be originated by the record file transmission source device and sent to the transmission destination device.

The Record Transmission command shall be formatted as illustrated in Figure A.78.

| Octets: | Variable | 1 | 0/2 | 0/2 | 0/2 | 0/1 | … | 0/1 |
|---|---|---|---|---|---|---|---|---|
| Data Type | - | 8-bit bitmap | Unsigned 16-bit integer | Unsigned 16-bit integer | | Character string | … | Character string |
| Field Name | ZCL Header | Transmit Options | File Index | File Start Record | Record Count | Record 1 | … | Record n |
| | | | | | | Record File Data | | |

**Figure A.78** Format of the Record Transmission Command

The Transmit Options field indicates the transmission options and status, including read or write, directory or not, end of file and transmission status. The format of Transmit Options field shall be illustrated in Figure A.77. The least significant bit, i.e. the bit 0 specifies whether the transmission is for read file request or not. If the bit field is set to one, it indicates the transmission is in response to the request of reading a file, or else it indicates the transmission is in response to the agreement of writing or modifying a file. Bit 1 specifies whether the file to be transmitted is a directory or not. If the bit field is set to one, it indicates the file to be transmitted is a directory, or else it indicates the file to be transmitted is not a directory. Bit 2 specifies whether the file data include the last record of the file. If the bit field is set to one, it indicates the file data include the last record of the file, or else it indicates the file data do not include the last record of the file. In the cases of modifying a record file, the EOF bit field indicates the way to modify the file. If the bit field is set to one, to include the last record of the file means to replace all remaining records from the File Start Record with the records carried in Record File Data field. If the bit field is set to zero, not to include the last record of the file means to replace the records from the File Start Record with the equal number of records carried in Record File Data field. Bit 3 and bit 4 are reserved. Bits 5 to 7 specify the status of the operation. Value 0b000 indicates SUCCESS status. Value 0b001 indicates FAILURE status. Other values are reserved. If this field is not SUCCESS, the File Index field, the File Start Record field, the Record Count field and the Record File Data field shall not exist, and then this command is to carry back the transmission status from the receiver. The File Index field indicates the index of the file, i.e. the same value as the field in Read File Request command, Write File Response command or Modify File Request command in different cases. The File Start Record field indicates the position of transmitting record file data at the original file, i.e. the number of records from the beginning of the original file. The Record Count field indicates how many records the file contains. The Record File Data field contains the content of the record file, i.e. the records of the sharing data. In the cases of writing or modifying a record file, the number of records contained in the Record File Data should be the same as the value indicated in the File Size field of corresponding Write File Request command or Record Count field of corresponding Modify Record Request command. Default Response command with INVALID_FIELD status may be returned by the client after receiving the Record Transmission command with incorrect Record File Data length. This command may use Partition Cluster.

## A.6 Data Rate Control Cluster

### A.6.1 Introduction

#### A.6.1.1 Scope and Purpose

This clause specifies a single cluster, the data rate control cluster, which provides commands and attributes for data rate control purpose of ZigBee devices (see [R14]). This cluster is to provide a standardized interface for the devices to decide the most appropriate data rate of themselves so that they can achieve good transmission performance and do not impact the normal operations of other devices.

This document should be used in conjunction with the ZigBee Cluster Library, Foundation Specification (see [R4]), which gives an overview of the library and specifies the frame formats and general commands used therein.

### A.6.2 General Description

An important technical requirement coming from several use cases (e.g. P2P Small Data Sharing or VoZ) is represented by the control of the impact of these applications on the network performances of a pre-installed ZigBee network. In order to achieve that by leveraging on the existing stack, the data rate control cluster should be defined in order to limit the throughput between the devices running with high duty cycle and the others. Performing an analysis on a pre-installed ZigBee network, it is possible for high duty cycle devices to reduce the throughput accordingly and spare the channel usage for the other devices. The analysis on latency and bandwidth requirements of neighbor devices should be performed. Because higher data rate causes more packet collisions or more back-offs, which means more latency, and higher data rate also means consuming more bandwidth while leaving less resources to other devices (see [R3]).

Any device implementing the client side of the data rate control cluster will be considered device which needs to control its data rate.

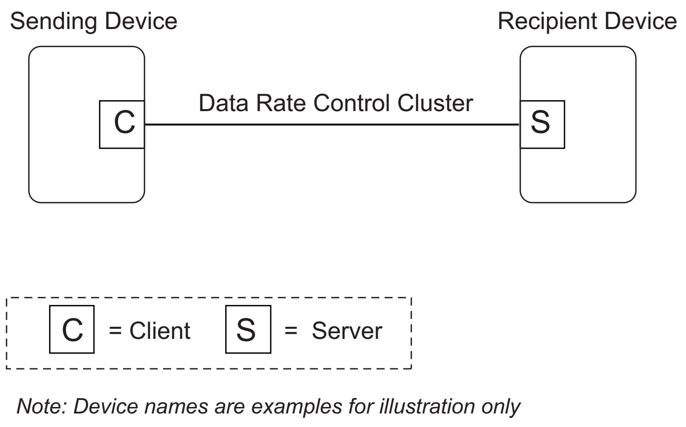

Sending Device
Recipient Device
Data Rate Control Cluster
C
S
C = Client
S = Server
Note: Device names are examples for illustration only

**Figure A.79** Typical Usage of the Data Rate Control Cluster

## A.6.3 Overview

This cluster provides attributes and commands for a device to report its latency and bandwidth requirements or acquire the latency and bandwidth requirements of others, and also for a device to set appropriate transmission parameters so as to enhance the transmission performance or reduce the impact on normal operations of others. Devices on the communication link should decide appropriate transmission parameters (e.g. apsInterframeDelay, polling rate, etc.) in order to set an appropriate data rate for them, according to the latency or bandwidth requirements of neighbor devices, or for their own performance enhancement purpose.

Data rate control procedure should be performed for the application which requires high data rate. Usually only the source device and destination device should be involved in the data rate control procedure. Intermediate device in the communication link may participate in the data rate control procedure, relay and process the necessary data rate control commands if all of the following preconditions are acquired: (1) The device supports the cluster. (2) Its previous hop participates in the procedure and wants it to be involved in also. (3) Its previous hop supports acquiring the next hop device address, e.g. using ZDO command Mgmt_Rtg_req. And to be noticed the routing may change during the transmission, so the implementer should decide whether to involve the intermediate devices and the detailed methods are implementer dependent. Traffic originated by the lower layer may be counted in and estimated during the data rate control procedure by the implementer specific method.

## A.6.4 Server

The attributes accessible on the server side of this cluster are latency and bandwidth requirements of devices.

### A.6.4.1 Dependencies

None.

### A.6.4.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table A.34.

**Table A.34** ***Data Rate Control*** **Attribute Sets**

| Attribute Set Identifier | Description |
|---|---|
| 0x000 | Transmission Requirement Parameters |
| 0x001 - 0xfff | Reserved |

#### A.6.4.2.1 *Transmission Requirement Parameters* Attribute Set

The *Transmission Requirement Parameters* attribute set contains the attributes summarized in Table A.35.

**Table A.35 Attributes of the** ***Transmission Requirement Parameter*** **Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0000 | *AverageLatency Requirement* | Unsigned 8-bit Integer | 0x00-0xff | ReadOnly | O |
| 0x0001 | *MaxLatency Requirement* | Unsigned 8-bit Integer | 0x00-0xff | ReadOnly | O |
| 0x0002 | *Bandwidth Requirement* | Unsigned 8-bit Integer | 0x00-0xff | Read/ Write | O |
| 0x0003-0x000f | Reserved | | | | |

#### A.6.4.2.1.1 *AverageLatencyRequirement* Attribute

For data rate control purpose, a device may investigate average latency requirements of its neighbors. A device may also report its average latency requirement to its neighbors. The so-called latency requirement is one-hop latency requirement so that it's convenient for data rate control calculation. It may be estimated according to the end to end latency requirement. The value of real average latency requirement shall be calculated by the formula: *AverageLatencyRequirement* * 50ms. The normal value of *AverageLatencyRequirement* is 0x01-0xfe. Value 0xff for *AverageLatencyRequirement* means the neighbor device has no average latency requirement. Value 0x00 is reserved.

#### A.6.4.2.1.2 *MaxLatencyRequirement* Attribute

For data rate control purpose, a device may investigate maximum latency requirements of its neighbors. A device may also report its maximum latency requirement to its neighbors. The so-called latency requirement is one-hop latency requirement so that it's convenient for data rate control calculation. It may be estimated according to the end to end latency requirement. Since there's no absolute maximum latency, the maximum latency here is defined as the latency value which is larger than 95% of latency values in all cases. The value of real maximum latency requirement shall be calculated by the formula: *MaxLatencyRequirement* * 100ms. The normal value of *MaxLatencyRequirement* is 0x01-0xfe. Value 0xff for *MaxLatencyRequirement* means the neighbor device has no maximum latency requirement. Value 0x00 is reserved.

#### A.6.4.2.1.3 *BandwidthRequirement* Attribute

For data rate control purpose, a device may investigate bandwidth requirements of its neighbors. A device may also report its bandwidth requirement to other devices. The transmitting device may scan the other ZigBee devices that are in the area (sharing the same channel and resources) and set the attribute accordingly. The value of real bandwidth requirement shall be equal to BandwidthRequirement (kbps). The normal value of BandwidthRequirement is 0x00-0xfa. Value 0xfb-0xff is reserved. A value of BandwidthRequirement equal to zero would be ignored.

### A.6.4.3 Commands Received

The received command IDs for the data rate control cluster are listed in Table A.36

**Table A.36 Command IDs for the Data Rate Control Cluster**

| Command Identifier Field Value | Description | Mandatory/ Optional |
|---|---|---|
| 0x00 | Path Creation | O |
| 0x01 | Data Rate Notification | O |
| 0x02 | Path Deletion | O |
| 0x03 - 0xff | Reserved | |

#### A.6.4.3.1 Path Creation Command

The Path Creation command is used to tell the devices in the communication link how to prepare for the data rate control procedure. It should be originated by the application source device and sent to the application destination device. And the involved devices may update parameters such as BandwidthRequirement according to the data rate field, e.g. to add the anticipated data rate value.

The Path Creation command shall be formatted as illustrated in Figure A.80.

| **Octets:** | **Variable** | **2** | **2** | **1** |
|---|---|---|---|---|
| Data Type | - | 16-bit data | 16-bit data | Unsigned 8-bit integer |
| Field Name | ZCL Header | Originator Address | Destination Address | Data Rate |

**Figure A.80** Format of the Path Creation Command

The originator address field indicates the network address of the device which originates the command at first, i.e. the application source address. The destination address field indicates the destination address of the device at which this command should be ended, i.e. the application destination address. The data rate field indicates the anticipated data rate, i.e. bandwidth to be reserved, for the communication link, and its unit is kbps.

#### A.6.4.3.2 Data Rate Notification Command

The Data Rate Notification command is used to notify the devices in the communication link the negotiated data rate, so that they can set their parameters such as BandwidthRequirement, or polling rate for sleeping end device. It shall be originated by the application source device and may be sent to the application destination device. Source device shall set transmission parameters such as apsInterframeDelay according to the set data rate.

The Data Rate Notification command shall be formatted as illustrated in Figure A.81.

| **Octets:** | **Variable** | **2** | **2** | **1** |
|---|---|---|---|---|
| Data Type | - | 16-bit data | 16-bit data | Unsigned 8-bit integer |
| Field Name | ZCL Header | Originator Address | Destination Address | Data Rate |

**Figure A.81** Format of the Data Rate Notification Command

The originator address field indicates the network address of the device which originates the command at first, i.e. the application source address. The destination address field indicates the application destination address. The data rate field indicates the data rate set by the source device and its unit is kbps.

### A.6.4.3.3 Path Deletion Command

The Path Deletion command is used to tell the devices in the communication link that the session has finished so that they can retrieve their parameters such as BandwidthRequirement or polling rate. It should be originated by the application source device and sent to the application destination device.

The Path Deletion command shall be formatted as illustrated in Figure A.82.

| **Octets:** | **Variable** | **2** | **2** |
|---|---|---|---|
| Data Type | - | 16-bit data | 16-bit data |
| Field Name | ZCL Header | Originator Address | Destination Address |

**Figure A.82** Format of the Path Creation Command

The originator address field indicates the network address of the device which originates the command at first, i.e. the application source address. The destination address field indicates the application destination address.

### A.6.4.4 Commands Generated

The generated command IDs for the Data Rate Control cluster are listed in Table A.37

**Table A.37 Generated Command IDs for the Data Rate Control Cluster**

| Command Identifier Field Value | Description | Mandatory/ Optional |
|---|---|---|
| 0x00 | Data Rate Control | M |
| 0x01 - 0xff | Reserved | |

#### A.6.4.4.1 Data Rate Control Command

The Data Rate Control command is used for the devices in the communication link to negotiate an appropriate data rate. It should be sent from the application destination device to the application source device. The source device or intermediate device should set a data rate according to the latency or bandwidth requirements of its neighbours. After receiving this command, the device in the communication link shall compare the data rate field with the data rate it sets previously, so as to get a smallest data rate among the devices in the communication link, i.e. smaller data rate would be set during the comparison. Then the source device shall use this smallest data rate value for setting the transmission data rate.

The Data Rate Control command shall be formatted as illustrated in Figure A.83.

| Octets: | Variable | 2 | 2 | 1 |
|---|---|---|---|---|
| Data Type | - | 16-bit data | 16-bit data | Unsigned 8-bit integer |
| Field Name | ZCL Header | Originator Address | Destination Address | Data Rate |

**Figure A.83** Format of the Data Rate Control Command

The originator address field indicates the network address of the device at which this command shall be ended, i.e. the application source address. The destination address field indicates the application destination address. The data rate field indicates the currently maximal data rate which can be set, and its unit is kbps.

### A.6.4.5 General Use of Data Rate Control Cluster

The devices involved in the high data rate transmission should carry out the data rate control procedure in order to decrease the impact on the normal operation of other devices. A general data rate control procedure is described below in several steps to provide a guideline to the implementers:

**1** A path creation command should be sent from the application source device to the application destination device.

**2** The device in the communication link should set an appropriate data rate according to the requirements of its neighbors, after it receives the path creation command, or after it sends out the path creation command if it's the source device. The detailed method should be that the device sends the Read Attributes command to its neighbors, to read their AverageLatencyRequirement, MaxLatencyRequirement or BandwidthRequirement attributes. And the neighbors should respond with the Read Attributes Response command. Those requirements attributes may be also reported actively by the neighbors using Report Attributes command. This operation may also be executed before step 1, i.e. those requirements attributes being read and stored inside the device beforehand. How to use the AverageLatencyRequirement, MaxLatencyRequirement or BandwidthRequirement attributes to calculate the appropriate data rate is left to the specific implementation.

**3** The destination device should send a Data Rate Control command to the source, after it receives the Path Creation command. Each device should compare the Data Rate field in the Data Rate Control command with the data rate it sets in step 2, and replace the Data Rate field with the data rate it sets if the former one is bigger. At last a smallest data rate will be acquired. So then an appropriate data rate for communication can be negotiated out among the devices in the communication link.

**4** A Data Rate Notification command should be sent out from the application source device to the application destination device.

**5** The devices in the communication link should control their data rate according to the negotiated data rate. The source device should set appropriate fragmentation transmission parameters according to the data rate, such as apsInterframeDelay for the APS fragmentation transmission, or InterframeDelay for the Partition Cluster. For example, APS fragmentation is used. Then the apsInterframeDelay parameter should be set appropriately according to the negotiated data rate. In this case, the average data rate to send out the bits contained in a block during the whole time which equal to the block transmission period plus the apsInterframeDelay interval, shall not bigger than the negotiated data rate. The acknowledgement bits related to the transmission may be counted in the bits to be sent in a block. The source device and the intermediate devices should update their *BandwidthRequirement* attribute according to the data rate. The destination device may tune its polling rate according to the data rate if it is an end device.

6. When the application transmission ends, the source device should send a Path Deletion command to the application destination device. Then the source device and the devices which have received the Path Deletion command should recover the related parameters such as BandwidthRequirement.

In the data rate control procedure, a communication link is identified by an originator address i.e. the address of the application source and a destination address i.e. the address of the application destination. So all the devices in the communication link are able to recognize which communication link the data rate control procedure is for.

## A.7 Voice over ZigBee Cluster

### A.7.1 Scope and Purpose

This section specifies a single cluster, the VoZ cluster, which provides commands and attributes for voice receiving and transmitting among ZigBee devices. This cluster is to provide a standardized interface for the devices to receive/transmit voice data packets.

This document should be used in conjunction with the ZigBee Cluster Library, Foundation Specification (see [R4]), which gives an overview of the library and specifies the frame formats and general commands used therein.

The cluster specified in this document is typically used for telecom applications, but may be used in any other application domains. This cluster may use Partition Cluster.

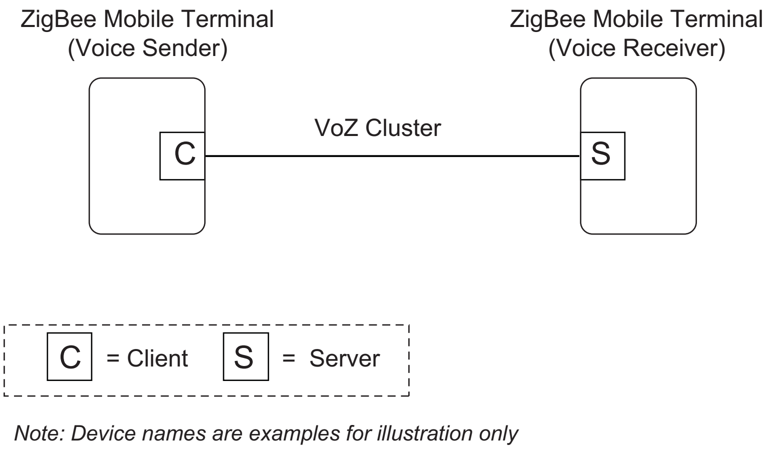

ZigBee Mobile Terminal (Voice Sender)
ZigBee Mobile Terminal (Voice Receiver)
VoZ Cluster
C
S
C = Client
S = Server
Note: Device names are examples for illustration only

**Figure A.84** Typical Usage of the VoZ Cluster

## A.7.2 Overview

This cluster provides attributes and commands for devices to receive/transmit their voice data. One of the devices plays a role of a receiver and the other does that of a sender. For example, a receiver receives voice data from the other MT (voice sender).

An important thing to notice for this VoZ cluster is that there are two different types of service for this cluster. One of them is voice transmission between humans (human-to-human). The other type of the service is voice transmission from human to device (human-to-device voice data transmission, i.e. voice command) in order to send a voice ‘command’ to a device. Therefore, the meaning of ‘voice’ delivery includes not only human voice delivery (human-to-human), but also voice delivery for device control (human-to-device, i.e. voice command). These two types of service will be referenced whenever necessary.

## A.7.3 Server

The server stores the data to be shared. It may response to the request from other devices and transmit the data to them, or it may actively request other devices to transmit the data to them.

### A.7.3.1 Dependencies

None.

### A.7.3.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table A.38.

**Table A.38 *VoZ* Attribute Sets**

| Attribute Set Identifier | Description |
|---|---|
| 0x000 | Voice Information |
| 0x001 - 0xfff | Reserved |

### A.7.3.2.1 *Establishment Information* Attribute Set

The *Establishment Information* attribute set contains the attributes summarized in Table A.39.

**Table A.39 Attributes of the *Voice Information* Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0000 | *CodecType* | 8-bit enumeration | G.711(PCM) (=0x01), G.726(ADPCM ) (=0x02), CELP(=0x03), AMR (=0x04) | Read/Write | M |
| 0x0001 | *SamplingFrequency* | 8-bit enumeration | SF_8K (=0x01), SF_7K (=0x02), SF_3_5K (=0x03) | Read/Write | M |
| 0x0002 | *Codecrate* | 8-bit enumeration | CR_64K (=0x01), CR_40K (=0x02), CR_32K (=0x03), CR_24K (=0x04), CR_16K (=0x05), CR_8K (=0x06), CR_6_3K (=0x07), CR_5_3K (=0x08), CR_AMR-NB (=0x09), CR_AMR-WB (=0x0a) | Read/Write | M |
| 0x0003 | *EstablishmentTimeout* | Uint8 | 0x01-0xff | - | M |
| 0x0004 | *CodecTypeSub1* | 8-bit enumeration | - | Read/Write | O |
| 0x0005 | *CodecTypeSub2* | 8-bit enumeration | - | Read/Write | O |
| 0x0006 | *CodecTypeSub3* | 8-bit enumeration | - | Read/Write | O |

**Table A.39 Attributes of the *Voice Information* Attribute Set (Continued)**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0007 | *Compression Type* | 8-bit enumeration | ALaw (=0x01), uLaw (=0x02) | - | O |
| 0x0008 | *Compression Rate* | 8-bit enumeration | - | - | O |
| 0x0009 | *OptionFlags* | 8-bit bitmap | 0x00-0xff | Read/Write | O |
| 0x000a | *Threshold* | Uint8 | 0x00-0xff | Read/Write | O |
| 0x000b - 0xffff | *Reserved* | N/A | N/A | N/A | N/A |

### A.7.3.2.1.1 *CodecType* Attribute

The *CodecType* attribute specifies the enumeration of the codec type. G.711 Codec is PCM (pulse code modulation) method by the ITU-T. G.726 Codec is ADPCM (adaptive differential PCM) method which has involved G.721 and G.723 ITU-T codec. CELP (code excited linear prediction) is voice codec of CDMA-based digital mobile communication system. AMR (adaptive multirate codec) is used to 3GP European mobile equipment.

### A.7.3.2.1.2 *SamplingFrequency* Attribute

The *SamplingFrequency* attribute specifies the enumeration of the sampling frequency (Hz).

PCM, ADPCM, CELP: 8KHz, AMR: 3.5KHz, 7KHz

### A.7.3.2.1.3 *CodecRate* Attribute

The *CodecRate* attribute specifies the enumeration of the codec rate (Kbps).

Various codec rates available.

PCM: 64Kbps, ADPCM: 40/32/24/16Kbps, CELP: 5.3/8Kbps, AMR: 5 ~ 12Kbps

### A.7.3.2.1.4 *EstablishmentTimeout* Attribute

The *EstablishmentTimeout* attribute sets timeout value to 1/10 sec in order to disconnect an establishment between devices when there is no response after the establishment.

#### A.7.3.2.1.5 *CodecTypeSub1*, *CodecTypeSub2*, *CodecTypeSub3* Attribute

*CodecTypeSub1*, *CodecTypeSub2*, and *CodecTypeSub3* attributes are used for additionally supportable Codecs other than the system default one. It has the same range value as that of *CodecType* attribute.

#### A.7.3.2.1.6 *Compression Type* Attribute

The *CompressionType* attribute specifies the enumeration of the compression type

**ALaw**: the compression technology for transmission data to minimize the quantification error in PCM, (Europe)

**uLaw**: the compression technology for transmission data to minimize the quantification error in PCM, (US, Japan)

#### A.7.3.2.1.7 *Compression Rate* Attribute

The *CodecRate* attribute specifies the enumeration of compression rate.

Compression rate is defined based on compression type.

#### A.7.3.2.1.8 *OptionFlags* Attribute

The *OptionFlags* attribute indicates the optional function. It shall be formatted as illustrated in Figure A.85.

| Bits: b0 | b1 | b2 | b3-b7 |
|---|---|---|---|
| Occupancy | PLC | VAD | Reserved |

**Figure A.85** Format of the *OptionFlags* Attribute

The Occupancy field specifies whether the occupancy sensor is active or not. If the Occupancy field is set to one, it indicates the occupancy sensor is active. If the Occupancy field is set to zero, it indicates the occupancy sensor is inactive.

**PLC (Packet Loss Concealment):** enabled in logic level high in order to correct voice data when there is loss for the data.

**VAD (Voice activity detection)**: enabled in logic level high in order to distinguish mute voice data from non-mute voice data.

#### A.7.3.2.1.9 *Threshold* Attribute

The *Threshold* attribute specifies the value for voice loudness in voice transmission.

### A.7.3.3 Commands Received

The received command IDs for the VoZ cluster are listed in Table A.40.

Before proceeding, please refer to sub-clause A.7.2 of the overview, and especially, to the service type that there are two types of service in this cluster. The commands in this section are developed and used not only for human-to-human voice delivery, but also for human-to-device voice delivery in order to send a voice command to control the device.

**Table A.40 Command IDs for the VoZ Cluster**

| Command Identifier Field Value | Description | Mandatory/Optional |
|---|---|---|
| 0x00 | Establishment Request | M |
| 0x01 | Voice Transmission | M |
| 0x02 | Voice Transmission Completion | O |
| 0x03 | Control Response | O |
| 0x04 - 0xff | Reserved | N/A |

### A.7.3.3.1 Establishment Request Command

The Establishment Request command is used for a device to request for a connection of the voice information from another device. It shall be originated by the voice transmission source device and sent to the transmission destination device.

The Establishment Request command shall be formatted as illustrated in Figure A.86.

For Codec Type, Sampling Frequency, Codec Rate and etc., please refer to Table A.39. For Service Type, please refer to the section of 8.7.2; the ServiceType equal to 0x00 indicates human-human service and 0x01 indicates human-device service. Besides, most commands in this section are developed and used for human-to-device communication

| Octets: | Variable | 1 | 1 | 1 | 1 | 1 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Data Type | - | 8-bit bitmap | 8-bit enumeration | 8-bit enumeration | 8-bit enumeration | 8-bit enumeration | 8-bit enumeration | 8-bit enumeration | 8-bit enumeration | 8-bit enumeration | 8-bit enumeration |
| Field Name | ZCL Header | Flag | Codec Type | Samp. Freq. | Codec Rate | Service Type | Codec TypeS 1 | Codec TypeS 2 | Codec TypeS 3 | Comp. Type | Comp. Rate |

**Figure A.86** Format of the Establishment Request Command

The Flag field value of Figure A.86 is set according to the bit values of Figure A.87 when a VoZ device has an optional attribute value such as CodecTypeSub1.

| Bits: b0 | b1 | b2 | b3 | b4-b7 |
|---|---|---|---|---|
| CodecTypeSub1 | CodecTypeSub2 | CodecTypeSub3 | Compression | Reserved |

**Figure A.87** Format of the Flag

### A.7.3.3.2 Voice Transmission Command

The Voice Transmission command is used for a device to transmit the voice data to other devices. It shall be originated by the voice transmission source device and sent to the voice transmission destination device. If required, Partition Cluster should be used for this command.

In case of transmitting multiple voice data, the Sequence Number in the ZCL Header should be sequentially increased in order to detect the loss of data and reassemble them.

| Octets: | Variable | Variable |
|---|---|---|
| Data Type | - | - |
| Field Name | ZCL Header | Voice Data |

**Figure A.88** Format of the Voice Transmission Command

### A.7.3.3.3 Voice Transmission Completion

The Voice Transmission Completion command is sent to the destination device when needed, after the source device transmits all voice data which should be transmitted.

The Voice Transmission Completion command shall be formatted as illustrated in Figure A.89.

| Octets: | Variable |
|---|---|
| Data Type | - |
| Field Name | ZCL Header |

**Figure A.89** Format of the Voice Transmission Completion Command

### A.7.3.3.4 Control Response Command

The Control Response command is used to respond with the success or failure of the control, when a device receives the Control command.

The Control Response command shall be formatted as illustrated in Figure A.90.

| Octets: | Variable | 1 |
|---|---|---|
| Data Type | - | 8-bit enumeration |
| Field Name | ZCL Header | ACK(=0x01)/ NAK(=0x00) |

**Figure A.90** Format of the Control Response Command

### A.7.3.4 Commands Generated

The generated command IDs for the VoZ cluster are listed in Table A.41.

Before proceeding, please refer to the sub-clause A.7.2 of overview, and especially, to the service type that there are two types of service in this cluster. The commands in this section are developed and used not only for human-to-human voice delivery, but also for human-to-device voice delivery in order to send a voice command to control the device.

**Table A.41 Generated Command IDs for the VoZ Cluster**

| Command Identifier Field Value | Description | Mandatory/Optional |
|---|---|---|
| 0x00 | Establishment Response | M |
| 0x01 | Voice Transmission Response | M |
| 0x02 | Control | O |
| 0x03 - 0xff | Reserved | |

#### A.7.3.4.1 Voice Transmission Response Command

The Voice Transmission Response command is to notify the sender of NACK. It shall be originated by the voice transmission destination device and sent to the voice transmission source device.

The Voice Transmission Response command shall be formatted as illustrated in Figure A.91.

| Octets: | Variable | 1 | 1 |
|---|---|---|---|
| Data Type | - | Uint8 | 8-bit enumeration |
| Field Name | ZCL Header | Sequence Number of ZCL Header | Error Flag |

**Figure A.91** Format of the Voice Transmission Response Command

If there is an error in processing the received voice data, the receiving device should respond with the Voice Transmission Response command with the sequence number of ZCL Header and the Error Flag set to an error reason according to Table A.42.

**Table A.42 The Error Flag of Voice Transmission Response**

| Error Flag Identifier Field Value | Description |
|---|---|
| 0x00 | Failure to decode voice data |
| 0x01 | Wrong order of voice data |
| 0x02 - 0xff | Reserved |

### A.7.3.4.2 Establishment Response Command

The Establishment Response command is to notify the device which previously requests for connecting the voice information. It shall be originated by the voice transmission destination device and sent to the voice transmission source device.

The Establishment Response command shall be formatted as illustrated in Figure A.92.

| Octets: | Variable | 1 | 1/0 |
|---|---|---|---|
| Data Type | - | 8-bit enumeration | 8-bit enumeration |
| Field Name | ZCL Header | ACK(=0x01)/ NAK(=0x00) | CodecType |

**Figure A.92** Format of the Establishment Response Command

When a receiving device receives the Establishment Request command with CodecType which is not supported, it responds with the Establishment Response command with NAK and CodecType supported by the device.

If the requested CodecType exists among CodecTypeSub1, CodecTypeSub2, and CodecTypeSub3, the CodecType field is set to the value.

If the device receives the Establishment Response command with NAK and CodecType, it first checks whether it supports the CodecType in the received command. If it supports, the device transmits the Establishment Request command with its CodecType again.

#### A.7.3.4.3 Control Command

The Control command is to control the voice transmission source device. It shall be originated by the voice transmission destination device and sent to the voice transmission source device.

For example, this command is used for such as walkie-talkie communication or radio listening.

The voice Control command shall be formatted as illustrated in Figure A.93.

| **Octets:** | **Variable** | **1** |
|---|---|---|
| Data Type | - | 8-bit enumeration |
| Field Name | ZCL Header | Control Type |

**Figure A.93** Format of the Control Command

The Control Type field indicates the control options, including the play operation (=0x01), the stop operation(=0x02), and the disconnection operation (=0x03). The play operation is to request for starting voice data transmission. The stop operation is to request for stopping voice data transmission. The disconnection operation is to terminate the connection between the voice transmission source/ destination devices.

### A.7.4 Client

#### A.7.4.1 Commands Received

The client receives the cluster-specific commands detailed in sub-clause A.7.3.4 as required by application profiles

#### A.7.4.2 Commands Generated

The client generates the cluster-specific commands detailed in sub-clause A.7.3.3 as required by application profiles.

## A.8 Enhancement of RSSI Location Cluster for Centralized Location

### A.8.1 Introduction

This document contains a proposal for the enhancement of the existing RSSI Location Cluster to permit the localization to be performed in a centralized way. Since this is not to be intended as a new cluster, but as enhancement of the existing one that has been designed only for the distributed localization, the text in the following chapters has to be merged with Chapter 3.13 of [R4]: for the sake of simplicity, the same chapter architecture has been followed (in order to find the correspondent paragraph of [R4] A.8.1 should be replaced by 3.13).

#### A.8.1.1 Overview

This cluster provides a means for exchanging Received Signal Strength Indication (RSSI) information among one hop devices as well as messages to report RSSI data to a centralized device that collects all the RSSI data in the network. An example of the usage of RSSI location cluster is shown in the following figure (Figure A.94).

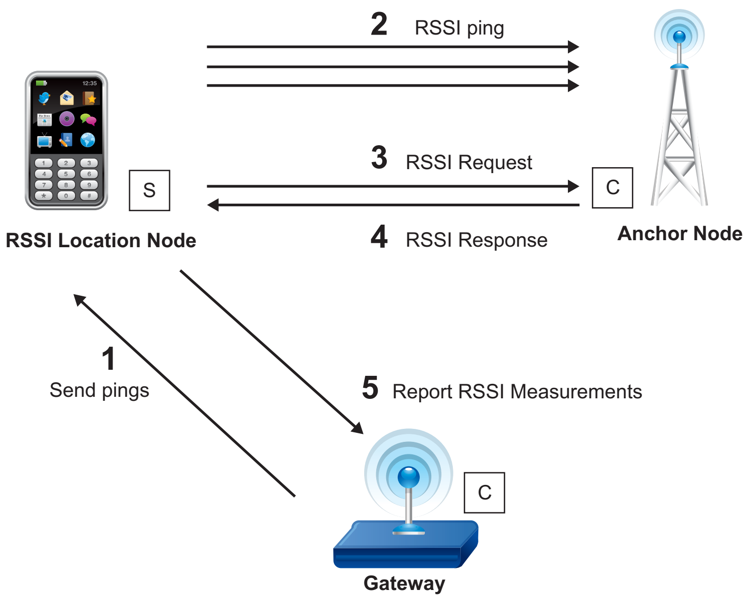


**Figure A.94** Example of Usage of RSSI Location Cluster

#### A.8.1.2 Server

##### A.8.1.2.1 Dependencies

None

##### A.8.1.2.2 Attributes

Few new attributes need to be added to be able to perform centralized location. In particular, a new *LocationType* has been added to the *Location Information* attribute set to consider centralized localization.

#### A.8.1.2.2.1 *Location Information* Attribute Set

No changes

##### A.8.1.2.2.1.1 *LocationType* Attribute

No changes.

##### A.8.1.2.2.1.2 *LocationMethod* Attribute

The "Centralized" method has been added to the *LocationMethod* attribute so that this can be considered as a method to perform localization into the ZigBee PAN.

Table A.43 replaces to Table 3.59 into the original text ([R4]). Additions and changes to the corresponding table in [R4] have been highlighted in bold.

**Table A.43 The *LocationMethod* Attribute**

| Value | Method | Description |
|---|---|---|
| 0x00 | Lateration | A method based on RSSI measurements from three or more sources. |
| 0x01 | Signposting | The location reported is the location of the neighboring device with the strongest received signal. |
| 0x02 | RF fingerprinting | RSSI signatures are collected into a database at commissioning time. The location reported is the location taken from the RSSI signature database that most closely matches the device's own RSSI signature. |
| 0x03 | Out of band | The location is obtained by accessing an out-of-band device (that is, the device providing the location is not part of the ZigBee network). |
| **0x04** | **Centralized** | **If the location is performed in a centralized way (e.g. by the GW). Different from the previous since here the device performing the localization is part of the ZigBee network.** |
| 0x05 - 0x3f | - | Reserved |
| 0x40 - 0xff | - | Reserved for manufacturer specific location methods. |

##### A.8.1.2.2.1.3 *LocationAge* Attribute

No changes.

#### A.8.1.2.2.1.4 *QualityMeasure* Attribute

No changes.

#### A.8.1.2.2.1.5 *NumberOfDevices* Attribute

No changes.

### A.8.1.2.2.2 Location Settings Attribute Set

No changes.

#### A.8.1.2.2.2.1 (up to 3.13.2.2.2.4 of [R4] have no changes)

#### A.8.1.2.2.2.2

No changes.

#### A.8.1.2.2.2.3

No changes.

#### A.8.1.2.2.2.4

No changes.

#### A.8.1.2.2.2.5 (3.13.2.2.2.5) Calculation Period

The *CalculationPeriod* attribute specifies the time in milliseconds between successive calculations of the device's location. If CalculationPeriod is less than the physically possible minimum period that the calculation can be performed, the calculation will be repeated as frequently as possible. In case of centralized location (*LocationMethod* attribute equal to centralized) the *CalculationPeriod* attribute specifies the period between successive RSSI ping commands.

#### A.8.1.2.2.2.6 (3.13.2.2.2.6 *NumberRSSIMeasurements* Attribute)

The *NumberRSSIMeasurements* attribute specifies the number of RSSI measurements to be used to generate one location estimate. The measurements are averaged to improve accuracy. *NumberRSSIMeasurements* must be greater than or equal to 1. In case of centralized location (*LocationMethod* attribute equal to centralized) the *NumberRSSIMeasurements* attribute specifies the number of successive RSSI ping commands to be sent by the server side of location cluster.

#### A.8.1.2.3 Commands Received

Two new received commands are added to the cluster. Table A.44 replaces Table 3.63 (of [R4]). Additional fields have been highlighted in bold.

**Table A.44 Commands Received**

| Command Identifier Field Value | Description | Mandatory/Optional |
|---|---|---|
| 0x00 | Set Absolute Location | M |
| 0x01 | Set device configuration | M |
| 0x02 | Get device configuration | M |
| 0x03 | Get location data | M |
| **0x04** | **RSSI Response** | **O** |
| **0x05** | **Send Pings**[a] | **O** |
| **0x06** | **Anchor node announce** | **O** |
| 0x07-0xff | Reserved | - |

**a. CCB #1053**

##### A.8.1.2.3.1 (up to 3.13.2.3.4 of [R4] have no changes but the following ones)

3.13.2.3.3.1: Figure 3.38 of [R4] should have octets equal to 8 not 1

3.13.2.3.4.1 Change line 33 into “bit 4 of the first octet “

##### A.8.1.2.3.2 RSSI Response Command

This command is sent by a device in response to a RSSI Request command.

#### A.8.1.2.3.2.1 Payload Format

The RSSI Response command shall be formatted as illustrated in Table A.45

**Table A.45 Format of the RSSI Response Command**

| Octets | 8 | 2 | 2 | 2 | 1 | 1 |
|---|---|---|---|---|---|---|
| DataType | IEEE address | Signed16-bit integer | Signed16-bit integer | Signed16-bit integer | Signed 8-bit integer | Unsigned 8-bit integer |
| FieldName | Replying Device | X | Y | Z | RSSI | NumberRSSIMeasurements |

The fields into the payload have the following meanings:

**Replying Device**

IEEE address of the neighbor that reply to the RSSI request

**X,Y,Z**

Coordinates of the replying node

**RSSI**

RSSI value registered by the replying node that refers to the radio link, expressed, in dBm between itself and the neighbor that performed the RSSI request.

**NumberRSSIMeasurements**

How many packets were considered to give the RSSI value (=1 meaning no mean is supported).

#### A.8.1.2.3.2.2 Effect on Receipt

On receipt of this commands the server side of the location cluster will wait for CalculationPeriod time and generate a Report RSSI Measurement command.

### A.8.1.2.3.3 Send Pings Command[13]

This command is used to alert a node to start sending multiple packets so that all its one hop neighbors can calculate the mean RSSI value of the radio link.

13. CCB #1053

#### A.8.1.2.3.3.1 Payload Format

Start Blasting command shall be formatted as illustrated in Table A.46. The address field contains the IEEE address of the node that have to perform the blasting (the destination node of this command) and the other fields of the payload correspond directly to the attributes with the same names. For details of their meaning and ranges see the descriptions of the individual attributes.

**Table A.46 Format of the Start Blasting Command**

| Octets | 8 | 1 | 2[a] |
|---|---|---|---|
| Data Type | IEEE Address | Unsigned 8-bit integer | Unsigned 16-bit integer |
| Field Name | Target Address | NumberRSSIMeasurements | CalculationPeriod |

a. CCB #1053

#### A.8.1.2.3.3.2 Effect on Receipt

On receipt of this command, the device shall update the attributes corresponding to (i.e. with the same names as) the payload fields and generate a number of RSSI Ping commands equal to NumberRSSIMeasurements waiting for CalculationPeriod time between successive transmission of pings.

### A.8.1.2.3.4 Anchor Node Announce Command

This command is sent by an anchor node when it joins the network, if it is already commissioned with the coordinates, to announce itself so that the central device knows the exact position of that device. This message should be either unicast to the central node or broadcast in the case that of unknown destination address.

#### A.8.1.2.3.4.1 Payload Format

Into the payload there are both the short and long addresses of the joining node as well as the coordinates of the node itself. 0xffff should be used if coordinates are not known.

**Table A.47 Anchor Node Announce Command Payload Format[a]**

| Octets | 8 | 2 | 2 | 2 |
|---|---|---|---|---|
| Data Type | IEEE Address | Signed 16-bit integer | Signed 16-bit integer | Signed 16-bit integer |
| Field Name | Anchor node IEEE address | X | Y | Z |

a. CCB #1053

### A.8.1.2.4 Commands Generated

Four new generated commands are added to the cluster. Table A.48 replaces Table 3.64 of [R4]. Additional fields have been highlighted in bold

Table A.48 Commands Generated

| Command Identifier Field Value | Description | Mandatory/Optional |
|---|---|---|
| 0x00 | Device configuration response | M |
| 0x01 | Location data response | M |
| 0x02 | Location data notification | M |
| 0x03 | Compact location data notification | M |
| 0x04 | RSSI Ping | M |
| **0x05** | **RSSI Request** | **O** |
| **0x06** | **Report RSSI Measurements** | **O** |
| **0x07** | **Request Own Location**[a] | **O** |
| 0x08-0xff | Reserved | |

a. **CCB #1053**

#### A.8.1.2.4.1 (up to 3.13.2.4.5 of [R4] have no changes)

#### A.8.1.2.4.2 RSSI Request Command

A device uses this command to ask to one, more, or all its one hop neighbors for the (mean) RSSI value they hear from itself.

##### A.8.1.2.4.2.1 Payload Format

The message is empty and may be used in broadcast (typical usage is broadcast with radius equal to one).

##### A.8.1.2.4.2.2 Effect on Receipt

On receipt of this command, the device shall respond by generating a RSSI Response command back to the sender of this request.

#### A.8.1.2.4.3 Report RSSI Measurements Command

This command is sent by a device to report its measurements of the link between itself and one or more neighbours. In a centralized location scenario, the device that sends this command is the device that needs to be localized.

#### A.8.1.2.4.3.1 Payload Format

The Report RSSI measurement command shall be formatted as illustrated in Table A.49.

**Table A.49 Format of the Report RSSI Measurement Command**[a]

| Octets | 8 | 1 | N |
|---|---|---|---|
| Data Type | IEEE Address | Unsigned 8-bit integer | Variable |
| Field Name | Measuring Device | N Neighbors | Neighbors Info |

a. CCB #1053

Neighbor info structure is reported in Table A.50[14].

**Table A.50 Neighbour Info Structure**

| Octets: | 8 | 2 | 2 | 2 | 1 | 1 |
|---|---|---|---|---|---|---|
| DataType | IEEE Address | Signed16-bit integer | Signed16-bit integer | Signed16-bit integer | Signed8-bit integer | Unsigned8-bit integer |
| FieldName | Neighbour | X | Y | Z | RSSI | NumberRSSIMeasurements |

The fields into the payload have the following meanings:

**N Neighbors**

Numbers of one hop neighbors that reported the RSSI. This field indicates how many Neighbors Info fields are present in the message.

**Measuring Device**

IEEE address of the device that report the measurements (i.e. the one that started the blast procedure)

**Neighbors Information:**

**X,Y,Z**

Coordinates (if present) of the neighbor

14. CCB #1053

**Neighbor**

IEEE address of the neighbor used to identify it if coordinates are either not present or not valid

**RSSI**

RSSI value registered by the neighbor that refer to the radio link between itself and measuring device

**NumberRSSIMeasurements**

How many packets were considered to give the RSSI value (=1 meaning is that no mean is supported)

##### A.8.1.2.4.4 Request Own Location Command

This command is sent by a node wishing to know its own location and it is sent to the device that performs the centralized localization algorithm.

###### A.8.1.2.4.4.1 Payload Format

The Request Own Location command payload shall be formatted as illustrated in Table A.51. The only field in the payload contains the IEEE address of the bile node, i.e. the node that wishes to know about its own location.

**Table A.51 Request Own Location Command Payload Format**

| Octets: | 8 |
|---|---|
| DataType | IEEE Address |
| FieldName | IEEE address of the blind node |

###### A.8.1.2.4.4.2 Effect on Receipt

The node receiving the Request Own Location command will then reply with a Set Absolute Location command, telling the requesting entity its location.

### A.8.1.3 Client

#### A.8.1.3.1 Dependencies

None

#### A.8.1.3.2 Attributes

None

#### A.8.1.3.3 Commands Received

The client receives the cluster-specific commands generated by the server (see A.8.1.2.4).

#### A.8.1.3.4 Commands Generated

The client generates the cluster-specific commands received by the server (see A.8.1.2.3), as required by application.

## A.9 Gaming Cluster

### A.9.1 Introduction

This cluster provides attributes and commands to support gaming functions of ZigBee-enabled mobile terminals.

#### A.9.1.1 Scope and Purpose

This document specifies a single cluster, the mobile gaming cluster, which provides commands and attributes to support gaming function for ZigBee-capable mobile terminals. This cluster is to provide a standardized interface for users to play networked games using ZigBee mobile terminals, such as cell phones, PDAs and ZigBee-enabled game consoles.

This document should be used in conjunction with the ZigBee Cluster Library, Foundation Specification, which gives an overview of the library and specifies the frame formats and general commands used therein.

### A.9.2 General Description

#### A.9.2.1 Introduction

The cluster specified in this document defines the attributes and commands needed to support network gaming functions provided by ZigBee.

A network game play usually starts with one device announcing the willingness to play and the games available for play in its vicinity. This requires the device implement the Information cluster as described in A.2 and be an Information Node (IN). Other devices which receive the announcements can join the announcing device for a game play. Short chatting messages can be exchanged by using the Chatting cluster defined in A.10 before a game starts. This allows the players to exchange information, such as, how much time they have for a play before the flight takes off.

The announcing device usually becomes the Master Device of the game and other devices the Slave Devices. The Master Device is responsible for controlling the membership of the game, such as joining and leaving the game. Control messages will be unicast, broadcast or multicast to other players in the game. A game usually starts when all players hit the start button. In a two-player game, any player can stop playing and end the game if he/she desires. In a multi-player game, individual player can quit at anytime without ending the game until the required minimum number of players is not satisfied.

The typical procedure of a mobile game play is shown in Figure A.95. Optional steps are shown in dashed boxes.

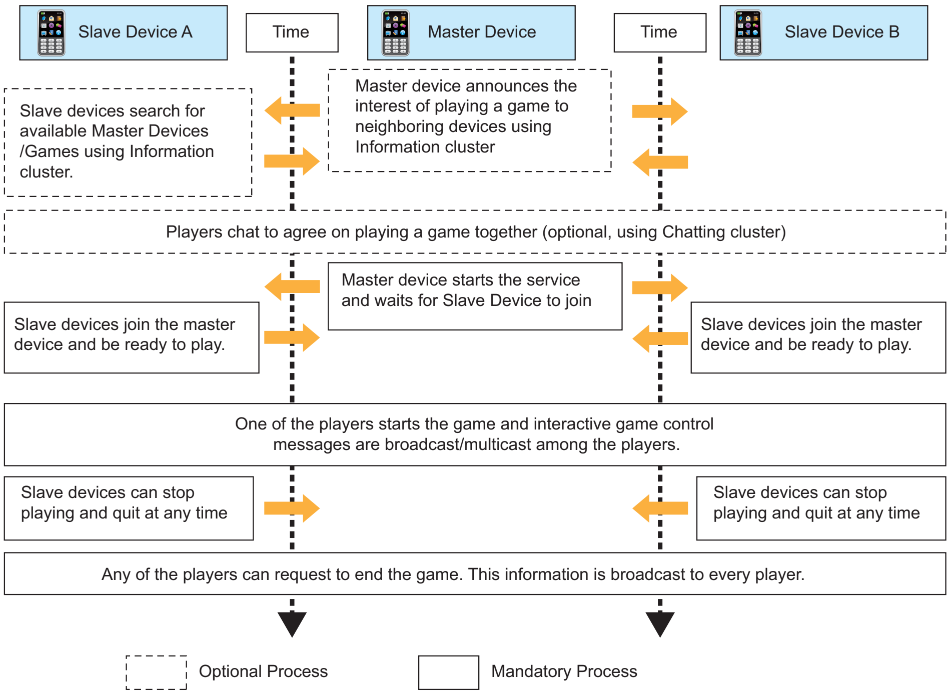

Slave Device A
Time
Master Device
Time
Slave Device B
Slave devices search for available Master Devices /Games using Information cluster.
Master device announces the interest of playing a game to neighboring devices using Information cluster
Players chat to agree on playing a game together (optional, using Chatting cluster)
Master device starts the service and waits for Slave Device to join
Slave devices join the master device and be ready to play.
Slave devices join the master device and be ready to play.
One of the players starts the game and interactive game control messages are broadcast/multicast among the players.
Slave devices can stop playing and quit at any time
Slave devices can stop playing and quit at any time
Any of the players can request to end the game. This information is broadcast to every player.
Optional Process
Mandatory Process

**Figure A.95** Typical Procedure of a Mobile Game Play

### A.9.2.2 Cluster List

This functional domain defines the mobile gaming cluster which provides the attributes and commands required to support mobile game playing using ZigBee terminals. The list of clusters is listed in Table A.52.

**Table A.52** **Clusters Specified for the Mobile Gaming Functional Domain**

| Cluster Name | Description |
|---|---|
| Mobile Gaming | Attributes and commands to support mobile game playing using ZigBee terminals. |

It is worthwhile to note although there are two kinds of devices, the Master Device and the Slave Device, all these devices are actually running in a peer-to-peer mode in the network. In the client/server point of view, all devices have to be both client and server, despite the fact that the Master Device may provide more services than the Slave Devices. The services provided by the Slave Devices include responding to game announcements, responding to inquiries from other devices, processing action control commands and etc. An example of client/server relationship of mobile gaming cluster with 3 players is illustrated in Figure A.96

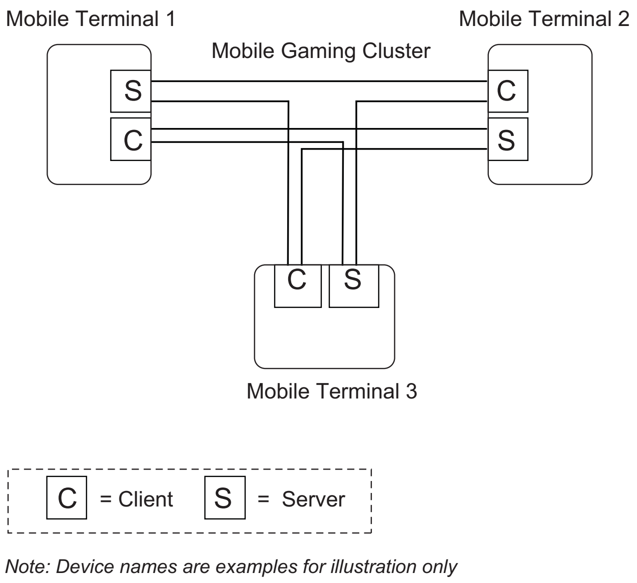


**Figure A.96** Typical Usage of the Mobile Gaming Cluster

## A.9.3 Server

Both Master Device and Slave Device function as both server and client simultaneously. More server functions are accomplished by Master Device than by Slave Devices. This document will not specify exactly what functions shall be implemented by the Master Device and the Slave Device. The decision is left to the application and may differ from one game to another.

### A.9.3.1 Dependencies

This cluster does not depend on any other existing clusters. However, in order to successfully play a network game, the following clusters may also need to be implemented in the same device where the mobile gaming cluster is implemented.

- Basic cluster and Groups cluster as described in [R4]
- Information cluster as described in A.2
- Chatting cluster as described in A.10

### A.9.3.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table A.53.

**Table A.53** ***Mobile Gaming*** **Attribute Sets**

| Attribute Set Identifier | Description |
|---|---|
| 0x000 | *General Information* Attribute Set |
| 0x001-0x002 | *Game* Attribute Set |
| 0x003- 0xfff | Reserved |

#### A.9.3.2.1 *General Information* Attribute Set

The *General Information* attribute set contains the attributes summarized in Table A.54.

Table A.54 Attributes of the *General Information* Attribute Set

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0000 | *PlayerName* | Character String | | ReadOnly | M |
| 0x0001 | *NbOfGames* | Unsigned 8-bit Integer | 0x00 -0xfe | ReadOnly | M |
| 0x0002 | *ListOfGames* | Character String | | ReadOnly | M |
| 0x0003 | *Announcement Interval* | Unsigned 16-bit Integer | 0x0000 - 0xffff | ReadOnly | M |
| 0x0004-0x000f | Reserved | | | | |

##### A.9.3.2.1.1 *PlayerName* Attribute

The *PlayerName* attribute specifies the name of the player in character string. This is typically the name of the device owner. It is used to identify the players when multiple players are in the game.

##### A.9.3.2.1.2 *NbOfGames* Attribute

The *NbOfGames* attribute indicates the number of games this device is able to play. This number should be equal to the number of entries of the *Detailed Game Information* Attribute Set in the table of *Mobile Gaming* attribute sets (see Table A.52). The maximum number of games supported in this version of document is 254.

##### A.9.3.2.1.3 *ListOfGames* Attribute

The *ListOfGames* attribute lists the available games this device is able to play. In the list, each game's *GameID* is separated by semicolon (;). Note the data type is Character String so the implementer should not try to interpret *GameID* as integer. The entire list shall end with a full stop (.). Therefore, the format of the list is as below.

"ID #1; ID #2; …; ID #n."

An example of the *ListOfGames* string is shown below.

"3; 7; …; n."

The clients that receive this list shall be able to parse the list. By matching the GameID to the available game list announced by the Information cluster, the client shall be able to show the list of games and their IDs correctly on the displays of the mobile terminals for the player to choose from.

#### A.9.3.2.1.4 *AnnouncementInterval* Attribute

The *AnnouncementInterval* attribute specifies the duration of the interval, in the unit of seconds, between two consecutive game announcements. A value equal to zero is not allowed.

### A.9.3.2.2 *Game* Attribute Set

The *Game* Attribute Set contains the attributes of the game which is currently in play or soliciting players. It is summarized in Table A.55. In order to allow flexibility in the game attributes, an alternative approach is described in A.9.5.

**Table A.55 Attributes of the *Game* Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0010 | *GameID* | Unsigned 16-bit Integer | 0x0001-0x00fe | ReadOnly | M |
| 0x0011 | *NameOfGame* | Character String | | ReadOnly | M |
| 0x0012 | *GameMaster* | Boolean | | ReadOnly | M |
| 0x0013 | *Status* | 8-bit Bitmap | 0x00-0xff | ReadOnly | M |
| 0x0014 | *CurrentNbOfPlayers* | Unsigned 8-bit Integer | 0x00-0xff | ReadOnly | M |
| 0x0015 | *ListOfCurrentPlayers* | Character String | | ReadOnly | M |
| 0x0016 | *MaxNbOfPlayers* | Unsigned 8-bit Integer | 0x00-0xff | ReadOnly | M |
| 0x0017 | *MinNbOfPlayers* | Unsigned 8-bit Integer | 0x00-0xff | ReadOnly | M |
| 0x0018 | *CurrentGameLevel* | Unsigned 16-bit Integer | 0x0000-0xffff | ReadOnly | M |
| 0x0019 | *ScoreOfThisPlayer* | Unsigned 16-bit Integer | 0x0000-0xffff | ReadOnly | M |
| 0x001a | *Timer1* | Unsigned 16-bit Integer | 0x0000-0xffff | ReadOnly | M |

**Table A.55 Attributes of the *Game* Attribute Set (Continued)**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x001b | *Timer2* | Unsigned 16-bit Integer | 0x0000-0xffff | ReadOnly | M |
| 0x001c | *Timer3* | Unsigned 16-bit Integer | 0x0000-0xffff | ReadOnly | M |
| 0x001d | *Counter1* | Unsigned 16-bit Integer | 0x0000-0xffff | ReadOnly | M |
| 0x001e | *Counter2* | Unsigned 16-bit Integer | 0x0000-0xffff | ReadOnly | M |
| 0x001f | *Downloadable* | Boolean | 0-1 | ReadOnly | O |
| 0x0020-0x002f | Reserved | - | - | - | - |

#### A.9.3.2.2.1 *GameID* Attribute

The *GameID* attribute specifies the ID of the game which is currently being announced for play or is in play.

#### A.9.3.2.2.2 *NameOfGame* Attribute

The *NameOfGame* attribute specifies the name of the game which is currently being announced for play or is in play.

#### A.9.3.2.2.3 *GameMaster* Attribute

The *GameMaster* attribute specifies whether this device is able to, or willing to, be the Master Device of the game which is currently being announced for play or is playing. In most cases, the devices which send the announcements will have this attribute set to TRUE to indicate that they will be the Master Device of the game. However, if the attribute is set to FALSE, this device is only announcing the interest of playing a game but is not able to be the Master Device of the game.

#### A.9.3.2.2.4 *Status* Attribute

The *Status* attribute specifies the status of the game which is currently being played or accepting new players. The bitmap is arranged in the following format (Figure A.97). The bits set to “1”mean TRUE, while if set to “0” mean FALSE.

| Bit 0 | Bit 1 | Bit 2 | Bit 3 | Bit 4 | Bit 5-7 |
|---|---|---|---|---|---|
| Announcing and joining | In Play | In Pause | New Player Allowed | All Players Ready | Reserved |

**Figure A.97** The *Status* Attribute

#### A.9.3.2.2.5 *CurrentNbOfPlayers* Attribute

The *CurrentNbOfPlayers* attribute specifies the number of players in the game who are currently playing or the number of players who have joined the game.

#### A.9.3.2.2.6 *ListOfCurrentPlayers* Attribute

The *ListOfCurrentPlayers* attribute provides a list of the players in the game who are currently playing or the list of players who have joined the game. It follows the format specified in sub-clause A.9.3.2.1.3.

#### A.9.3.2.2.7 *MaxNbOfPlayers* Attribute

The *MaxNbOfPlayers* attribute specifies the maximum number of players the game can accommodate. The game cannot start if the actual number of players has exceeded this value. For example, if a poker game can take at most 4 people to play, the value of this attribute will be set to 4. *MaxNbOfPlayers* must be >= *MinNbofPlayers* attribute.

#### A.9.3.2.2.8 *MinNbOfPlayers* Attribute

The *MinNbOfPlayers* attribute specifies the minimum number of players required for playing the game. The game cannot start if this number has not been reached. For example, if a poker game requires at least 4 people to play, the value of this attribute will be set to 4.

#### A.9.3.2.2.9 *CurrentGameLevel* Attribute

The *CurrentGameLevel* attribute specifies the difficulty level of the game which is currently being played.

#### A.9.3.2.2.10 *ScoreOfThisPlayer* Attribute

The *ScoreOfThisPlayer* attribute specifies the score of this player in the game which is currently being played.

###### A.9.3.2.2.11 *Timer1* Attribute

The *Timer1* attribute specifies the value of the first timer of the game which is currently being played. The applications will determine the meaning of the timer.

###### A.9.3.2.2.12 *Timer2* Attribute

The *Timer2* attribute specifies the value of the second timer of the game which is currently being played. The applications will determine the meaning of the timer.

###### A.9.3.2.2.13 *Timer3* Attribute

The *Timer3* attribute specifies the value of the third timer of the game which is currently being played. The applications will determine the meaning of the timer.

###### A.9.3.2.2.14 *Counter1* Attribute

The *Counter1* attribute specifies the value of the first counter of the game which is currently being played. The applications will determine the meaning of the counter.

###### A.9.3.2.2.15 *Counter2* Attribute

The *Counter2* attribute specifies the value of the second counter of the game which is currently being played. The applications will determine the meaning of the counter.

###### A.9.3.2.2.16 *Downloadable* Attribute

The *Downloadable* attribute specifies whether the game can be downloaded by the device announcing it. The download of the game may be performed using out-of-band mechanism (e.g. Bluetooth).

### A.9.3.3 Commands Received

The server may receive the following commands for the mobile gaming cluster (Table A.56)

**Table A.56 Command IDs for the Mobile Gaming Cluster**

| Command Identifier Field Value | Description | Mandatory/Optional |
|---|---|---|
| 0x00 | Search game | M |
| 0x01 | Join game | M |
| 0x02 | Start game | M |
| 0x03 | Pause game | M |
| 0x04 | Resume game | M |
| 0x05 | Quit game | M |
| 0x06 | End game | M |
| 0x07 | Start over | M |
| 0x08 | Action control | M |
| 0x09 | Download Game | O |
| 0x0a - 0xff | Reserved | - |

#### A.9.3.3.1 Search Game Command

The Search Game command is used for a device to search for a specific game for play or a list of games that are currently available from one or more of its neighbouring devices. It shall be originated by the device that wants to search the game and sent to other devices in its neighbourhood.

The Search Game command shall be formatted as illustrated in Figure A.98.

| Octets: | Variable | 1 | 0/2 |
|---|---|---|---|
| Data type | - | 8-bit enumeration | Unsigned 16-bit Integer |
| Field name | ZCL Header | Specific Game | GameID |

**Figure A.98** Payload Format of the Search Game Command

The Specific Game field is of type 8-bit enumeration and takes only two values, 0x00 and 0x01. Value 0x00 indicates a list of available games is requested while value 0x01 indicates only the game specified in the GameID field of this command is solicited. The GameID field is of type Unsigned 16-bit Integer. When this command is received, the device shall search its *General Information* Attribute Set for the list of games or the availability of the specific game.

#### A.9.3.3.2 Join Game Command

The Join Game command is used for a device to join another device for playing a specific game. It shall be originated by the device that wants to join the game and sent to the device which announced the game.

The Join Game command shall be formatted as illustrated in Figure A.99.

| **Octets:** | **Variable** | **2** | **1** | **Variable** |
|---|---|---|---|---|
| Data type | - | Unsigned 16-bit Integer | Boolean | Character string |
| Field name | ZCL Header | Game ID | Join as Master | Name of Game |

**Figure A.99** Payload Format of the Join Game Command

The Game ID field is of type Unsigned 16-bit Integer and indicates the ID of the game this device is joining. The Join as Master field is of type Boolean. The value of TRUE indicates this device wishes to be the Master Device of the game, even though it is other device that announced the game. This usually happens when the announcing device indicates that it is not willing to be the Master Device in its Game Announcement command. The Name of Game indicates the name of the game. The Name of Game field is of type Character Strings.

#### A.9.3.3.3 Start Game Command

The Start Game command is used for a device to inform other devices that it is ready for a play. In most cases this happens when the start button on the mobile terminal is hit. The game will begin when all players have informed the rest of the players their readiness. This command contains no payload and shall be originated by the device that wants to start the game and sent to all other devices in the game.

#### A.9.3.3.4 Pause Game Command

The Pause Game command is used for a device to inform other devices that it has temporarily suspended a game in play. In most cases this happens when the pause button on the mobile terminal is hit. The game may be suspended until the Resume Game command is received. This command contains no payload and shall be originated by the device that wants to pause and sent to all other devices in the game.

#### A.9.3.3.5 Resume Game Command

The Resume Game command is used for a device to inform other devices that it wants to continue the game from the point where it was paused. In most cases this happens when the resume button on the mobile terminal is hit. If more than one player is in the pause state, the game will resume when all players have sent the Resume Game command. This command contains no payload and shall be originated by the devices that want to resume the game and sent to all other devices in the game.

#### A.9.3.3.6 Quit Game Command

The Quit Game command is used for a device to inform other devices that it wants to quit itself from a game which is in play. In most cases this happens when the quit button on the mobile terminal is hit. When there are only two players in the game, the function of Quit Game is equivalent to End Game. For games with more than two players, one or more players exiting from the game will not stop the game. The game will have to stop when the minimum requirement of the number of players cannot be satisfied. This command contains no payload and shall be originated by the device that wants to quit itself from the game and sent to all other devices in the game.

#### A.9.3.3.7 End Game Command

The End Game command is used for a device to inform other devices that it wants to end the game which is in play. No matter the number of players in the game, no one can play any more once it is ended. In most cases this happens when the end game button on the mobile terminal is hit. Depending on the authority of the sender, this command may or may not be executed (i.e. the master unit may send it). This command contains no payload and shall be originated by the device that wants to end the game and sent to all other devices in the game.

#### A.9.3.3.8 Start Over Command

The Start Over command is used for a device to inform other devices that it wants to stop the current progress of a game and start from the very beginning of the game. In most cases this happens when the start over button on the mobile terminal is hit. Depending on the authority of the sender, this command may or may not be executed. This command contains no payload and shall be originated by the device that wants to start over the game and sent to all other devices in the game.

#### A.9.3.3.9 Action Control Command

The Action Control command is used for a device to inform other devices the actions it just took in the game in play. In most cases this happens when one or more function buttons or keys on the mobile terminal is/are hit. This command shall be originated by the device that just took the actions and sent to all other devices in the game.

The Action Control command shall be formatted as illustrated in Figure A.100.

| Octets: | Variable | 4 |
|---|---|---|
| Data type | - | 32-bit bitmap |
| Field name | ZCL Header | Actions |

**Figure A.100** Payload Format of the Action Control Command

The Actions field is a 32-bit bitmap and its meaning is illustrated in Table A.57.

**Table A.57 Bit Assignment of the Actions Field**

| Bit | Meaning |
|---|---|
| $Bit_0$ | Number Key “0” Pressed |
| $Bit_1$ | Number Key “1” Pressed |
| $Bit_2$ | Number Key “2” Pressed |
| $Bit_3$ | Number Key “3” Pressed |
| $Bit_4$ | Number Key “4” Pressed |
| $Bit_5$ | Number Key “5” Pressed |
| $Bit_6$ | Number Key “6” Pressed |
| $Bit_7$ | Number Key “7” Pressed |
| $Bit_8$ | Number Key “8” Pressed |
| $Bit_9$ | Number Key “9” Pressed |
| $Bit_{10}$ | “*” Key Pressed |
| $Bit_{11}$ | “#” Key Pressed |
| $Bit_{12}$ | Function Button #1 Pressed |
| $Bit_{13}$ | Function Button #2 Pressed |
| $Bit_{14}$ | “BACK” Button Pressed |
| $Bit_{15}$ | “CLEAR” Button Pressed |
| $Bit_{16}$ | “VOLUME UP” Button Pressed |
| $Bit_{17}$ | “VOLUME DOWN” Button Pressed |
| $Bit_{18}$ | Joystick Moved Up |
| $Bit_{19}$ | Joystick Moved Down |
| $Bit_{20}$ | Joystick Moved Left |

**Table A.57 Bit Assignment of the Actions Field (Continued)**

| | |
|---|---|
| $Bit_{21}$ | Joystick Moved Right |
| $Bit_{22}$ | Joystick Pressed |
| $Bit_{23}$ -$Bit_{31}$ | Reserved |

Note it is possible that multiple bits are set in an action field. For example, when both $Bit_{12}$ and $Bit_1$ are presented, it means the player had just pressed Function Button #1 and the number key "1" at the same time.

#### A.9.3.3.10 Download Game Command

The Download Game command is used for a device to request a download of the game program from the device which announces it. Whether the game is available for download is indicated in the detailed game information attribute set (Downloadable field). This command contains no payload and shall be originated by the device that requests the download and sent to the device which announced the game.

### A.9.3.4 Commands Generated

The generated commands for the mobile gaming cluster are listed in Table A.58

**Table A.58 Generated Command IDs for the Mobile Gaming Cluster**

| Command Identifier Field Value | Description | Mandatory/Optional |
|---|---|---|
| 0x00 | Game announcement | M |
| 0x01 | General response | M |
| 0x02 - 0xff | Reserved | - |

#### A.9.3.4.1 Game Announcement Command

The Game Announcement command is used for a device to announce to other devices its willingness to play a specific game or to provide a list of games it is able to play. The command can be sent in two modes, the active mode and the passive mode. Under the active mode, the announcements are sent periodically without other devices requesting them. The announcement interval can be obtained from the *AnnouncementInterval* attribute of the general information attribute set (Table A.53). Under the passive mode, the announcements are sent as the responses to the search game commands received from other devices. It shall be originated by the device that announces the game and sent to either the device which is searching for the game (the passive mode) or all other devices in its neighborhood (the active mode).

The Game Announcement command shall be formatted as illustrated in Figure A.101.

| Octets: | Variable | 2 | 1 | Variable |
|---|---|---|---|---|
| Data type | - | Unsigned 16-bit Integer | Boolean | Character string |
| Field name | ZCL Header | GameID | Game Master | List of Game |

**Figure A.101** Payload Format of the Game Announcement Command

The GameID field is of type Unsigned 16-bit Integer and indicates the game is being solicited. The Game Master field is of type Boolean. The value of TRUE indicates this device wishes to be the Master Device of the game. The List of Game field is also of type Character String and indicates a list of available games (represented by GameID) this device is able to play. In case the receivers are not interested in the game the sender is proposing, they can pick a game in the List of Game and suggest to the sender (by using the Search Game command) to play that one instead. If the sending device gets enough interests of playing the game it solicited, it may ignore the suggestion from the receivers. Otherwise, the sending device can accept the suggestion by announcing again with the name of the suggested game in its GameID field. In this case, the device which made the suggestion should be given preference to join the game.

### A.9.3.4.2 General Response Command

The General Response command is used for a device to respond to the commands it received from other devices. It shall be originated by the device that is responding and sent to either a specific device or all other devices in the network.

The General Response command shall be formatted as illustrated in Figure A.102.

| Octets: | Variable | 1 | 1 | Variable |
|---|---|---|---|---|
| Data type | - | Unsigned 8-bit Integer | 8-bit bitmap | Character string |
| Field name | ZCL Header | Command ID | Status | Message |

**Figure A.102** Payload Format of the General Response Command

The Command ID field is of type Unsigned 8-bit Integer and indicates the ID of the command to which this command is responding. The Status field is an 8-bit bitmap that indicating the status of the reception and execution of the command received. The Message field is of type Character Strings and gives detailed explanation of the status, especially when the status is Failure.

**Table A.59 Meanings of the Status Field**

| Bit | Meaning |
|---|---|
| $Bit_0$ | Success |
| $Bit_1$ | Failure |
| $Bit_2$ | Command not recognized |
| $Bit_3$ -$Bit_7$ | Reserved |

## A.9.4 Client

### A.9.4.1 Attributes

The client has no attributes.

### A.9.4.2 Commands Received

The client receives the cluster-specific commands detailed in sub-clause A.9.3.4.

### A.9.4.3 Commands Generated

The client generates the cluster-specific commands detailed in sub-clause A.9.3.3, as required by application.

## A.9.5 Announcing Available Games with Information Cluster

### A.9.5.1 Detailed Game Information

The detailed game information is distributed by the Information cluster. There are two ways a player can obtain the game information from other devices/players. If a device discovers any Information Nodes around it, it can search the contents carried by the information nodes and find the desired game to play. Once the device selects a specific game, it can issue a Search Game and then a Join Game command to participate in the game. In this process, GameID is the unique identifier of a game and it should maintain the same in both the information cluster and the game cluster.

In the second approach, a device can passively listen to the Game Announcement commands from its neighbor nodes. Once it receives an announcement command (which provides only GameID but no name of the game), it can find the detailed game information from the Information cluster from the same neighbor node. If the device decides to participate in the game, it shall send a Join Game command to the neighbor node.

The detailed game information carried by the Information Node or by another terminal supporting the Gaming and Information cluster is illustrated in Table A.59. The attributes of detailed game information should be mapped in the Information cluster content as it follows (Figure A.103):

- GameID should correspond to the ContentID of information content with title equal to “GameID” string; if multiple games are supported the Root Content (i.e. the content carried by ContentRootID) should have a title equal to “MultipleGames” string;
- The attributes of detailed game information may be mapped as children content of the GameID; in this case the Title of the contents corresponds to the Name field indicated in Table A.59, while the attribute value is mapped into the content data type octet string or character string.

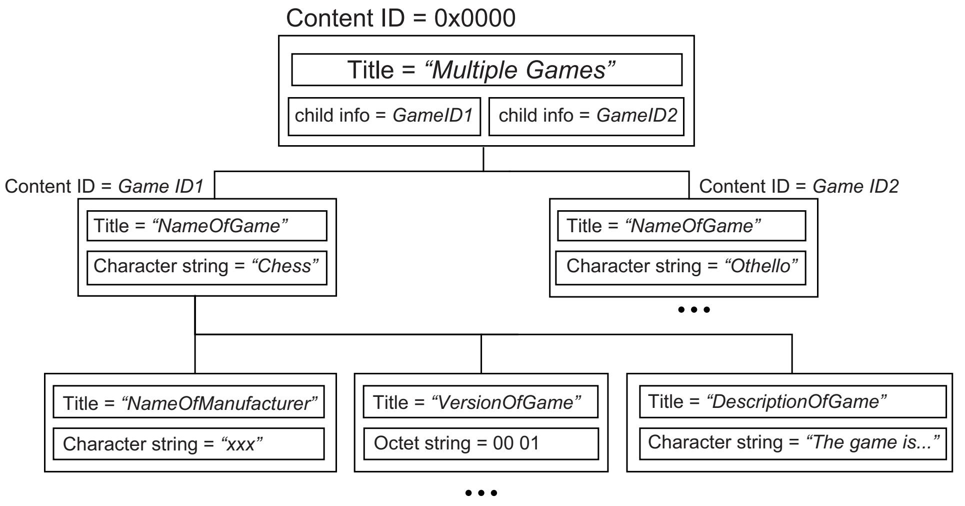

Content ID = 0x0000
Title = *"Multiple Games"*
child info = *GameID1*
child info = *GameID2*
Content ID = *Game ID1*
Content ID = *Game ID2*
Title = *"NameOfGame"*
Character string = *"Chess"*
Title = *"NameOfGame"*
Character string = *"Othello"*
Title = *"NameOfManufacturer"*
Character string = *"xxx"*
Title = *"VersionOfGame"*
Octet string = 00 01
Title = *"DescriptionOfGame"*
Character string = *"The game is..."*

**Figure A.103** Mapping of Detailed Game Information in the Information Content Carried by the Information Cluster

**Table A.60 Attributes of Detailed Game Information**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0000 | *GameID* | Unsigned 16-bit Integer | 0x0001-0x00fe | ReadOnly | M |
| 0x0001 | *NameOfGame* | Character Strings | | ReadOnly | M |
| 0x0002 | *NameOfManufacturer* | Character Strings | | ReadOnly | M |
| 0x0003 | *VersionOfGame* | Octet string | 0x00-0xff | ReadOnly | M |
| 0x0004 | *DescriptionOfGame* | Character Strings | | ReadOnly | M |
| 0x0005 | *MaxNbOfPlayers* | Octet string | 0x00-0xff | ReadOnly | M |
| 0x0006 | *MinNbOfPlayers* | Octet string | 0x00-0xff | ReadOnly | M |
| 0x0007 | *TotalDifficultyLevels* | Octet string | 0x00-0xff | ReadOnly | M |
| 0x0008 | *LevelOfThisPlayer* | Octet string | 0x00-0xff | ReadOnly | M |
| 0x0009 | *MinCPUSpeed* | Octet string | 0x0000-0xffff | ReadOnly | O |
| 0x000a | *MinMemorySize* | Octet string | 0x0000-0xffff | ReadOnly | O |

**Table A.60 Attributes of Detailed Game Information (Continued)**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x000b | *MinDisplayResolution Horizontal* | Octet string | 0x0000-0xffff | ReadOnly | O |
| 0x000c | *MinDisplayResolution Vertical* | Octet string | 0x0000-0xffff | ReadOnly | O |
| 0x000d | *MinNbOfColors* | Octet string | 0x0000-0xffff | ReadOnly | O |
| 0x000e | *GameDownloadable* | Boolean | TRUE/ FALSE | ReadOnly | O |
| 0x000f | Reserved | - | - | - | - |

### A.9.5.2 *GameID* Attribute

The *GameID* attribute specifies the ID of the game.

### A.9.5.3 *NameOfGame* Attribute

The *NameOfGame* attribute specifies the name of the game.

### A.9.5.4 *NameOfManufacturer* Attribute

The *NameOfManufacturer* attribute specifies the manufacturer of the game.

### A.9.5.5 *VersionOfGame* Attribute

The *VersionOfGame* attribute specifies the version of the game.

### A.9.5.6 *DiscriptionOfGame* Attribute

The *DescriptionOfGame* attribute contains brief description of the game.

### A.9.5.7 *MaxNbOfPlayers* Attribute

The *MaxNbOfPlayers* attribute specifies the maximum number of players the game can accommodate. The game cannot start if the number of players exceeds this value. For example, if a poker game can take at most 4 people to play, the value of this attribute will be set to 4. *MaxNbOfPlayers* must be >= *MinNbofPlayers* attribute.

#### A.9.5.8 *MinNbOfPlayers* Attribute

The *MinNbOfPlayers* attribute specifies the minimum number of players required to play the game. The game cannot start if this number is not reached. For example, if a poker game requires at least 4 people to play, the value of this attribute will be set to 4.

#### A.9.5.9 *TotalDifficultyLevels* Attribute

The *TotalDifficultyLevels* attribute specifies the total difficulty levels of the game.

#### A.9.5.10 *LevelOfThisPlayer* Attribute

The *LevelOfThisPlayer* attribute specifies the highest level this player has been reached in the game.

#### A.9.5.11 *MinCPUSpeed* Attribute

The *MinCPUSpeed* attribute specifies the minimum requirement for the CPU speed, in the unit of MHz, in order to play the game.

#### A.9.5.12 *MinMemorySize* Attribute

The *MinMemorySize* attribute specifies the minimum requirement for the user accessible memory size, in the unit of KBytes, in order to play the game.

#### A.9.5.13 *MinDisplayResolutionHorizontal* Attribute

The *MinDisplayResolutionHorizontal* attribute specifies the minimum requirement for the horizontal display resolution, in the unit of pixels, in order to play the game.

#### A.9.5.14 *MinDisplayResolutionVertical* Attribute

The *MinDisplayResolutionVertical* attribute specifies the minimum requirement for the vertical display resolution, in the unit of pixels, in order to play the game.

#### A.9.5.15 *MinNbOfColors* Attribute

The *MinNbOfColors* attribute specifies the minimum requirement for the number of colors the device screen can display in order to play the game.

#### A.9.5.16 *GameDownloadable* Attribute

The *GameDownloadable* attribute indicates whether the announced game can be downloaded from the device which announces it if other players do not have it installed.

# A.10 Chatting Cluster

## A.10.1 Introduction

### A.10.1.1 Scope and Purpose

This document specifies a single cluster, the Chatting cluster, which provides commands and attributes for sending chat messages among ZigBee devices. This cluster is to provide a standardized interface for people using ZigBee devices to chat with each other like they using instant messaging applications through Internet. The transaction sequence numbers used in the ZCL command frames for the Chatting cluster should be the same for the requests and responses; the default responses should use also the same transaction sequence numbers of the related commands in order to match the correspondent packets[15].

There are two kinds of chatting scenarios:

**1** Centralized Server

In this kind of scenario a centralized server is used for managing and controlling the messaging between the different ZigBee nodes. Different chat sessions can be made available by the server. The node entering the ZigBee network may search for the available chat sessions and join one of them after choosing one out of different available sessions. Different nodes can join one chat session and can interact with each other.

**2** Ad Hoc Chat Sessions

In this kind of scenario no infrastructure is needed. Any ZigBee node can start and manage a chat session. A ZigBee node in a particular ZigBee network can start a chat session. A node should only be a chairman of one chat session, i.e. it should only start one chat session. It is recommended to do so, since in the ad hoc scenario, the chairman can be any devices which may have low computing power and capability, and maintaining more than one session may be difficult for the devices. To start a chat session it has to decide a unique identifier for the chat session. This identifier shall be unique among all the chat sessions in the networks. For this requirement the implementer shall make it mandatory for a node to select a chat identifier which will be unique in the whole ZigBee network. The identifier may be set the same as the address of the device that starts the session, so as to guarantee its uniqueness.

This document should be used in conjunction with the ZigBee Cluster Library, Foundation Specification (see [R4]), which gives an overview of the library and specifies the frame formats and general commands used therein.

15. CCB #1114

The cluster specified in this document is typically used for telecom applications, but may be used in any other application domains. This cluster may use Partition Cluster. And this cluster may be used in conjunction with Billing cluster [R7], e.g. the operator may need to charge for the chat.

This cluster provides attributes and commands for devices to send chatting messages to each other.

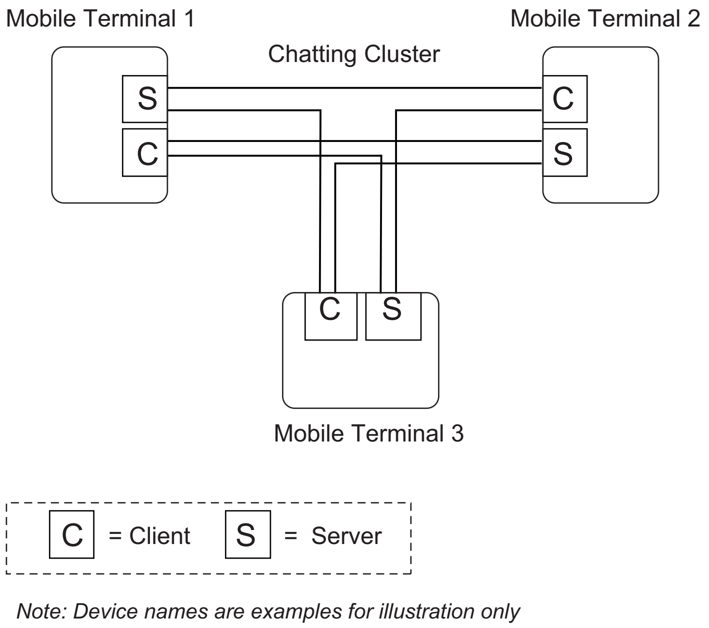


**Figure A.104** Typical Usage of the Chatting Cluster

## A.10.2 Server

The server manages the list of participants, the chat session ID, etc. It can respond to the devices which are going to join chat sessions, or it can ask some one to leave the chat session. The server has the functions which are more related to managing the session than sending chatting messages.

The messages in the chat sessions are usually sent by the multicast method. So the server should also manage the chat group. There is an example which can be a guideline for implementing. Once the server forms a new chat session, it should form a new group. The group ID may be the same as the chat session ID. If a new user joins the chat session, it should also join the chat group. To fulfill this, the server should use the Add Group command specified in groups cluster [R4] to add the newcomer to the chat group. And if a user leaves the chat session, it should also leave the chat group. To fulfill this, the server should use the Remove Group command specified in groups cluster [R4] to remove the user from the chat group.

### A.10.2.1 Dependencies

This cluster does not depend on any other existing clusters. However, in order to successfully fulfill the chatting, Information cluster, Groups cluster and Billing cluster may also need to be implemented in the same device where the chatting cluster is implemented.

### A.10.2.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table A.61.

**Table A.61** ***Chatting*** **Attributes Sets**

| Attribute Set Identifier | Description |
|---|---|
| 0x000 | User related |
| 0x001 | Chat Session Related |
| 0x002 - 0xfff | Reserved |

#### A.10.2.2.1 *User Related* Attribute Set

The *User Related* attribute set contains the attributes summarized in Table A.62.

**Table A.62 Attributes of the *User Related* Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0000 | U_ID | Unsigned 16-bit integer | 0x0000-0xffff | ReadOnly | M |
| 0x0001 | Nickname | Character string | | ReadOnly | M |
| 0x0002-0x000f | Reserved | | | | |

##### A.10.2.2.1.1 *U_ID* Attribute

The *U_ID* attribute is the unique identification of the user in the chat room. It may be same as the address given to the device while joining in the ZigBee network, or may be same as the 2 least significant bytes of UserID of Billing cluster [R7]. The value 0xffff means this attribute is not set.

##### A.10.2.2.1.2 *Nickname* Attribute

The *Nickname* attribute is a unique display name of the user identified by the U_ID while talking in the public chat room. User sets the *Nickname* while joining the chat room.

#### A.10.2.2.2 *Chat Session Related* Attribute Set

The *Chat Session Related* set contains the attributes summarized in Table A.63.

**Table A.63 Attributes of the *Chat Session Related* Attribute Set**

| Identifier | Name | Type | Range | Access | Mandatory/ Optional |
|---|---|---|---|---|---|
| 0x0010 | C_ID | Unsigned 16-bit integer | 0x0000-0xffff | ReadOnly | M |
| 0x0011 | Name | Character string | | ReadOnly | M |
| 0x0012 | EnableAddChat | Boolean | TRUE/ FALSE | ReadOnly | O |
| 0x0013-0x001f | Reserved | | | | |

##### A.10.2.2.2.1 *C_ID* Attribute

The *C_ID* attribute is the unique identification of a chat room. It is assigned by the chat server while creating a new chat room following the user command or chosen by the chairman. It may be same as the chat group ID. If the server maintains several chat rooms, this attribute should be set as the ID of the latest formed chat room. The value 0xffff means this attribute is not set.

##### A.10.2.2.2.2 *Name* Attribute

The *Name* attribute is the name or topic of the chat room which is identified by the *C_ID* attribute.

##### A.10.2.2.2.3 *EnableAddChat* Attribute

The *EnableAddChat* attribute indicates whether the server permits other users to add new chat rooms in it. TRUE (0x01) indicates the server permit other users to add new chat rooms, while FALSE (0x00) indicates not permit to do so.

### A.10.2.3 Commands Received

The received commands IDs for the chatting cluster are listed in Table A.64.

**Table A.64 Command IDs for the Chatting Cluster**

| Command Identifier Field Value | Description | Mandatory/Optional |
|---|---|---|
| 0x00 | Join Chat Request | M |
| 0x01 | Leave Chat Request | M |
| 0x02 | Search Chat Request | M |
| 0x03 | Switch Chairman Response | O |
| 0x04 | Start Chat Request | O |
| 0x05 | ChatMessage | M |
| 0x06 | Get Node Information Request | O |
| 0x07 - 0xff | Reserved | |

#### A.10.2.3.1 Join Chat Request Command

The Join Chat Request command is used for a node to request to join one chatting session.

The Join Chat request command shall be formatted as illustrated in Figure A.105.

| Octets: | Variable | 2 | Variable | 2 |
|---|---|---|---|---|
| Data Type | - | Unsigned 16-bit integer | Character string | Unsigned 16-bit integer |
| Field Name | ZCL Header | U_ID | Nickname | C_ID |

**Figure A.105** Format of the Join Chat Request Command

The U_ID field indicates unique identification of the user in the chat room.

The Nickname field is type of character string which is a unique display name of the user while talking in the public chat room.

The C_ID field is unique identification of a chat room. It indicates the ID of the chat room which the client wants to join.

This command should be unicast to the server which manages the chat room indicated by the C_ID.

### A.10.2.3.2 Leave Chat Request Command

The Leave Chat Request command is used for a node to request to leave one chatting session. The client may require the Default Response command to be sent from the server so that to confirm the leave command has been successfully received.

The Leave Chat request command shall be formatted as illustrated in Figure A.106.

| Octets: | Variable | 2 | 2 |
|---|---|---|---|
| Data Type | - | Unsigned 16-bit integer | Unsigned 16-bit integer |
| Field Name | ZCL Header | C_ID | U_ID |

**Figure A.106** Format of the Leave Chat Request Command

The C_ID field indicates the ID of the chat room which the node wants to leave. The U_ID field indicates unique identification of the user in the chat room.

This command should be unicast to the server which manages the chat room the user to leave.

### A.10.2.3.3 Search Chat Request Command

The Search Chat Request command is used for a node to request to search for the available chat session on the server.

The Search Chat Request command shall contain no payload and shall be originated by the devices which want to have a chat with others and sent to the server. It may be broadcast in the network.

### A.10.2.3.4 Switch Chairman Response Command

The Switch Chairman Response command is used for nodes to response to the Switch Chairman Request command.

The Switch Chairman Response command shall be formatted as illustrated in Figure A.107.

| Octets: | Variable | 2 | 2 |
|---|---|---|---|
| Data Type | - | Unsigned 16-bit integer | Unsigned 16-bit integer |
| Field Name | ZCL Header | C_ID | U_ID |

**Figure A.107** Format of the Switch Chairman Response Command

The C_ID field in the command indicates the ID of the chat room of which the receiving node is the old chairman. The U_ID field in the command is the unique ID of node which wants to be the chairman of the chat room.

This command shall be unicast to the chairman, announcing that the node indicated by the U_ID volunteers to be the new chairman.

#### A.10.2.3.5 Start Chat Request Command

The Start Chat Request command is used for a device to request to create one chat session. The new chat session to be created shall be managed by the responder. That is, once the chat session is created, the responder but not the requester will be the chairman of the chat room.

The Start Chat request command shall be formatted as illustrated in Figure A.108.

| Octets: | Variable | Variable | 2 | Variable |
|---|---|---|---|---|
| Data Type | - | Character string | Unsigned 16-bit integer | Character string |
| Field Name | ZCL Header | Name | U_ID | Nickname |

**Figure A.108** Format of the Start Chat Request Command

The Name field indicates the topic of the chat room. The U_ID field indicates unique identification of the user in the chat room. The Nickname field indicates the Nickname set by the requester.

The command is originated by the devices which want to create one chat room and attract others who have the same interest in the topic. It should be unicast to the server which manages the chat rooms.

#### A.10.2.3.6 ChatMessage Command

The ChatMessage command is used for chatting, i.e. one node to send a message to other nodes. In the case of peer chatting, such as to exchange some private messages, it may be unicast to the chairman first, the chairman may forward this command with unicast method to the destination user. The ChatMessage command may be sent directly to the destination node in peer chatting case if the destination network address and endpoint are known by the sender. Get Node Information Request and Response commands shall be used to acquire the necessary network address and endpoint information. In the case of normal chatting (a message to be sent to the whole room), all nodes in the same chat room are expected to receive the message. The ChatMessage command should be multicast to other nodes in the chat room.

In peer chatting case, if the command contains illegal parameter such as non-existing U_ID field, the chairman should return a Default Response command with INVALID_FIELD status.

The ChatMessage command shall be formatted as illustrated in Figure A.109.

| Octets: | Variable | 2 | 2 | 2 | Variable | Variable |
|---|---|---|---|---|---|---|
| Data Type | - | Unsigned 16-bit integer | Unsigned 16-bit integer | Unsigned 16-bit integer | Character string | Character string |
| Field Name | ZCL Header | Destination U_ID | Source U_ID | C_ID | Nickname | Message |

**Figure A.109** Format of the ChatMessage Command

The Destination U_ID field indicates the destination node's U_ID. The Source U_ID field indicates the source node's U_ID. The C_ID indicates the ID of the chat room which the sender belongs to. The Nickname field indicates the sender's Nickname, which shall be in Character string data type.

In the case of peer chatting, the Destination U_ID field and the Source U_ID field shall be set to the specific nodes' U_ID. In the case of normal chatting (sending a message to all the users in the chat room), Destination U_ID shall be set to 0xffff while Source U_ID shall be set to the specific source node's U_ID.

### A.10.2.3.7 Get Node Information Request Command

The Get Node Information Request command is used for peer chatting to get the network address and endpoint number of the peer node, in order to use ChatMessage command to send private message to the node. When one wants to send private massage to another node in the same chatting session, it shall check whether it has that node's network address and endpoint number. If not, it shall send this command to the server.

The Get Node Information Request command shall be formatted as illustrated in Figure A.110.

| Octets: | Variable | 2 | 2 |
|---|---|---|---|
| Data Type | - | Unsigned 16-bit integer | Unsigned 16-bit integer |
| Field Name | ZCL Header | C_ID | U_ID |

**Figure A.110** Format of the Get Node Information Request Command

The C_ID field indicates the ID of the chat room which the investigated node belongs to. The U_ID field indicates the U_ID of the node to be investigated.

This command should be unicast to the chairman node. A chatting table should be maintained by the chairman. It may be also maintained by other nodes. When a chairman has assigned a U_ID to a node it shall add related information into the chatting table, and when a node leaves the chatting session it shall remove the record of the leaving device from the table. A node may get the address of another node from the chairman by using the Get Node Information Request command. Once it gets the information, the node may store it for future usage. The detail format of the chatting table is implementer dependent. An example of each item of the table may be illustrated as Figure A.111. The node number field indicates the number of NodeInformation field. The NodeInformation field is as specified in section Figure A.119.

| C_ID | Node number | NodeInformation 1 | … | NodeInformation n |
|---|---|---|---|---|

**Figure A.111** Format of an Item of the Chatting Table

### A.10.2.4 Commands Generated

The generated commands IDs for the Chatting cluster are listed in Table A.65.

**Table A.65 Generated Command IDs for the Chatting Cluster**

| Command Identifier Field Value | Description | Mandatory/ Optional |
|---|---|---|
| 0x00 | Start Chat Response | O |
| 0x01 | Join Chat Response | M |
| 0x02 | User left | M |
| 0x03 | User Joined | M |
| 0x04 | Search Chat Response | M |
| 0x05 | Switch Chairman Request | O |
| 0x06 | Switch Chairman Confirm | O |
| 0x07 | Switch Chairman Notification | O |
| 0x08 | Get Node Information Response | O |
| 0x09 - 0xff | Reserved | |

#### A.10.2.4.1 Start Chat Response Command

The Start Chat Response command is used for server to response to the Start Chat request command. If successful, the server shall then form a new chat room and make itself the chairman.

The Start Chat Response command shall be formatted as illustrated in Figure A.112.

| **Octets:** | **Variable** | **1** | **0/2** |
|---|---|---|---|
| Data Type | - | 8-bit enumeration | Unsigned 16-bit integer |
| Field Name | ZCL Header | Status | C_ID |

**Figure A.112** Format of the Start Chat Response Command

The Status field indicates the status of the previous request. Its values shall be those as specified in [R4]. If it is SUCCESS, the C_ID field shall exist, or else the C_ID field shall not exist. The C_ID field indicates the unique identification of the chat room and it is assigned by the server. If the server doesn't permit to add a new chat room, i.e. the attribute EnableAddChat being set to FALSE, the server shall return this command with FAILURE Status.

This command shall be unicast to the requester.

#### A.10.2.4.2 Join Chat Response[16] Command

The Join Chat Response command is used for server to respond to the Join Chat Request command.

The Join Chat Response command shall be formatted as illustrated in Figure A.113.

| **Octets:** | **Variable** | **1** | **2** | **0/2** | **Variable** | **Variable** | **0/2** | **Variable** |
|---|---|---|---|---|---|---|---|---|
| Data Type | - | 8-bit enumeration | Unsigned 16-bit integer | Unsigned 16-bit integer | Character string | - | Unsigned 16-bit integer | Character string |
| Field Name | ZCL Header | Status | C_ID | U_ID 1 | Nickname 1 | … | U_ID n | Nickname n |

**Figure A.113** Format of the Join Chat Response Command

The Status field indicates the status of the previous request. Its values shall be those as specified in [R4]. If it is SUCCESS, the list of the U_ID and Nickname fields shall exist, or else the list shall not exist. The C_ID field indicates the ID of the chat room which the server manages. It shall be the same as the C_ID field in the corresponding Join Chat Request command. The list of the U_ID and Nickname fields indicate other participants in the chat room. Each U_ID field and the Nickname field respectively indicate the unique ID and the nickname of each user in the chat room.

16. CCB #1113

This command shall be unicast to the requester. After receiving this command, the node should check whether it has received the Add Group command from the chairman. If not, it should wait for that command so as to know which group it belongs to. How long it should wait for the command is specific to the implementation.

#### A.10.2.4.3 User Left Command

The User Left command is used for server to inform other participants that some one has left the chat room.

The User Left command shall be formatted as illustrated in Figure A.114.

| **Octets:** | **Variable** | **2** | **2** | **Variable** |
|---|---|---|---|---|
| Data Type | - | Unsigned 16-bit integer | Unsigned 16-bit integer | Character string |
| Field Name | ZCL Header | C_ID | U_ID | Nickname |

**Figure A.114**Format of the User Left Command

The C_ID indicates the ID of the chat room which the user left. The U_ID field indicates the left participant's unique ID in the chat room. The Nickname field is the nickname of the left participant.

The command shall be multicast to all users in the same chat room.

#### A.10.2.4.4 User Joined Command

The User Joined command is used for server to inform other participants that some one has just joined the chat room.

The User Joined command shall be formatted as illustrated in Figure A.115.

| **Octets:** | **Variable** | **2** | **2** | **Variable** |
|---|---|---|---|---|
| Data Type | - | Unsigned 16-bit integer | Unsigned 16-bit integer | Character string |
| Field Name | ZCL Header | C_ID | U_ID | Nickname |

**Figure A.115**Format of the User Joined Command

The C_ID indicates the ID of the chat room which the newcomer joined. The U_ID field indicates the newcomer's unique ID in the chat room. The Nickname field is the nickname of the newcomer.

This command should be multicast to all users in the same chat room. When the newcomer receives the command, it shall compare the U_ID field in the command with U_ID of itself, if same, it shall ignore the command.

### A.10.2.4.5 Search Chat Response Command

The Search Chat Response command is used for server to respond to the Search Chat Request command.

The Search Chat Response command shall be formatted as illustrated in Figure A.116.

| **Octets:** | **Variable** | **1** | **0/2** | **Variabl e** | **Variab le** | **0/2** | **Variabl e** |
|---|---|---|---|---|---|---|---|
| Data Type | - | 8-bit bitmap | Unsigned 16-bit integer | Character string | … | Unsigned 16-bit integer | Character string |
| Field Name | ZCL Header | Options | C_ID 1 | Name 1 | … | C_ID n | Name n |

**Figure A.116** Format of the Search Chat Response Command

The Options field indicates the options of this command. Bit 0 of the Options field indicates whether the server permits other users to add new chat rooms in it. The value 0b0 means permit while 0b1 means not permit. Other bits of this field are reserved. The list of the C_ID and Name fields indicates the information of the available chat rooms. Each C_ID field and Name field indicate respectively the chat room identification and topic of each available chat room. It's recommended at least one chat room should be maintained by the server. If no chat room is maintained, the list of C_ID and Name fields shall not exist.

This command should be unicast to the requester. Only the chairman can send out this command, and the list of chat room information shall only contain the information of the chat rooms which it manages. The server may also broadcast this command to notify other users which chat rooms it manages. After receiving this command, the network address and endpoint number should be extracted from the network layer header and APS header, so as to acquiring the chairman's network address and endpoint number.

### A.10.2.4.6 Switch Chairman Request Command

In the case of ad-hoc chat, when a chairman wants to leave the chat session, he can use this command to appoint a new chairman out of the participating Devices which can continue to manage the chat room.

The Switch Chairman Request command shall be multicast to every device which is in the same chat room. It shall be formatted as illustrated inFigure A.117

| **Octets:** | **Variable** | **2** |
|---|---|---|
| Data Type | - | Unsigned 16-bit integer |
| Field Name | ZCL Header | C_ID |

**Figure A.117** Format of the Switch Chairman Request Command

The C_ID field indicates the ID of the chat room where the chairman is requested to be changed.

### A.10.2.4.7 Switch Chairman Confirm Command

The Switch Chairman Confirm command is used by the old chairman to inform the node which the chairman has selected to be the new chairman.

The Switch Chairman Confirm command shall be formatted as illustrated in Figure A.118.

| **Octets:** | **Variable** | **2** | **Variable** | **Variable** | **Variable** |
|---|---|---|---|---|---|
| Data Type | - | Unsigned 16-bit integer | - | … | - |
| Field Name | ZCL Header | C_ID | NodeInformation 1 | … | NodeInformation n |

**Figure A.118** Format of the Switch Chairman Confirm Command

The C_ID field indicates the ID of the chat room which the chairman manages. The NodeInformation field is formatted as illustrated in Figure A.119. Each NodeInformation field contains information about a node participating in the chat session. This field shall contain the following sub-fields, the U_ID sub-field, Address sub-field, Endpoint sub-field and the Nickname sub-field. The U_ID sub-field, Address sub-field, Endpoint sub-field and Nickname sub-field indicate the node's unique ID, network address, endpoint number and nickname respectively. This command shall be unicast to the new chairman.

| **Octets:** | **2** | **2** | **1** | **Variable** |
|---|---|---|---|---|
| Data Type | Unsigned 16-bit integer | 16-bit data | Unsigned 8-bit integer | Character string |
| Sub-field Name | U_ID | Address | Endpoint | Nickname |

**Figure A.119** Format of the NodeInformation Field

### A.10.2.4.8 Switch Chairman Notification Command

The Switch Chairman Notification command is used by the old chairman to inform other participants in the chat room about the change in the chairman. The Switch Chairman Confirm command shall be formatted as illustrated in Figure A.120.

| Octets: | Variable | 2 | 2 | 2 | 1 |
|---|---|---|---|---|---|
| Data Type | - | Unsigned 16-bit integer | Unsigned 16-bit integer | 16-bit data | Unsigned 8-bit integer |
| Field Name | ZCL Header | C_ID | U_ID | Address | Endpoint |

**Figure A.120** Format of the Switch Chairman Notification Command

The C_ID field is the ID of the chat room which the chairman manages. The U_ID field is the unique ID of the node which is the new chairman of the chat room. The Address field and the Endpoint field are the network address and the endpoint number of the new chairman respectively. The command should be multicast to other nodes in the same chat room.

### A.10.2.4.9 Get Node Information Response Command

The Get Node Information Response command is used by the server to give a response to the Get Node Information Request command, so that the requesting node can obtain the desired information including network address and endpoint number of a specific node. If successful, the server shall provide the node information in the response command.

The Get Node Information Response command shall be formatted as illustrated in Figure A.121.

| Octets: | Variable | 1 | 2 | 2 | 0/2 | 0/1[a] | Variable |
|---|---|---|---|---|---|---|---|
| Data Type | - | 8-bit enumeration | Unsigned 16-bit integer | Unsigned 16-bit integer | 16-bit data | Unsigned 8-bit integer | Character string |
| Field Name | ZCL Header | Status | C_ID | U_ID | Address | Endpoint | Nickname |

a. CCB #1095

**Figure A.121** Format of the Get Node Information Response Command

The Status indicates the status of the previous request. Its values shall be those as specified in [R4]. If it is SUCCESS, the Address field, the Endpoint field and the Nickname field shall exist, or else those fields shall not exist. The C_ID field and the U_ID field shall be the same as the corresponding Get Node Information Request command. The C_ID field is the ID of the chat room which the investigated node belongs to. The U_ID field is the unique ID of the investigated node. The Address field, the Endpoint field and the Nickname field indicate the network address, the endpoint number and the nickname of the investigated node respectively. The command shall be unicast to requester. After receiving this command, the node may store the information of the investigated node for future usage.

### A.10.3 Client

#### A.10.3.1 Commands Received

The client receives the cluster-specific commands detailed in sub-clause A.10.2.4 as required by application profiles.

#### A.10.3.2 Commands Generated

The client generates the cluster-specific commands detailed in sub-clause A.10.2.3 as required by application profiles.

## A.11 ISO 7816 Protocol Tunnel

### A.11.1 Scope and Purpose

This document specifies a single cluster, the ISO7816 Tunnel cluster, which provides commands and attributes for mobile office solutions including ZigBee devices.

This cluster is to provide a standardized interface to enable a scenario of authorization management on mobile office devices (e.g. access to PC resources)

This document should be used in conjunction with the ZigBee Cluster Library, Foundation Specification (see [R4]), which gives an overview of the library and specifies the frame formats and general commands used therein.

## A.11.2 Definitions

The definitions used in the ISO 7816 Protocol Tunnel has been inserted in Table A.66.

**Table A.66 Definitions Used in ISO7816 Protocol Tunnel Description**

| Term | Definition |
|---|---|
| Target Device | A Computer System on which User has to perform authentication in order to access to information services |
| User Token | A device used by a Target Device to authenticate and authorize User |
| ZigBee-enabled Virtual SmartCard | A SmartCard equipped with a ZigBee node |

## A.11.3 General Description

The cluster specified in this document is typically used for telecom applications, but may be used in any other application domains.

## A.11.4 Overview[17]

This cluster provides attributes and commands to tunnel ISO7816 APDUs, enabling solution such as Mobile Office, i.e a mechanism to authenticate and authorize Users on shared Computer System (said Target Device) by means of a ZigBee-enabled Virtual Smartcard (generically said User Token).

A Target Device, enabled by the server side of this cluster, and a User Token (supporting a client side of this cluster) can establish a connection and exchange information by means of ISO7816 APDU messages over ZigBee network.

## A.11.5 Server

### A.11.5.1 Dependencies

Since ISO7816 protocol may use APDU frames larger than typical ZigBee payload, stack fragmentation or Partition Cluster shall be supported by the devices supporting this cluster.

17. CCB #1148

### A.11.5.2 Attributes

The ISO7816 Tunnel cluster contains the attribute shown in Table A.67.

**Table A.67 Attributes for the ISO7816 Tunnel Cluster**

| Identifier | Name | Type | Range | Access | Default | Mandatory/ Optional |
|---|---|---|---|---|---|---|
| 0x0001 | Status | Unsigned 8-bit integer | 0x00-0x01 | Read only | 0x00 | M |

#### A.11.5.2.1 *Status* Attribute

The *Status* attribute specifies the Server internal state.

Values and usage of this attribute are application dependent, e.g. server busy (client connected). Server supports only one client connection at a time.

The Status values are shown in Table A.68

**Table A.68 Status Values**

| Meaning | Values |
|---|---|
| 0x00 | FREE |
| 0x01 | BUSY |
| 0x02-0xFF | Reserved |

### A.11.5.3 Commands Received

The cluster-specific commands received by the ISO7816 Tunnel server cluster are listed in Table A.69.

**Table A.69 Received Command IDs for the ISO7816 Tunnel Cluster**

| Command Identifier Field Value | Description | Mandatory/ Optional |
|---|---|---|
| 0x00 | Transfer APDU | M |
| 0x01 | Insert SmartCard | M |
| 0x02 | Extract SmartCard | M |
| 0x03 - 0xff | Reserved | |

### A.11.5.3.1 Transfer APDU Command

#### A.11.5.3.1.1 Payload Format

The Transfer APDU command shall be formatted as illustrated in Figure A.122.

| **Bits** | Variable |
|---|---|
| **Data Type** | Octet String |
| **Field Name** | APDU |

**Figure A.122** Format of the Transfer APDU Command

#### A.11.5.3.1.2 APDU Field

The APDU field is of variable length and is a ISO7816 APDU as defined in the ISO7816 standard [R5].

#### A.11.5.3.1.3 When Generated

This command is generated when an ISO7816 APDU has to be transferred across a ZigBee tunnel.

#### A.11.5.3.1.4 Effect on Receipt

On receipt of this command, a device shall process the ISO7816 APDU as specified in the ISO7816 standard [R19].

### A.11.5.3.2 Insert Smart Card

#### A.11.5.3.2.1 Payload Format

No payload is needed for the Insert Smart Card command.

#### A.11.5.3.2.2 When Generated

This command is generated when a User Token insertion has to be sent to Server.

#### A.11.5.3.2.3 Effect on Receipt

On receipt of this command:

- if the *Status* attribute is equal to BUSY the Server shall send a Default Response with status FAILURE.
- if the *Status* attribute is equal to FREE and the bit ‘Disable Default Response’ of the Frame control field of the ZCL Header is set to zero, the Server shall send respond with status SUCCESS. It also shall set its Status Attributes to BUSY and it can start to exchange APDUs with Client over ISO7816 Tunnel.

#### A.11.5.3.3 Extract Smart Card

##### A.11.5.3.3.1 Payload Format

No payload is needed for the Insert Smart Card command.

##### A.11.5.3.3.2 When Generated

This command is generated when a User Token extraction has to be sent to Server.

##### A.11.5.3.3.3 Effect on Receipt

On receipt of this command:

- if the *Status* attribute is equal to FREE, the Server shall send a Default Response with status FAILURE.
- if the *Status* attribute is equal to BUSY and the bit 'Disable Default Response' of the of the Frame control field of the ZCL Header is set to zero, the Server shall send respond with status SUCCESS. It shall also set its Status Attributes to FREE and after this, Server shall not be able to exchange APDUs with Client over ISO7816 Tunnel.

### A.11.5.4 Commands Generated

The cluster-specific commands generated by the ISO7816 Tunnel server cluster are listed in Table A.70

**Table A.70 Generated Command IDs for the ISO7816 Tunnel Cluster**

| Command Identifier Field Value | Description | Mandatory/Optional |
|---|---|---|
| 0x00 | Transfer APDU | M |
| 0x01 – 0xff | Reserved | |

### A.11.5.5 Transfer APDU

#### A.11.5.5.1 Payload Format

The Transfer APDU command shall be formatted using the same command "Transfer APDU" in paragraph sub-clause A.11.5.3.1. The effect on receipt is the same as reported in sub-clause A.11.5.3.1.4.

## A.11.6 Client

### A.11.6.1 Dependencies

None.

### A.11.6.2 Attributes

The client cluster has no attributes.

### A.11.6.3 Commands Received

The client receives the cluster-specific commands detailed in sub-clause A.11.5.4 as required by application profiles

### A.11.6.4 Commands Generated

The client generates the cluster-specific commands detailed in sub-clause A.11.5.3 as required by application profiles.

**This page intentionally blank**