ZigBee™ Alliance

ZigBee Document 095264r18

ZigBee Over-the-Air Upgrading Cluster

Revision 18

Version 1.0

March 14, 2010

Sponsored by:
ZigBee Alliance

Accepted for release by:
This document has not yet been accepted for release by the ZigBee Alliance Board of Directors.

Abstract:
This is an implementation specification requirements document that describes the ZigBee Over The Air (OTA) image (FW) Upgrade. The document is to provide a standard for Over The Air message format for upgrading image along with necessary commands and parameters to ensure interoperability of Over The Air upgrading. Certain security aspects will also be enforced or recommended to protect the network from any vulnerability that may be exposed from the Over The Air upgrading.

Keywords:
ZigBee, Over The Air (OTA), Upgrade Server, Upgrade Client, digital signature, certificate

Elements of ZigBee Alliance specifications may be subject to third party intellectual property rights, including without limitation, patent, copyright or trademark rights (such a third party may or may not be a member of ZigBee). ZigBee is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

This document and the information contained herein are provided on an "AS IS" basis and ZigBee DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO (A) ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OF THIRD PARTIES (INCLUDING WITHOUT LIMITATION ANY INTELLECTUAL PROPERTY RIGHTS INCLUDING PATENT, COPYRIGHT OR TRADEMARK RIGHTS) OR (B) ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT. IN NO EVENT WILL ZIGBEE BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OF DATA, INTERRUPTION OF BUSINESS, OR FOR ANY OTHER DIRECT, INDIRECT, SPECIAL OR EXEMPLARY, INCIDENTIAL, PUNITIVE OR CONSEQUENTIAL DAMAGES OF ANY KIND, IN CONTRACT OR IN TORT, IN CONNECTION WITH THIS DOCUMENT OR THE INFORMATION CONTAINED HEREIN, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE. All Company, brand and product names may be trademarks that are the sole property of their respective owners.

The above notice and this paragraph must be included on all copies of this document that are made.

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

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:

Rob Alexander
Ember Corporation
25 Thomson Place
Boston, MA 02210
Email: rob.alexander@ember.com
Phone: +1.617.951.1244

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.

Participants

The following is a list of those who were members of the ZigBee Alliance Firmware OTA Task Group leadership when this document was released:

Jack McPeck: *Chair*

Skip Ashton: *Vice-Chair*

Wally Barnum: Co-*Vice-Chair*

Michael Cowan: *Technical Editor*

The following members made specific contributions to this document:

Jack McPeck
Skip Ashton
Don Sturek
Dan Lohman
Wally Barnum
Areeya Vimayangkoon
Rob Alexander
Colby Gore
Don Sturek
John Mani
Jens Klostergaard Lyngsoe
Jeff Blevins

Table of Contents

List of Figures

List of Tables

Change history

Table 1 - Document revision change history shows the change history for this specification.

Table 1 - Document revision change history

Revision	Version	Description
00	0.1	Initial draft incorporating ideas from Ember (Areeya Vimayangkoon and Rob Alexander)
01	0.1	Incorporated comments from Colby Gore, Don Sturek, Daniel Lohman, John Mani, Wally Barnum, Jens Klostergaard Lyngsoe, Benno Ritter, Jeff Blevins, Rob Alexander and Areeya Vimayangkoon
02	0.1	Incorporated comments from 095373r03ZB OTA Upgrades TRD LB comments and more individual comments from HA work group, Daniel Lohman, Rob Alexander and Areeya Vimayangkoon
03	0.1	Incorporated comments from Rob Alexander, Dan Lohman, and Wally Barnum.
04	0.1	Incorporated comments from Dan Lohman, Michael Cowan, Wally Barnum and Rob Alexander (as of 11/18/09): usage of disable default response, image notify command modification, new QuerySpecificFile request and response commands, change data type for UTC time and signature
05	0.1	Accepted changes from revision 03 and 04
06	0.1	Incorporated comments from Dan Lohman, Rob Alexandra regarding: byte ordering of OTA header data, clarify QueryNextImageResponse payload, adding bit control for hw version in OTA header, use octet instead of octet string data type for image data and add block size to imageBlockResponse command. Add assigned cluster id for OTA cluster of 0x0018
07	0.1	Incorporated comments from SE 1.1 test event. There are several important changes made to the specification from testing results and several discussions during the test event. Changes have impacted almost all cluster commands as well as OTA header. There will also be a new OTA cluster specific status; replacing the use of ZCL status. Specific description of possible error cases for each command along with recommended or mandated actions that need to be taken.
12	0.7	Incorporated comments from 0.7 letter ballot.
13	0.7	Incorporated comments from re-ballot.
14	0.9	Addition of sections describing ECDSA signature

		verification and calculation.
15	0.9	Added descriptions of Manufacturer ID and image type ID
16	0.9	Updated for CCBs 1314 and 1315.

1 Introduction

1.1 Purpose

The objective of this document is to provide detailed technical requirements for Over The Air image upgrade. This document captures the TRD resolution analysis and presents a clear methodology for implementation of the OTA Upgrade cluster using the existing ZigBee stack(s), ZigBee Cluster Library and this OTA cluster specification.

The main goal of Over The Air Upgrade cluster is to provide an interoperable mean for devices from different manufacturers to upgrade each other’s image. Additionally, the OTA Upgrade cluster defines a mechanism by which security credentials, logs and configuration file types are accessible by offering a solution that utilizes a set of optional and mandatory commands.

1.2 Scope

The document will only describe features that require implementation in order to be ZigBee OTA upgrade (cluster) certified. Other optional features including using multicast for sending upgrade messages and (upgrade) cloning will not be discussed in this document.

Currently, only Application Bootloader support is required in order to support ZigBee OTA Upgrade cluster. MAC Bootloader upgrading is not supported at the moment.

2 References

2.1 ZigBee Alliance Documents

[R1] 084912r05ZB_ZARC_Interest-OTA_Upgrades_MRD.doc

[R2] 085028r01ZB_ZARC_Interest-OTA_Upgrades_TRD.doc

[R3] 075123r02ZB_AFG-ZigBee_Cluster_Library_Specification.pdf

[R4] 075356r15ZB_AMI_PTG-AMI_Profile_Specification.pdf

[R5] 053474r18ZB_TSC-ZigBee-Specification.pdf

3 Definitions

3.1 Term Definitions

Application Bootloading

Bootloading method where the device has memory for receiving and storing a new image while it is running its current stack and application. The new image is fully received before it is applied.

OTA Upgrade Server

The ZigBee device class that sends the commands and image to a client device as part of OTA upgrade process. How the server retrieves the upgrade image is outside the scope of this document.

OTA Upgrade Client

The ZigBee device that is the target for receiving and upgrading its (running) image.

Vendor

A company or an entity that provides the software. Used in the specification when referring to ZigBee stack provider.

Manufacturer

A company or a group of companies that produce the device including both the software and the hardware. A device's manufacturer may refer to several entities; each contributing parts to make up the device.

3.2 Conformance levels

3.2.1 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.

3.2.2 may: A key word that indicates flexibility of choice with *no implied preference*.

3.2.3 shall: A key word indicating a mandatory requirement. Designers are *required* to implement all such mandatory requirements.

3.2.4 should: A key word indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase *is recommended*.

4 Acronyms and abbreviations

AES	Advanced Encryption Standard
AMI	Advanced Metering Infrastructure *or* Advanced Metering Initiative
ESP	Energy Service Portal
EUI64	Extended Universal Identifier-64
HA	Home Automation (Application Profile)
HAN	Home Area Network
IHD	In-Home Display
MRD	Market Requirements Document
NAN	Neighborhood Area Network
OTA	Over the Air
PAN	Personal Area Network
ZED	ZigBee End Device (equivalent to IEEE's RFD – Reduced Functionality Device)
ZR	ZigBee Router (equivalent to IEEE's FFD – Full Functionality Device)
SE	Smart Energy (Application Profile)
TC	Trust Center
TRD	Technical Requirements Document
ZCL	ZigBee Cluster Library
ZDO	ZigBee Device Object
ZDP	ZigBee Device Profile

5 General description

5.1 Introduction

The existing OTA upgrade methods available are platform specific, not OTA interoperable and do not provide a common framework for upgrading networks that support a mix of devices from multiple platforms and ZigBee Stack vendors.

The intent of this document is to provide an interoperable OTA upgrade of new image for devices deployed in the field. As long as the device supports the OTA Upgrade cluster and it is certified by an approved test house, its image shall be upgradeable by another device from the same or different manufacturer that also implemented and certified the OTA Upgrade cluster.

OTA Upgrade cluster will also require that in order to support OTA upgrade, the device will need to have an application bootloader installed as well as sufficient memory (external or internal) to store the newly loaded image. An application bootloader uses the running ZigBee stack and application to retrieve and store a new image. Depending upon the manufacturer, the image may consist of a bootloader image, a ZigBee stack image or only a patch to the application image. Whatever comprises the OTA upgrade image being sent to the node does not concern the ZigBee OTA cluster and it is outside the scope of this document.

To use an application bootloader, the device is required to have sufficient memory (internal or external) to store the newly downloaded OTA upgrade image. By doing so, the current running image is not overwritten until the new image has been successfully downloaded. It also allows the possibility of a node saving an image in its memory and forwarding that image to another node. Application bootloading provides flexibility of when the device decides to download new OTA upgrade image as well as when the device decides to switch to running the new image.

Since the bootloading is done at the application level, it automatically makes use of various features already offered by the ZigBee Network Layer and Application Sub Layer (APS) including the ability to bootload a device that is multiple hops away, message retries to increase reliability, and security. It also allows the network to continue to operate normally while the bootload is in progress. In addition, it supports bootloading of sleeping (RxOnWhenIdle=FALSE) devices.

The application bootload messages are built upon typical ZigBee messages, with additional ZigBee Cluster Library (ZCL) header and payload and ZigBee OTA cluster specific payload.

5.2 Cluster list

The clusters defined in this document are listed in Table 2 - Clusters specified in this document.

Table 2 - Clusters specified in this document

Cluster Name	Cluster ID	Description
OTA Upgrade	0x0019	Parameters and commands for upgrading image on devices Over The Air.

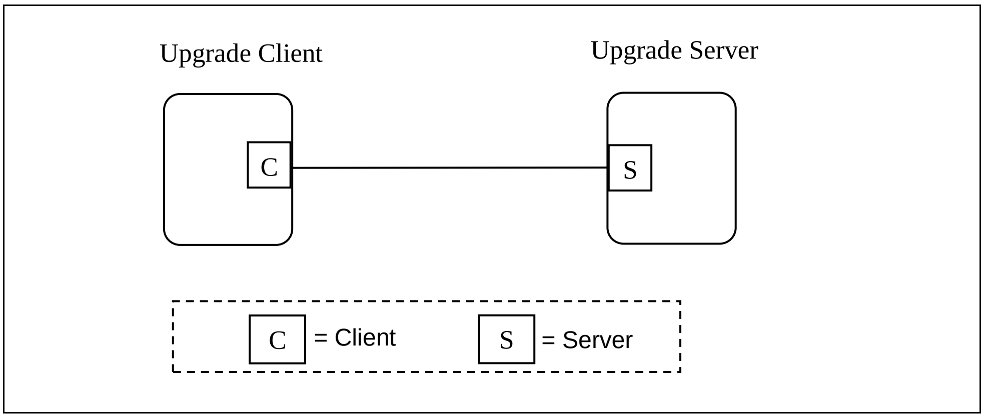

Figure 1 - Typical Usage of OTA Upgrade Cluster

Upgrade Client is a device to be upgraded with new image. Upgrade server is a device that has the new image to send to the client. This document shall specify how the client discovers the server, the over the air message format between the client and server, and the means for the server to signal the client to switch to running the new image.

It is possible that the upgrade server may have several OTA upgrade images from different manufacturers. How the upgrade server receives these OTA upgrade images and how it stores and manages them are outside the scope of this document.

In addition to the typical use case of transferring new firmware images to client devices, OTA Upgrade cluster may also be used to transfer device specific file types such as log, configuration or security credentials (needed for upgrading from SE 1.0 or SE 1.1 to SE 2.0). The cluster provides flexibility in OTA header and a set of optional commands that make transferring of such file types possible. It is up to each application profile and manufacturer to decide whether to implement and mandate such features.

6 OTA Upgrade Cluster

6.1 Overview

The cluster provides a standard way to upgrade devices in the network via OTA messages. Thus the upgrade process may be performed between two devices from different manufacturers. Devices are required to have application bootloader and additional memory space in order to successfully implement the cluster.

It shall be the responsibility of the server to notify the clients when the update images are available The client may be upgraded, downgraded or reinstalled. The upgrade server must know which client devices to upgrade and to what file version. The upgrade server may be notified of such information via the backend system. For ZR client, the server shall send a message to notify the device when updated image is available. For ZED client, the client shall query (poll) the server periodically.

The cluster is implemented in such a way that the client service works on both ZED and ZR devices. Being able to handle polling is mandatory for all devices while being able to push is optional. Hence, all devices must be able to use a ‘poll’ mechanism to send query message to the server in order to see if the server has any new file for it. The polling mechanism also puts fewer resources on the upgrade server. It is ideal to have the server maintain as little state as possible since this will scale when there are hundreds of clients in the network. The upgrade server is not required to keep track of what pieces of an image that a particular client has received; instead the client shall do that. Lastly poll makes more sense for devices that may need to perform special setup to get ready to receive an image, such as unlocking flash or allocating space for the new image.

6.2 Security

Security for the OTA Upgrade cluster encompasses these areas: image verification, image transport, and image encryption. The OTA Upgrade cluster is intended to be used with all ZigBee application profiles. Security mechanism of each application profile dictates the security level of over-the-air image upgrading. For example application profile with strict security policies (such as Smart Energy) may support image signature as well as encryption in both network and APS layers; while Home Automation application profile may only support network encryption . Each application profile must decide the list of required security policies for their use of the OTA Upgrade cluster.

6.2.1 Terminology

There are many aspects to security. These can be broken down into the following areas: confidentiality, integrity, authentication, availability, and non-repudiation. Authorization and auditing can also be considered as part of security.

Confidentiality – This is the requirement that no third party can read data that is not intended for that party. This is discussed below in Image Encryption.

Integrity – This is the property of security where the recipient can verify that data was not modified between the time it was initially distributed by the sender and when it was received by the intended recipient. Modifications to the data by third parties can be detected. This is discussed in Image Verification below.

Authentication – This is the property where the identity of the sender of data can be verified by the intended recipient. This is discussed in Image Verification below.

Availability – This refers to the property that resources are available when they are required and cannot be unfairly consumed by an attacker. The OTA Upgrade cluster does not address this; as such it will not be discussed any further.

Non-repudiation – This refers to the property where a sender/receiver cannot deny that a security exchange took place. The OTA Upgrade cluster does not address this; as such it will not be discussed any further.

6.2.2 Image Verification

It is strongly encouraged that there is a means to verify the authenticity and integrity of the bootload image. This is most often accomplished through asymmetric encryption technologies (i.e. public/private keys) where only one device is able to create a digital signature but many devices are able to verify it. Bootload images may be signed by the private key of the manufacturer with that signature appended to the image that is transported to the device. Once the complete image has been received the signature is verified using the public key of the signer.

Devices may be pre-installed with the certificate (public key) of the device that created the signature, or they may receive the certificate over-the-air. How the signer's security data is obtained is considered outside the scope of the OTA Upgrade cluster and is manufacturer specific. When signer certificates are sent over-the-air and not pre-installed, it is recommended that the transportation of the certificate be done using encryption from a trusted source to reduce the chance an attacker may inject their own signer certificate into the device.

Images with verification mechanisms built in may be transported over insecure communication mechanisms while still maintaining their authenticity and integrity. In fact it is likely that the originator of the upgrade image (the manufacturer) will not be directly connected to any ZigBee networks and therefore distribute the upgrade image across other mediums (such as the internet) before arriving on the ZigBee network. In that case it is crucial that the Image Verification be independent of the communication medium. Any attempts to tamper with the signature or the data itself must be detected and will cause the upgrade image to be rejected by the target device. An attacker that crafts its own signed image and tries to have it accepted will be rejected since that image will not be signed by the manufacturer's signing authority.

It is up to each profile to determine the minimum requirements for image verification. For Smart Energy profile there is already a minimum set of security requirements included in NEMA SG-AMI 1-2009. The OTA Upgrade cluster will communicate the methods in use for basic image verification. Individual manufacturers are free to augment this and provide their own extensions. Those extensions are outside the scope of the OTA Upgrade cluster.

Without asymmetric encryption technology, a device is limited in its ability to authenticate images. Images may be encrypted with symmetric keys such that only those devices that need to decrypt the image have access to the key. However the security of this system is dependent on the security of all devices that have access to the symmetric key.

6.2.3 Image Transport

When there is a means to verify the authenticity of the bootload images, transport of the images in a secure fashion provides little additional security to the integrity and authenticity. What secure transportation provides is a means to communicate the policies about *when* a device should perform upgrade or *what version* it should upgrade to.

A secured ZigBee network uses network security for all messages, but that does not provide point-to-point security. APS security should be used to assure that messages are sent from only the trusted source (the upgrade server). This will be utilized to provide implicit and explicit authorization by the upgrade server about when devices will *initiate* bootload events.

Distribution of upgrade image via broadcast or multicast messages is not recommended because its lack of point-to-point security. Reception of upgrade images via broadcast or multicast should not be inferred as authorization by the upgrade server to initiate the upgrade. In this case, the act of receiving the image and upgrading it should be split up into separate events. The latter communications should be done via unicast to verify that the upgrade server has authorized a device to upgrade to a previously received image. Application profiles must determine what level of authorization is required by the upgrade server.

6.2.4 Image Signature

For certain application profiles, the OTA Upgrade cluster provides mechanisms to sign the OTA file to protect the authenticity and integrity of the image. The application profiles, must determine if this is required.

6.3 OTA File Format

6.3.1 General Structure

The OTA file format is composed of a header followed by a number of sub-elements. The header describes general information about the file such as version, the manufacturer that created it, and the device it is intended for. Sub-elements in the file may contain upgrade data for the embedded device, certificates, configuration data, log messages, or other manufacturer specific pieces. Below is an example file.

Table 3 - Sample OTA File

Octets	**Variable**	**Variable**	**Variable**	**Variable**
Data	OTA Header	Upgrade Image	Signer Certificate	Signature

The OTA header will not describe details of the particular sub-elements. Each sub-element is self-describing. With exception of a few sub-elements, the interpretation of the data contained is up to the manufacturer of the device.

6.3.2 OTA Header Format

Table 4 - OTA Header Fields

Octets	Data Types	Field Names	Mandatory/Optional
4	Unsigned 32-bit integer	OTA upgrade file identifier	M
2	Unsigned 16-bit integer	OTA Header version	M
2	Unsigned 16-bit integer	OTA Header length	M
2	Unsigned 16-bit integer	OTA Header Field control	M

2	Unsigned 16-bit integer	Manufacturer code	M
2	Unsigned 16-bit integer	Image type	M
4	Unsigned 32-bit integer	File version	M
2	Unsigned 16-bit integer	ZigBee Stack version	M
32	Character string	OTA Header string	M
4	Unsigned 32-bit integer	Total Image size (including header)	M
0/1	Unsigned 8-bit integer	Security credential version	O
0/8	IEEE Address	Upgrade file destination	O
0/2	Unsigned 16-bit integer	Minimum hardware version	O
0/2	Unsigned 16-bit integer	Maximum hardware version	O

The first entry of the table above (OTA upgrade file identifier) represents the first field in the OTA header, and the last entry represents the last field. The endianness used in each data field shall be little endian in order to be compliant with general ZigBee messages.

Please refer to section 2.5.2 in [R3] for more description on data types.

6.3.2.1 OTA Upgrade File Identifier

The value is a unique 4-byte value that is included at the beginning of all ZigBee OTA upgrade image files in order to quickly identify and distinguish the file as being a ZigBee OTA cluster upgrade file, without having to examine the whole file content. This helps distinguishing the file from other file types on disk. The value is defined to be “0x0BEEF11E”.

6.3.2.2 OTA Header Version

The value enumerates the version of the header and provides compatibility information. The value is composed of a major and minor version number (one byte each). The high byte (or the most significant byte) represents the major version and the low byte (or the least significant byte) represents the minor version number. A change to the minor version means the OTA upgrade file format is still backward compatible, while a change to the major version suggests incompatibility.

The current OTA header version shall be 0x0100 with major version of “01” and minor version of “00”.

6.3.2.3 OTA Header Length

This value indicates full length of the OTA header in bytes, including the OTA upgrade file identifier, OTA header length itself to any optional fields. The value insulates existing software against new fields that may be added to the header. If new header fields added are not compatible with current running software, the implementations should process all fields they understand and then skip over any remaining bytes in the header to process the image or signing certificate. The value of the header length depends on the value of the OTA header field control, which dictates which optional OTA header fields are included.

6.3.2.4 OTA Header Field Control

The bit mask indicates whether additional information such as Image Signature or Signing Certificate are included as part of the OTA Upgrade Image.

Table 5 - OTA Header Field Control Bitmask

Bits	Name
0	Security Credential Version Present
1	Device Specific File
2	Hardware Versions Present
3 - 15	Reserved

Security credential version present bit indicates whether security credential version field is present or not in the OTA header.

Device specific file bit in the field control indicates that this particular OTA upgrade file is specific to a single device.

Hardware version present bit indicates whether minimum and maximum hardware version fields are present in the OTA header or not.

6.3.2.5 Manufacturer Code

This is the ZigBee assigned identifier for each member company. When used during the OTA upgrade process, manufacturer code value of 0xffff has a special meaning of a wild card. The value has a ‘match all’ effect. OTA server may send a command with wild card value for manufacturer code to match all client devices from all manufacturers.

6.3.2.6 Image Type

The manufacturer should assign an appropriate and unique image type value to each of its devices in order to distinguish the products. This is a manufacturer specific value. However, the OTA Upgrade cluster has reserved the last 64 values of image type value to indicate specific file types such as security credential, log, and configuration. When a client wants to request one of these specific file types, it shall use one of the reserved image type values instead of its own (manufacturer specific) value when requesting the image via Query Next Image Request command.

Table 6 - Image Type Values

File Type Values	File Type Description
0x0000 – 0xffbf	Manufacturer Specific
0xffc0	Security credential
0xffc1	Configuration
0xffc2	Log
0xffc3 – 0xfffe	Reserved (unassigned)
0xffff	Reserved: wild card

Image type value of 0xffff has a special meaning of a wild card. The value has a ‘match all’ effect. For example, the OTA server may send Image Notify command with image type value of 0xffff to indicate to a group of client devices that it has all types of images for the clients. Additionally, the OTA server may send Upgrade End Response command with image type value of 0xffff to indicate a group of clients, with disregard to their image types, to upgrade.

6.3.2.7 File Version

For firmware image, the file version represents the release and build number of the image’s application and stack. The application release and build numbers are manufacturer specific, however, each manufacturer should obtain stack release and build numbers from their stack vendor. OTA Upgrade cluster makes the recommendation below regarding how the file version should be defined, in an attempt to make it easy for humans and upgrade servers to determine which versions are newer than others. The upgrade server should use this version value to compare with the one received from the client.

The server may implement more sophisticated policies to determine whether to upgrade the client based on the file version. A higher file version number indicates a newer file.

Table 7 - Recommended File Version Definition

Application Release	Application Build	Stack Release	Stack Build
1 byte	1 byte	1 byte	1 byte
8-bit integer	8-bit integer	8-bit integer	8-bit integer

For example,

File version A: 0x10053519 represents application release 1.0 build 05 with stack release 3.5 b19.

File version B: 0x10103519 represents application release 1.0 build 10 with stack release 3.5 b19.

File version C: 0x10103701 represents application release 1.0 build 10 with stack release 3.7 b01.

File version B is newer than File version A because its application version is higher while File version C is newer than File version B because its stack version is higher.

For device specific files, the file version value may be defined differently than that for firmware image to represent version scheme of different image types. For example, version scheme for security credential data may be different than that of log or configuration file. The specific implementation is manufacturer specific.

Note that a binary-coded decimal convention (BCD) concept is used here for version number. This is to allow easy conversion to decimal digits for printing or display, and allows faster decimal calculations.

6.3.2.8 ZigBee Stack Version

This information indicates the ZigBee stack version that is used by the application. This provides the upgrade server an ability to coordinate the distribution of images to devices when the upgrades will cause a major jump that usually breaks the over-the-air compatibility, for example, from ZigBee Pro to upcoming ZigBee IP. The values below represent currently available ZigBee stack versions

Table 8 - ZigBee Stack Version Values

ZigBee Stack Version Values	Stack Name
0x0000	ZigBee 2006
0x0001	ZigBee 2007
0x0002	ZigBee Pro
0x0003	ZigBee IP
0x0004 – 0xffff	Reserved

6.3.2.9 OTA Header String

This is a manu facturer specific string that may be used to store other necessary information as seen fit by each manufacturer. The idea is to have a human readable string that can prove helpful during development cycle. The string is defined to occupy 32 bytes of space in the OTA header.

6.3.2.10 Total Image Size

The value represents the total image size in bytes. This is the total of data in bytes that shall be transferred over-the-air from the server to the client. In most cases, the total image size of an OTA upgrade image file is the sum of the OTA header and the actual file data (along with its tag) lengths. If the image is a signed image and contains a certificate of the signer, then the Total image size shall also include the signer certificate and the signature (along with their tags) in bytes.

This value is crucial in the OTA upgrade process. It allows the client to determine how many image request commands to send to the server to complete the upgrade process.

6.3.2.11 Security Credential Version

This information indicates security credential version type, such as SE1.0 or SE2.0 that the client is required to have, before it shall install the image. One use case for this is so that after the client has downloaded a new image from the server, it should check if the value of security credential version allows for running the image. If the client's existing security credential version does not match or is outdated from what specified in the OTA header, it should obtain new security credentials before upgrading to running the new image.

Table 9 - Security Credential Version

Security Credential Version Values	Security Credential Version Types
0x00	SE 1.0

0x01	SE 1.1
0x02	SE 2.0
0x03 – 0xff	Reserved

6.3.2.12 Upgrade File Destination

If Device Specific File bit is set, it indicates that this OTA file contains security credential/certificate data or other type of information that is specific to a particular device. Hence, the upgrade file destination field (in OTA header) should also be set to indicate the IEEE address of the client device that this file is meant for.

6.3.2.13 Minimum Hardware Version

The value represents the earliest hardware platform version this image should be used on. This field is defined as follows:

Table 10 - Hardware Version Format

Version	Revision
1 byte	1 byte
8-bit integer	8-bit integer

The high byte represents the version and the low byte represents the revision.

6.3.2.14 Maximum Hardware Version

The value represents the latest hardware platform this image should be used on. The field is defined the same as the Minimum Hardware Version (above).

The hardware version of the device should not be earlier than the minimum (hardware) version and should not be later than the maximum (hardware) version in order to run the OTA upgrade file.

6.3.3 Sub-element Format

Sub-elements in the file are composed of an identifier followed by a length field, followed by the data. The identifier provides for forward and backward compatibility as new sub-elements are introduced. Existing devices that do not understand newer sub-elements may ignore the data.

Table 11 - Sub-element format

Octets	**2-bytes**	**4-bytes**	**Variable**
Data	Tag ID	Length Field	Data

6.3.3.1 Tag ID

The tag identifier denotes the type and format of the data contained within the sub-element. The identifier is one of the values from Table 12 below.

6.3.3.2 Length Field

This value dictates the length of the rest of the data within the sub-element in bytes. It does not include the size of the Tag ID or the Length Fields.

6.3.3.3 Data

The length of the data in the sub-element must be equal to the value of the Length Field in bytes. The type and format of the data contained in the sub-element is specific to the Tag.

6.3.4 Tag Identifiers

Sub-elements are generally specific to the manufacturer and the implementation. However this specification has defined a number of common identifiers that may be used across multiple manufacturers.

Table 12 - Tag Identifiers

Tag Identifiers	Description
0x0000	Upgrade Image
0x0001	ECDSA Signature
0x0002	ECDSA Signing Certificate
0x0003 – 0xefff	Reserved
0xf000 – 0xffff	Manufacturer Specific Use

Manufacturers may define tag identifiers for their own use and dictate the format and behavior of devices that receive images with that data.

6.3.5 ECDSA Signature Sub-element

The ECDSA Signature sub-element contains a signature for the entire file as means of insuring that the data was not modified at any point during its transmission from the signing device.

If an image contains an ECDSA Signature Sub-element it shall be the last sub-element in the file.

Table 13 - ECDSA Signature

Octets	2-bytes	4-bytes	8-bytes	42-bytes
Data	Tag ID: 0x0001	Length Field: 0x00000032	Signer IEEE Address	Signature Data

6.3.5.1 Signer IEEE Address

This field shall contain the IEEE address of the device that created the signature, in little endian format.

6.3.5.2 Signature Data

This field shall contain the ECDSA signature data, and is generated as described in the section *ECDSA Signature Calculation*.

6.3.6 ECDSA Signing Certificate Sub-element

This sub-element is used to include information about the authority that generated the signature for the OTA file.

Table 14 - ECDSA Signing Certificate Sub-element

Octets	2-bytes	4-bytes	48-bytes
Data	Tag ID: 0x0002	Length Field: 0x00000030	ECDSA Certificate

6.3.6.1 ECDSA Certificate

This shall contain the data for the ECDSA certificate of the device. The certificate shall be formatted as described in the Smart Energy specification [R4].

6.3.7 OTA File Naming

OTA Upgrade cluster provides recommendation below regarding OTA Upgrade image file naming convention and extension. This is an effort to assist the upgrade server in sorting different image files received from different manufacturers.

The OTA Upgrade image file name should contain the following information at the beginning of the name with each field separated by a dash ("-"): manufacturer code, image type and file version. The value of each field stated should be in hexadecimal number and in capital letter. Each manufacturer may append more information to the name as seen fit to make the name more specific. The OTA Upgrade file extension should be ".zigbee".

An example of OTA Upgrade image file name and extension is "1001-00AB-10053519-upgradeMe.zigbee".

6.3.8 Signatures

It is up to the application profile to determine whether or not a signature is necessary for over the air upgrade files. If a profile has mandated the use of signatures then a device adhering to that profile shall only accept images that have a signature sub-element. If such a device receives an OTA file that does not contain a signature sub-element then the device will discard the image and proceed with any further processing required by the specific application profile. The device must verify the signature as described in the following sections prior to acting on any data inside the file.

If a profile does not require the use of signatures then devices may still choose to use images with signatures. However it is highly recommended that such a device only accept images either with signatures or without, but not accept both. A device greatly reduces its security if it will accept signed or unsigned upgrade files.

6.3.8.1 ECDSA Signature Calculation

It is expected that in most all cases the signer device is not a real ZigBee device and is not part of any ZigBee network. Therefore the signer's IEEE is not a real ZigBee device address, but the address of a virtual device that exists only to sign upgrade images for a manufacturer and or a set of products. Its address should be separate from the block of device addresses produced by a manufacturer as certified ZigBee devices.

The signature calculation shall be performed as follows:

1. A valid OTA image shall have previously been created including all the necessary header fields, tags, and their data, in the image.
2. An ECDSA signer certificate tag sub-element shall be constructed with the certificate of the signing device, and appended to the image.
3. An ECDSA signature tag sub-element shall be constructed including only the tag ID, the length of the tag (50 bytes), and the signer's IEEE address. No actual signature data shall be included yet. The tag shall be appended to the image.
4. The OTA image header shall be updated with a new total image size, including the signature certificate tag sub-element that was added, and the full size of the ECDSA signature tag sub-element (56 bytes).
5. A message digest shall be calculated over the entire image.
    a. The message digest shall be computed by using the Matyas-Meyer-Oseas cryptographic hash specified in section B.6 of [R5]. This uses the extended AES-MMO hash proposed as a change to an earlier version of [R5].
6. The CA public key of the device that issued the signer's certificate shall be obtained.
7. The signer device's public key shall be obtained by extracting it from the signer certificate.
8. The ECDSA algorithm shall be used to calculate the signature using the message digest, the CA's public key, and the signer device's public key.
9. The r and s components of the signature shall both be appended to the image. The r component shall be appended first, and then the s component.

6.3.8.2 ECDSA Signature Verification

The signature of a completely downloaded OTA file shall be verified as follows.

1. The ZigBee device shall first determine if the signer of the image is an authorized signer.
    a. It does this by extracting the signer IEEE from ECDSA signature tag sub-element.
        i. If an ECDSA signature tag sub-element is not found in the image then the image shall be discarded as invalid and no further processing shall be done.
    b. The device shall compare the extracted signer IEEE with the list of local, known, authorized signers and determines if there is a match.
    c. If no match is found then the image shall be discarded as invalid and no further processing shall be done.
2. The device shall then obtain the certificate associated with the signer IEEE.
    a. The device shall extract the signer certificate data from the ECDSA signing certificate sub-element.
        i. If there is no ECDSA signing certificate tag sub-element then it shall discard the image as invalid and no further processing shall be done.
    b. The device shall verify that the signer IEEE address within the ECDSA signature tag sub-element matches the subject field of the ECDSA signing certificate sub-element.
        i. Note: The subject field IEEE is in big-endian format and the signer IEEE is in little endian format.
    c. If the addresses do not match then the image shall be discarded as invalid and no further processing shall be done.
3. The device shall then obtain the CA public key associated with the signer.
    a. The device shall obtain the IEEE of the CA public key from the issuer field within the ECDSA certificate data of the ECDSA signing certificate sub-element.

    b. If the IEEE of the CA does not match its list of known CAs, or the public key for that CA could not be locally obtained, then the image shall be discarded as invalid and no further processing shall be done.

4. The device shall then calculate the message digest of the image.
    a. The digest shall be calculated using the Matyas-Meyer-Oseas cryptographic hash function over the entire image except for the signature data of the ECDSA signature sub-element.
        i. Note: The calculation shall include the signature tag ID of the ECDSA signature sub-element, the length field of the ECDSA signature sub-elment, and the signer IEEE field of the ECDSA signature sub-element.
5. The signer's public key shall be obtained by extracting it from the signer certificate.
6. The device shall then pass the calculated digest value, signer certificate, and CA public key to the ECDSA verification algorithm.
7. If the ECDSA algorithm returns success, then the image shall be considered valid.
8. If the ECDSA algorithm returns any other result, then the image shall be discarded as invalid and no further processing shall be done.

6.4 Discovery of the Upgrade Server

Before becoming part of the network, a device may be preprogrammed with the IEEE address of the authorized upgrade server. In this case, once the device is part of the network, it shall discover the network address of the upgrade server via ZDO network address discovery command.

If the device is not preprogrammed with the upgrade server's IEEE address, the device shall discover the upgrade server before it participates in any upgrade process. The device shall send Match Descriptor Request (ZDO command) to discover an upgrade server by specifying a single OTA cluster ID in the input Cluster attribute. If the receiving node is an upgrade server, it shall reply with Match Descriptor Response, with the (active) endpoint that the OTA cluster is implemented on, hence, identifying itself as acting as server in OTA Upgrade cluster. Since Match descriptor request may be sent as unicast or broadcast, the client may get multiple responses if there are more than one server in the network. The client shall use the first response received[1].. Each application profile should specify the frequency of OTA server discovery done by the client. After discovering the OTA server's short ID via the ZDO Match descriptor request the client shall discover the IEEE address of the upgrade server via ZDO IEEE address discovery command and store the value in UpgradeServerID attribute.

A node shall have an application link key with the Upgrade server; it shall request one prior to any OTA operations.

1. If the upgrade server is the trust center, it should use its trust center link key.
2. If the upgrade server is not the trust center, the device shall perform partner link key request, in case of SE profile or use the global link key in case of other profiles.

6.5 Server and Client

The server must be able to store one or more OTA upgrade image(s). The server may notify devices in the network when it receives new OTA upgrade image by sending an Image Notify Command. The Image Notify Command will be received reliably only on ZR devices since ZED devices may have their radio off at the time. The Image Notify Command may be sent as unicast or broadcast. If sent as broadcast, the message also has a jitter mechanism built in to avoid the server being overwhelmed by the requests from the clients. If sent as unicast, the client shall ignore the jitter value.

[1] CCB 1314.

The client device will send Query Next Image Request Command if the information in the Image Notify Command is of interest and after applying the jitter value. All devices shall send in a Query Next Image Request Command periodically regardless of whether an Image Notify was sent by the OTA server. [2]

When the device has received a response to its query indicating a new OTA upgrade image is available, the client device shall request blocks of the OTA upgrade image. The process continues until the client receives all image data. At that point, the client shall verify the integrity of the whole image received and send Upgrade End Request Command along with the upgrade status. The server shall notify the client of when to upgrade to new image in the Upgrade End Response.

It is the responsibility of the server to ensure that all clients in the network are upgraded. The server may be told which client to upgrade or it may keep a database of all clients in the network and track which client has not yet been upgraded.

6.5.1 Sleepy Devices

The upgrade server has no reliable way to immediately notify the sleepy devices of the availability of new OTA Upgrade image, hence, the devices shall query the server periodically to learn if there are new images available. The query for new upgrade image may be done as a separate event or it may be done in addition to normal scheduled communication between the device and the server. The frequency as to how often the sleepy devices query the server shall be specified by each application profile. Moreover, it is important to realize that the frequency that the sleepy device checks for new image (sending Query Next Image command) determines how often the particular node could be upgraded. This rate will also drive how fast code updates may be pushed out to each network. For the SE 1.x to SE 2.0 transition, if sleepy devices only check in once a month for the new image then it will likely to take over a month to complete the transition. If the application profile fails to set any requirement on the sleepy device checking for new images then it is unlikely that the OTA upgrade feature will work reliably for those devices.

It is a recommendation that sleepy devices shall make their best effort to poll more rapidly during the OTA Upgrade Image download process in order to ensure that the download completes in a timely manner. However, it is acknowledged that some sleepy devices may not be able to do so due to limitation on their batteries or due to other reasons such as battery-less/Green Power devices. Hence, such devices may take much longer to complete the download process.

6.6 Dependencies

Each device that wishes to implement the OTA Upgrade cluster shall have the following:

- ZigBee Device Object (ZDO) match descriptor request and response commands. The command is used to discover upgrade server.

- ZigBee Cluster Library (ZCL) global commands and basic cluster attributes.

- Application Bootloader: To actually upgrade existing image with newly installed one on the additional memory space. The implementation of the Bootloader along with its specification, for example, where it lives and its size are outside the scope of this document.

- Additional Memory Space shall be large enough to hold the whole OTA Upgrade Image: It is important to be able to store the new image until the device receives a signal from the server to switch to running the image. This is because it may be necessary for all devices in the network to switch their images at once if the new image is not OTA compatible with the old one.

[2] CCB 1315

In addition, if the client device is composed of multiple processors; each requires separate image, then the additional memory space shall be large enough to hold all the images for all the processors that make up the device. In case of server devices, its additional memory space will depend on how many images the devices are planning to hold.

The specification of the additional memory space and its connection to the processor is outside the scope of this document.

6.7 OTA Cluster Attributes

Below are attributes defined for OTA Upgrade cluster. Currently, all attributes are client side attributes (only stored on the client). There is no server side attribute at the moment. All attributes with the exception of UpgradeServerID should be initialized to their default values before being used.

Table 15 - Attributes of OTA Upgrade Cluster

Attribute Identifier	Name	Type	Range	Access	Default	Mandator / Optiona
0x0000	*UpgradeServerID*	IEEE Address	-	Read	0xffffffffffffffff	M
0x0001	*FileOffset*	Unsigned 32-bit integer	0x00000000 – 0xffffffff	Read	0xffffffff	O
0x0002	*CurrentFileVersion*	Unsigned 32-bit integer	0x00000000 – 0xffffffff	Read	0xffffffff	O
0x0003	*CurrentZigBeeStackVersion*	Unsigned 16-bit integer	0x0000 – 0xffff	Read	0xffff	O
0x0004	*DownloadedFileVersion*	Unsigned 32-bit integer	0x00000000 – 0xffffffff	Read	0xffffffff	O
0x0005	*DownloadedZigBeeStackVersion*	Unsigned 16-bit integer	0x0000 – 0xffff	Read	0xffff	O
0x0006	*ImageUpgradeStatus*	8-bit enumeration	0x00 – 0xff	Read	0x00	M
0x0007	*Manufacturer ID*	Unsigned 16-bit integer	0x0000-0xffff	Read	-	O
0x0008	*Image Type ID*	Unsigned 16-bit integer	0x0000-0xffff	Read	-	O

6.7.1 *UpgradeServerID* Attribute

The attribute is used to store the IEEE address of the upgrade server resulted from the discovery of the upgrade server’s identity. If the value is set to a non-zero value and corresponds to an IEEE address of a device that is no longer accessible, a device may choose to discover a new Upgrade Server depending on its own security policies.

The attribute is mandatory because it serves as a placeholder in a case where the client is programmed, during manufacturing time, its upgrade server ID. In addition, the attribute is used to identify the current upgrade server the client is using in a case where there are multiple upgrade servers in the network. The attribute is also helpful in a case when a client has temporarily lost connection to the network (for example, via a reset or a rejoin), it shall try to rediscover the upgrade server via network address discovery using the IEEE address stored in the attribute.

By default the value is 0xffffffffffffffff, which is an invalid IEEE address (according to [R3]). The attribute is a client-side attribute and stored on the client. Please refer to section 6.4 for description on OTA server discovery.

6.7.2 *FileOffset* Attribute

The parameter indicates the current location in the OTA upgrade image. It is essentially the (start of the) address of the image data that is being transferred from the OTA server to the client. The attribute is optional on the client and is made available in a case where the server wants to track the upgrade process of a particular client.

6.7.3 *CurrentFileVersion* Attribute

The file version of the running firmware image on the device. The information is available for the server to query via ZCL read attribute command. The attribute is optional on the client.

6.7.4 *CurrentZigBeeStackVersion* Attribute

The ZigBee stack version of the running image on the device. The information is available for the server to query via ZCL read attribute command. The attribute is optional on the client.

6.7.5 *DownloadedFileVersion* Attribute

The file version of the downloaded image on additional memory space on the device. The information is available for the server to query via ZCL read attribute command. The information is useful for the OTA upgrade management, so the server shall ensure that each client has downloaded the correct file version before initiate the upgrade. The attribute is optional on the client.

6.7.6 *DownloadedZigBeeStackVersion* Attribute

The ZigBee stack version of the downloaded image on additional memory space on the device. The information is available for the server to query via ZCL read attribute command. The information is useful for the OTA upgrade management, so the server shall ensure that each client has downloaded the correct ZigBee stack version before initiate the upgrade. The attribute is optional on the client.

6.7.7 *ImageUpgradeStatus* Attribute

The upgrade status of the client device. The status indicates where the client device is at in terms of the download and upgrade process. The status helps to indicate whether the client has completed the download process and whether it is ready to upgrade to the new image. The status may be queried by the server via ZCL read attribute command. Hence, the server may not be able to reliably query the status of ZED client since the device may have its radio off. The attribute is optional on the client.

Table 16 - Image Upgrade Status Attribute Values

Image Upgrade Status Values	Description
0x00	Normal
0x01	Download in progress
0x02	Download complete
0x03	Waiting to upgrade
0x04	Count down
0x05	Wait for more
0x06 – 0xff	Reserved

Normal status typically means the device has not participated in any download process. Additionally, the client shall set its upgrade status back to Normal if the previous upgrade process was not successful.

Download in progress status is used from when the client device receives SUCCESS status in the Query Next Image Response command from the server prior to when the device receives all the image data it needs.

Download complete status indicates the client has received all data blocks required and it has already verified the OTA Upgrade Image signature (if applied) and has already written the image onto its additional memory space. The status will be modified as soon as the client receives Upgrade End Response command from the server.

Wait to upgrade status indicates that the client is told by the server to wait until another (upgrade) command is sent from the server to indicate the client to upgrade its image.

Count down status indicates that the server has notified the client to count down to when it shall upgrade its image.

Wait for more (upgrade) image indicates that the client is still waiting to receive more OTA upgrade image files from the server. This is true for a client device that is composed of multiple processors and each processor requires different image. The client shall be in this state until it has received all necessary OTA upgrade images, then it shall transition to Download complete state.

6.7.8 Manufacturer ID

This attribute shall reflect the ZigBee assigned value for the manufacturer of the device. See also section 6.3.2.5.

6.7.9 Image Type ID

This attribute shall indicate the image type identifier of the file that the client is currently downloading, or a file that has been completely downloaded but not upgraded to yet. The value of this attribute shall be 0xFFFF when the client is not downloading a file or is not waiting to apply an upgrade.

6.8 OTA Cluster Parameters

Below are defined parameters for OTA Upgrade cluster server. These values are considered as parameters and not attributes because their values tend to change often and are not static. Moreover, some of the parameters may have multiple values on the upgrade server at one instance. For example, for DataSize parameter, the value may be different for each OTA upgrade process. These parameters are included in commands sent from server to client. The parameters cannot be read or written via ZCL global commands.

Table 17 - Parameters of OTA Upgrade Cluster

Name	Type	Range	Default	Mandatory / Optional
QueryJitter	Unsigned 8-bit integer	0x01 – 0x64	0x32	M
DataSize	Unsigned 8-bit integer	0x00 – 0xff	0xff	M
OTAImageData	Octet	Varied	all 0xff's	M
CurrentTime	Unsigned 32-bit integer	0x00000000 – 0xffffffff	0xffffffff	M
UpgradeTime or RequestTime	Unsigned 32-bit integer	0x00000000 – 0xffffffff	0xffffffff	M

6.8.1 *QueryJitter* Parameter

The parameter is part of Image Notify Command sent by the upgrade server. The parameter indicates whether the client receiving Image Notify Command should send in Query Next Image Request command or not.

The server chooses the parameter value between 1 and 100 (inclusively) and includes it in the Image Notify Command. On receipt of the command, the client will examine other information (the manufacturer code and image type) to determine if they match its own values. If they do not, it shall discard the command and no further processing shall continue. If they do match then it will determine whether or not it should query the upgrade server. It does this by randomly choosing a number between 1 and 100 and comparing it to the value of the QueryJitter parameter received. If it is less than or equal to the QueryJitter value from the server, it shall continue with the query process. If not, then it shall discard the command and no further processing shall continue.

By using the QueryJitter parameter, it prevents a single notification of a new OTA upgrade image from flooding the upgrade server with requests from clients.

6.8.2 *DataSize* Parameter

A value that indicates the length of the OTA image data included in the (Image Block Response) command payload sent from the server to client.

6.8.3 *OTAImageData* Parameter

This is a part of OTA upgrade image being sent over the air. The length of the data is dictated by the data size parameter. The server does not need to understand the meaning of the data, only the client does. The data may also be compressed or encrypted to increase efficiency or security.

The parameter is in octet data type and is used with the file offset value (defined in section 6.7.2) to indicate the location of the data and the data size value to indicate the length of the data.

6.8.4 *CurrentTime and UpgradeTime/RequestTime* Parameters

If CurrentTime and UpgradeTime are used in the command (ex. Upgrade End Response), the server uses the parameters to notify the client when to upgrade to the new image. If CurrentTime and RequestTime are used in the command (ex. Image Block Response), the server is notifying the client when to request for more upgrade data. The CurrentTime indicates the current time of the OTA server. The UpgradeTime indicates the time that the client shall upgrade to running new image. The RequestTime indicates when the client shall request for more data.

The value of the parameters and their interpretation may be different depending on whether the devices support ZCL Time cluster or not. If ZCL Time cluster is supported, the values of both parameters may indicate the UTC Time values that represent the Universal Time Coordinated (UTC) time. If the device does not support ZCL Time cluster, then it shall compute the offset time value from the difference between the two time parameters. The resulted offset time is in seconds. A device that does support the time cluster may use offset time instead of UTC Time when it sends messages that reference the time according to Table 18 - Meaning of CurrentTime and UpgradeTime Parameters.

The table below shows how to interpret the time parameter values depending on whether Time cluster is supported on the device. The intention here is to be able to support a mixed network of nodes that may not all support Time cluster.

Table 18 - Meaning of CurrentTime and UpgradeTime Parameters

***CurrentTime* Value**	***UpgradeTime or RequestTime* Value**	**Description**
0x00000000	Any	Device shall use UpgradeTime or RequestTime as an offset time from now.
0x00000001 – 0xfffffffe	Any	Server supports Time cluster; client shall use UpgradeTime/RequestTime value as UTCTime if it also supports Time cluster or it shall compute the offset time if it does not.
Any	0xffffffff	The client should wait for a (upgrade) command from the server. Note that value of 0xffffffff should not be used for RequestTime.

Using value of all 0xFF's for UpgradeTime to indicate a wait (for Upgrade End Response command from the server) on ZED client devices is not recommended since upgrade server should not be assumed to know the wake up cycle of the end device, hence, it is not guaranteed that the end device will receive the upgrade command. If the wait value (0xffffffff) is used on ZED client, the client should keep querying the server at a reasonable rate (not faster than once every 60 minutes) to see if it is time to upgrade.

Using value of all 0xFF’s for RequestTime to indicate an indefinite wait time is not recommended. If the server does not know when it will have the image data ready, it shall use a reasonable wait time and when the client resend the image request, the server shall keep telling it to wait. There is no limit to how many times the server should the client to wait for the upgrade image. Using value of 0xffffffff shall cause the client to wait indefinitely and server may not have a way to tell the client to stop waiting especially for ZED client.

6.9 OTA Upgrade Diagram

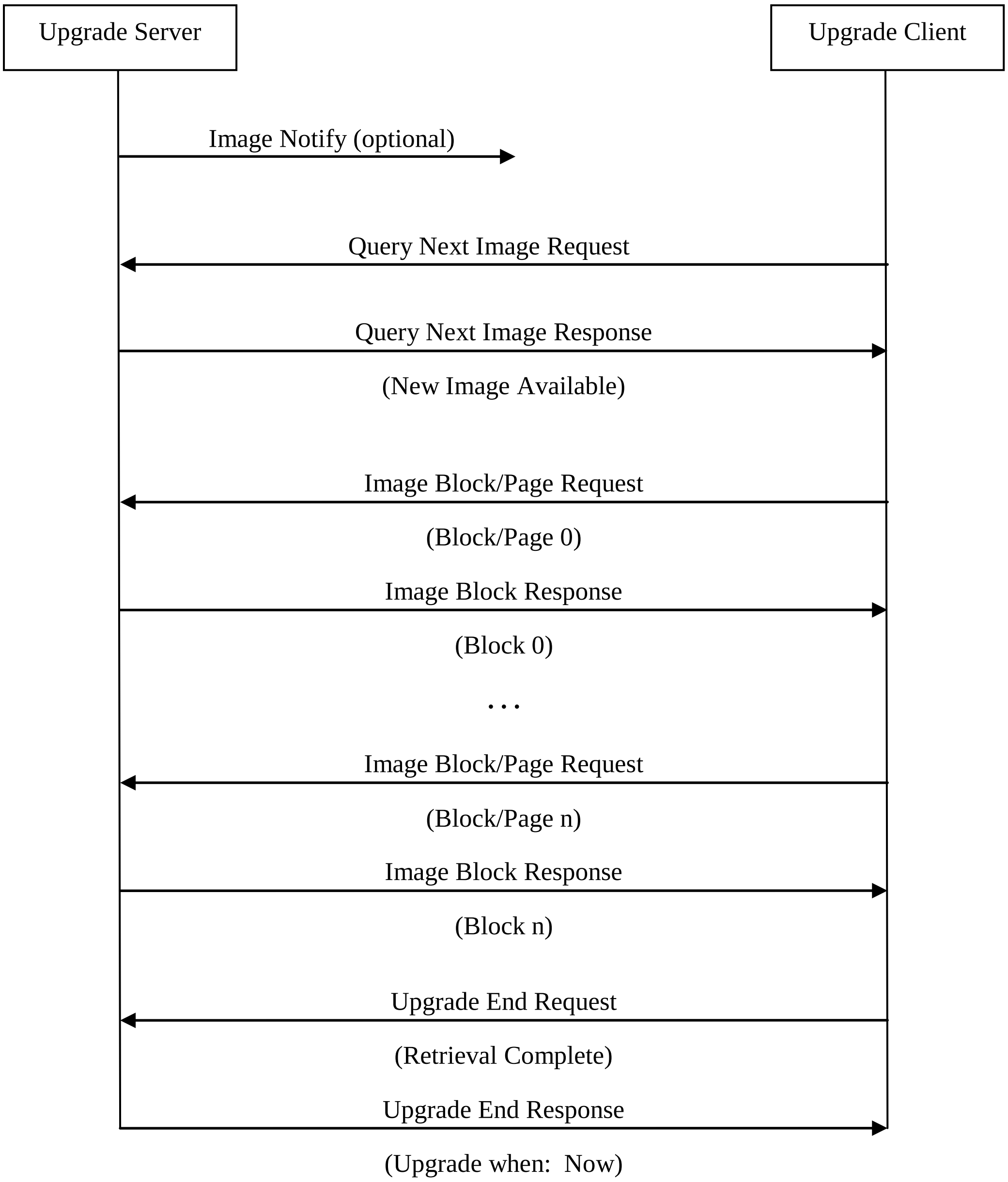

Figure 2 - OTA Upgrade Message Diagram

Please refer to section 6.10 for the command description used in the diagram above.

6.10 Command Frames

OTA upgrade messages do not differ from typical ZigBee APS messages so the upgrade process should not interrupt the general network operation. All OTA Upgrade cluster commands shall be sent with APS retry option, hence, require APS acknowledgement; unless stated otherwise.

OTA Upgrade cluster commands, the frame control value shall follow the description below:

- Frame type is 0x01: commands are cluster specific (not a global command).
- Manufacturer specific is 0x00: commands are not manufacturer specific.
- Direction: shall be either 0x00 (client->server) or 0x01 (server->client) depending on the commands.
- Disable default response is 0x00 for all OTA request commands sent from client to server: default response command shall be sent when the server receives OTA Upgrade cluster request commands that it does not support or in case an error case happens. A detailed explanation of each error case along with its recommended action is described for each OTA cluster command.
- Disable default response is 0x01 for all OTA response commands (sent from server to client) and for broadcast/multicast Image Notify command: default response command is not sent when the client receives a valid OTA Upgrade cluster response commands or when it receives broadcast or multicast Image Notify command. However, if a client receives invalid OTA Upgrade cluster response command, a default response shall be sent. A detailed explanation of each error case along with its recommended action is described for each OTA cluster command.

6.10.1 OTA Cluster Command Identifiers

Value for command identifier should be one of the values in table 8 below.

Table 19 - OTA Upgrade cluster command frames

Command Identifier Field Value	Description	Direction	Disable Default Response	Mandatory /Optional
0x00	*Image Notify*	Server -> Client(s) (0x01)	Set if sent as broadcast or multicast; Not Set if sent as unicast	O
0x01	*Query Next Image Request*	Client -> Server (0x00)	Not Set	M

Command Identifier Field Value	Description	Direction	Disable Default Response	Mandator y /Optional
0x02	*Query Next Image Response*	Server -> Client (0x01)	Set	M
0x03	*Image Block Request*	Client -> Server (0x00)	Not Set	M
0x04	*Image Page Request*	Client -> Server (0x00)	Not Set	O
0x05	*Image Block Response*	Server -> Client (0x01)	Set	M
0x06	*Upgrade End Request*	Client -> Server (0x00)	Not Set	M
0x07	*Upgrade End Response*	Server -> Client (0x01)	Set	M
0x08	*Query Specific File Request*	Client -> Server (0x00)	Not Set	O
0x09	*Query Specific File Response*	Server -> Client (0x01)	Set	O

6.10.2 OTA Cluster Status Codes

OTA Upgrade cluster uses ZCL defined status codes during the upgrade process. These status codes are included as values in status field in payload of OTA Upgrade cluster's response commands and in default response command. Some of the status codes are new and are still in the CCB process (CC-1176) in order to be included in the ZCL specification. The status codes should be included in the r04 of the ZCL specification.

Table 20 - Status Code defined and used by OTA Upgrade Cluster

ZCL Status Code	Value	Description
SUCCESS	0x00	Success Operation
ABORT	0x95	Failed case when a client or a server decides to abort the upgrade process.
NOT_AUTHORIZED	0x7E	Server is not authorized to upgrade the client
INVALID_IMAGE	0x96	Invalid OTA upgrade image (ex. failed signature validation or signer information check or

		CRC check)
WAIT_FOR_DATA	0x97	Server does not have data block available yet
NO_IMAGE_AVAILABLE	0x98	No OTA upgrade image available for a particular client
MALFORMED_COMMAND	0x80	The command received is badly formatted. It usually means the command is missing certain fields or values included in the fields are invalid ex. invalid jitter value, invalid payload type value, invalid time value, invalid data size value, invalid image type value, invalid manufacturer code value and invalid file offset value
UNSUP_CLUSTER_COMMAND	0x81	Such command is not supported on the device
REQUIRE_MORE_IMAGE	0x99	The client still requires more OTA upgrade image files in order to successfully upgrade

6.10.3 Image Notify Command

The purpose of sending Image Notify command is so the server has a way to notify client devices of when the OTA upgrade images are available for them. It eliminates the need for ZR client devices having to check with the server periodically of when the new images are available. However, all client devices still need to send in Query Next Image Request command in order to officially start the OTA upgrade process.

6.10.3.1 Payload Format

Table 21 - Format of Image Notify Command Payload

Octets	**1**	**1**	**0/2**	**0/2**	**0/4**
Data Type	8-bit Enumeration	Unsigned 8-bit	Unsigned 16-bit	Unsigned 16-bit	Unsigned 32-bit
Field Name	Payload type	Query jitter	Manufacturer code	Image type	(new) File version

6.10.3.2 Payload Field Definitions

6.10.3.2.1 Image Notify Command Payload Type

Table 22 - Image Notify Command Payload Type

Payload Type Values	Description
0x00	Query jitter
0x01	Query jitter and manufacturer code
0x02	Query jitter, manufacturer code, and image type
0x03	Query jitter, manufacturer code, image type, and new file version
0x04 – 0xff	Reserved

6.10.3.2.2 Query Jitter

See section 6.8.1 for detailed description.

6.10.3.2.3 Manufacturer Code

Manufacturer code when included in the command should contain the specific value that indicates certain manufacturer. If the server intends for the command to be applied to all manufacturers then the value should be omitted. See [R3] section 2.3.1.2 for detailed description.

6.10.3.2.4 Image Type

Image type when included in the command should contain the specific value that indicates certain file type. If the server intends for the command to be applied to all image type values then the wild card value (0xffff) should be used. See section 6.3.2.6 for detailed description.

6.10.3.2.5 (new) File Version

The value shall be the OTA upgrade file version that the server tries to upgrade client devices in the network to. If the server intends for the command to be applied to all file version values then the wild card value (0xffffffff) should be used. See section 6.3.2.7 for detailed description.

6.10.3.3 When Generated

For ZR client devices, the upgrade server may send out a unicast, broadcast, or multicast indicating it has the next upgrade image, via an Image Notify command. Since the command may not have APS security (if it is broadcast or multicast), it is considered purely informational and *not authoritative*. Even in the case of a unicast, ZR shall continue to perform the query process described in later section.

When the command is sent with payload type value of zero, it generally means the server wishes to notify all clients disregard of their manufacturers, image types or file versions. Query jitter is needed to protect the server from being flooded with clients' queries for next image.

The server may choose to send the Image Notify command to a more specific group of client devices by choosing higher payload type value. Only devices with matching information as the ones included in the Image Notify command will send back queries for next image.

However, payload type value of 0x03 has a slightly different effect. If the client device has all the information matching those included in the command including the new file version, the device shall then ignore the command. This indicates that the device has already gone through the upgrade process. This is to prevent the device from downloading the same image version multiple times. This is true if the command is sent as unicast or broadcast/multicast.

Query jitter value indicates how the server wants to space out the responses from the client; generally as a result of sending the command as broadcast or multicast. The client will only respond back if it randomly picks a value that is equal or smaller than the query jitter value. When sending Image Notify command as broadcast or multicast, the Disable Default Response bit in ZCL header must be set (to 0x01) to avoid the client from sending any default response back to the upgrade server. This agrees with section 2.4.12 in [R3].

If the command is sent as unicast, the payload type value may be zero and the Query jitter value may be the maximum value of 100 to signal the client to send in Query Next Image Request. The server may choose to use other payload type values besides zero when sending as unicast. However, since the server already knows the specific client (address) it wants to upgrade so other information is generally irrelevant.

The upgrade server may choose to send Image Notify command to avoid having ZR clients sending in Query Next Image Request to it periodically.

6.10.3.4 Effect on Receipt

On receipt of a unicast Image Notify command, the device shall always send a Query Next Image request back to the upgrade server. This provides a way for the server to force reinstallation of image on the device.

On receipt of a broadcast or multicast Image Notify command, the device shall keep examining each field included in the payload with its own value. For each field, if the value matches its own, it shall proceed to examine the next field. If values in all three fields (naming manufacturer code, image type and new file version) match its own values, then it shall discard the command. The new file version in the payload shall match either the device's current running file version or the downloaded file version (on the additional memory space).

If manufacturer code or the image type values in the payload does not match the device's own value, it shall discard the command. For payload type value of 0x01, if manufacturer code matches the device's own value, the device shall proceed. For payload type value of 0x02, if both manufacturer code and image type match the device's own values, the device shall proceed. For payload type value of 0x03, if both manufacturer code and image type match the device's own values but the new file version is not a match, the device shall proceed. In this case, the (new) file version may be lower or higher than the device's file version to indicate a downgrade or an upgrade of the firmware respectively.

To proceed, the device shall randomly choose a number between 1 and 100 and compare it to the value of the QueryJitter value in the received message. If the generated value is less than or equal to the received value for QueryJitter, it shall query the upgrade server. If not, then it shall discard the message and no further processing shall continue.

By using the QueryJitter field, a server may limit the number of devices that will query it for a new OTA upgrade image, preventing a single notification of a new software image from flooding the upgrade server with requests.

In application profiles that mandate APS encryption for OTA upgrade cluster messages, OTA messages sent as broadcast or multicast should be dropped by the receivers.

6.10.3.5 Handling Error Cases

The section describes all possible error cases that the client may detect upon reception invalid Image Notify command from the server, along with the action that shall be taken.

For invalid broadcast or multicast Image Notify command, for example, out-of-range query jitter value is used, or the reserved payload type value is used, or the command is badly formatted, the client shall ignore such command and no processing shall be done. In addition, the broadcast/multicast command shall have disable default response bit in the ZCL frame control set to 0x01.

The cases below describe how to handle invalid Image Notify command that is sent as unicast. Such command shall not have default response bit set.

6.10.3.5.1 Malformed Command

Scenarios for this error case include unicast Image Notify command with payload type of non-zero value, unicast Image Notify command with query jitter value that is not 100, broadcast Image Notify command with out of range query jitter value. In such scenario, the client should ignore the invalid message and shall send default response command with MALFORMED_COMMAND status to the server.

6.10.4 Query Next Image Request Command

6.10.4.1 Payload Format

Table 23 - Format of Query Next Image Request Command Payload

Octets	**1**	**2**	**2**	**4**	**0/2**
Data Type	Unsigned 8-bit	Unsigned 16-bit	Unsigned 16-bit	Unsigned 32-bit	Unsigned 16-bit
Field Name	Field control	Manufacturer code	Image type	(Current) File version	Hardware version

6.10.4.2 Payload Field Definitions

6.10.4.2.1 Query Next Image Request Command Field Control

The field control indicates whether additional information such as device's current running hardware version is included as part of the Query Next Image Request command.

Table 24 - Query Next Image Request Field Control Bitmask

Bits	**Name**

0	Hardware Version Present
1-7	Reserved

6.10.4.2.2 Manufacturer Code

The value shall be the device's assigned manufacturer code. Wild card value shall not be used in this case. See [R3] section 2.3.1.2 for detailed description.

6.10.4.2.3 Image Type

The value shall be between 0x0000 - 0xffbf (manufacturer specific value range). See section 6.3.2.6 for detailed description. For other image type values, Query Specific File Request command should be used.

6.10.4.2.4 (current) File Version

The file version included in the payload represents the device's current running image version. Wild card value shall not be used in this case. See section 6.3.2.7 for more detailed description.

6.10.4.2.5 (optional) Hardware Version

The hardware version if included in the payload represents the device's current running hardware version. Wild card value shall not be used in this case. See section 6.3.2.13 for hardware version format description.

6.10.4.3 When Generated

Client devices shall send a Query Next Image Request command to the server to see if there is new OTA upgrade image available. ZR devices may send the command after receiving Image Notify command. ZED device shall periodically wake up and send the command to the upgrade server. Client devices query what the *next* image is, based on their own information.

6.10.4.4 Effect on Receipt

The server takes the client's information in the command and determines whether it has a suitable image for the particular client. The decision should be based on specific policy that is specific to the upgrade server and outside the scope of this document.. However, a recommended default policy is for the server to send back a response that indicates the availability of an image that matches the manufacturer code, image type, and the highest available file version of that image on the server. However, the server may choose to upgrade, downgrade, or reinstall clients' image, as its policy dictates. If client's hardware version is included in the command, the server shall examine the value against the minimum and maximum hardware versions included in the OTA file header.

How the server retrieves and stores the clients' file is also outside the scope of this document. The server may have a backend communication to retrieve the images or it may have database software to manage file storage.

6.10.4.5 Handling Error Cases

All error cases resulting from receiving Query Next Image Request command are handled by the corresponding Query Next Image Response command with the exception of the malformed request command described below that is handled by default response command. Please refer to section 6.10.5.2 for more information regarding how the Query Next Image response command is generated.

6.10.4.5.1 Malformed Command

Upon reception a badly formatted Query Next Image Request command, for example, the command is missing one of the payload fields; the server shall send default response command with MALFORMED_COMMAND status to the client and it shall not process the command further.

6.10.5 Query Next Image Response Command

6.10.5.1 Payload Format

Table 25 - Format of Query Next Image Response Command Payload

Octets	**1**	**0/2**	**0/2**	**0/4**	**0/4**
Data Type	Unsigned 8-bit	Unsigned 16-bit	Unsigned 16-bit	Unsigned 32-bit	Unsigned 32-bit
Field Name	Status	Manufacturer code	Image type	File version	Image size

6.10.5.2 Payload Field Definitions

6.10.5.2.1 Query Next Image Response Status

Only if the status is SUCCESS that other fields are included. For other (error) status values, only status field shall be present. See section 6.10.2 for a complete list and description of OTA Cluster status codes.

6.10.5.2.2 Manufacturer Code

The value shall be the one received by the server in the Query Next Image Request command. See [R3] section 2.3.1.2 for detailed description.

6.10.5.2.3 Image Type

The value shall be the one received by the server in the Query Next Image Request command. See section 6.3.2.6 for detailed description.

6.10.5.2.4 File Version

The file version indicates the image version that the client is required to install. The version value may be lower than the current image version on the client if the server decides to perform a downgrade. The version value may be the same as the client's current version if the server decides to perform a reinstall. However, in general, the version value should be higher than the current image version on the client to indicate an upgrade. See section 6.3.2.7 for more description.

6.10.5.2.5 Image Size

The value represents the total size of the image (in bytes) including header and all sub-elements. See section 6.3.2.10 for more description.

6.10.5.3 When Generated

The upgrade server sends a Query Next Image Response with one of the following status: SUCCESS, NO_IMAGE_AVAILABLE or NOT_AUTHORIZED. When a SUCCESS status is sent, it is considered to be the explicit authorization to a device by the upgrade server that the device may upgrade to a specific software image.

A status of NO_IMAGE_AVAILABLE indicates that the server is authorized to upgrade the client but it currently does not have the (new) OTA upgrade image available for the client. In this case, for ZR client, it should wait for the next Image Notify command from the server. For ZED client, it shall continue sending Query Next Image Request to the server periodically until the image becomes available.

A status of NOT_AUTHORIZED indicates the server is not authorized to upgrade the client. In this case, the client may perform discovery again to find another upgrade server. The client may implement an intelligence to avoid querying the same unauthorized server.

6.10.5.4 Effect on Receipt

A status of SUCCESS in the Query Next Image response indicates to the client that the server has a new OTA upgrade image. The client shall begin requesting blocks of the image using the Image Block Request command. A ZED client may choose to change its wake cycle to retrieve the image more quickly.

6.10.5.5 Handling Error Cases

The Query Next Image Response command shall have the disable default response bit set. Hence, if the command is received successfully, no default response command shall be generated. However, the default response shall be generated to indicate the error cases below.

6.10.5.5.1 Malformed Command

Upon reception a badly formatted Query Next Image Response command, for example, the command is missing one of the payload field, other payload fields are included when the status field is not SUCCESS, the image type value included in the command does not match that of the device or the manufacturer code included in the command does not match that of the device; the client should ignore the message and shall send default response command with MALFORMED_COMMAND status to the server.

6.10.6 Image Block Request Command

6.10.6.1 Payload Format

Table 26 - Format of Image Block Request Command Payload

Octets	**1**	**2**	**2**	**4**	**4**	**1**	**0/8**
Data Type	Unsigned 8-bit	Unsigned 16-bit	Unsigned 16-bit	Unsigned 32-bit	Unsigned 32-bit	Unsigned 8-bit	IEEE Address
Field Name	Field control	Manufacturer code	Image type	File version	File offset	Maximum data size	Request node address

6.10.6.2 Payload Field Definitions

6.10.6.2.1 Image Block Request Command Field Control

Field control value is used to indicate additional optional fields that may be included in the payload of Image Block Request command. Currently, the device is only required to support field control value of 0x00; support for other field control value is optional.

Field control value 0x00 (bit 0 not set) indicates that the client is requesting a generic OTA upgrade file; hence, there is no need to include additional fields. The value of Image Type included in this case shall be manufacturer specific.

Field control value of 0x01 (bit 0 set) means that the client's IEEE address is included in the payload. This indicates that the client is requesting a device specific file such as security credential, log or configuration; hence, the need to include the device's IEEE address in the image request command. The value of Image type included in this case shall be one of the reserved values that are assigned to each specific file type.

Table 27 - Image Block Request Field Control Bitmask

Bits	**Name**
0	Request node's IEEE address Present
1 - 7	Reserved

6.10.6.2.2 Manufacturer Code

The value shall be that of the client device assigned to each manufacturer by ZigBee. See [R3] section 2.3.1.2 for detailed description.

6.10.6.2.3 Image Type

The value shall be between 0x0000 - 0xffbf (manufacturer specific value range). See section 6.3.2.6 for detailed description.

6.10.6.2.4 File Version

The file version included in the payload represents the OTA upgrade image file version that is being requested. See section 6.3.2.7 for more detailed description.

6.10.6.2.5 File Offset

The value indicates number of bytes of data offset from the beginning of the file. It essentially points to the location in the OTA upgrade image file that the client is requesting the data from. The value reflects the amount of (OTA upgrade image file) data (in bytes) that the client has received so far.

See section 6.7.2 for more description.

6.10.6.2.6 Maximum Data Size

The value indicates the largest possible length of data (in bytes) that the client can receive at once. The server shall respect the value and not send the data that is larger than the maximum data size. The server may send the data that is smaller than the maximum data size value, for example, to account for source routing payload overhead if the client is multiple hops away. By having the client send both file offset and maximum data size in every command, it eliminates the burden on the server for having to remember the information for each client.

6.10.6.2.7 (optional) Request Node Address

This is the IEEE address of the client device sending the Image Block Request command.

6.10.6.3 When Generated

The client device requests the image data at its leisure by sending Image Block Request command to the upgrade server. The client knows the total number of request commands it needs to send from the image size value received in Query Next Image Response command.

The client repeats Image Block Requests until it has successfully obtained all data. Manufacturer code, image type and file version are included in all further queries regarding that image. The information eliminates the need for the server to remember which OTA Upgrade Image is being used for each download process.

6.10.6.4 Effect on Receipt

The server uses the manufacturer code, image type, and file version to uniquely identify the OTA upgrade image request by the client. It uses the file offset to determine the location of the requested data within the OTA upgrade image.

6.10.6.5 Handling Error Cases

In most cases, the server sends Image Block Response command in response to the client’s Image Block Request command. However, with the exception of a few error cases described below that the server shall send default response command as a response.

6.10.6.5.1 Malformed Command

Upon reception a badly formatted Image Block Request command, for example, the command is missing one of the payload field or the file offset value requested by the client is invalid, for example, the value is larger than the total image size; the server should ignore the message and it shall send default response command with MALFORMED_COMMAND status to the client.

6.10.6.5.2 No Image Available

If either manufacturer code or image type or file version information in the request command is invalid or the OTA upgrade file for the client for some reason has disappeared which result in the server no longer able to retrieve the file, it shall send default response command with NO_IMAGE_AVAILABLE status to the client. After three attempts, if the client keeps getting the default response with the same status, it should go back to sending Query Next Image Request periodically or waiting for next Image Notify command.

6.10.6.5.3 Command Not Supported

If the client sends image request command with field control value of 0x01 that indicates device specific file request and if the server does not support such request, it shall send default response with UNSUP_CLUSTER_COMMAND status. Upon reception of such response, the client should terminate the attempt to request the device specific file and it may try to query different server.

6.10.7 Image Page Request Command

6.10.7.1 Payload Format

Table 28 - Image Page Request Command Payload

Octets	**1**	**2**	**2**	**4**	**4**	**1**	**2**	**2**	**0/8**
Data Type	Unsigned 8-bit	Unsigned 16-bit	Unsigned 16-bit	Unsigned 32-bit	Unsigned 32-bit	Unsigned 8-bit	Unsigned 16-bit	Unsigned 16-bit	IEEE Address
Field Name	Field control	Manufacturer code	Image type	File version	File offset	Maximum data size	Page size	Response Spacing	Request node address

6.10.7.2 Payload Field Definitions

6.10.7.2.1 Image Page Request Command Field Control

Field control value is used to indicate additional optional fields that may be included in the payload of Image Page Request command. Currently, the device is only required to support field control value of 0x00; support for other field control value is optional.

Field control value 0x00 indicates that the client is requesting a generic OTA upgrade file; hence, there is no need to include additional fields. The value of Image Type included in this case shall be manufacturer specific.

Field control value of 0x01 means that the client's IEEE address is included in the payload. This indicates that the client is requesting a device specific file such as security credential, log or configuration; hence, the need to include the device's IEEE address in the image request command. The value of Image type included in this case shall be one of the reserved values that are assigned to each specific file type.

Table 29 - – Image Page Request Field Control Bitmask

Bits	Name
0	Request node's IEEE address Present
1 - 7	Reserved

6.10.7.2.2 Manufacturer Code

The value shall be that of the client device assigned to each manufacturer by ZigBee. See [R3] section 2.3.1.2 for detailed description.

6.10.7.2.3 Image Type

The value shall be between 0x0000 - 0xffbf (manufacturer specific value range). See section 6.3.2.6 for detailed description.

6.10.7.2.4 File Version

The file version included in the payload represents the OTA upgrade image file version that is being requested. See section 6.3.2.7 for more detailed description.

6.10.7.2.5 File Offset

The value indicates number of bytes of data offset from the beginning of the file. It essentially points to the location in the OTA upgrade image file that the client is requesting the data from. The value reflects the amount of (OTA upgrade image file) data (in bytes) that the client has received so far.

See section 6.7.2 for more description.

6.10.7.2.6 Maximum Data Size

The value indicates the largest possible length of data (in bytes) that the client can receive at once. The server shall respect the value and not send the data that is larger than the maximum data size. The server may send the data that is smaller than the maximum data size value, for example, to account for source routing payload overhead if the client is multiple hops away. By having the client send both file offset and maximum data size in every command, it eliminates the burden on the server for having to remember the information for each client.

6.10.7.2.7 Page Size

The value indicates the number of bytes to be sent by the server before the client sends another Image Page Request command. In general, page size value shall be larger than the maximum data size value.

6.10.7.2.8 Response Spacing

The value indicates how fast the server shall send the data (via Image Block Response command) to the client. The value is determined by the client. The server shall wait at the minimum the (response) spacing value before sending more data to the client. The value is in milliseconds

6.10.7.2.9 (optional) Request Node Address

This is the IEEE address of the client device sending the Image Block Request command.

6.10.7.3 When Generated

The support for the command is optional. The client device may choose to request OTA upgrade data in one page size at a time from upgrade server. Using Image Page Request reduces the numbers of requests sent from the client to the upgrade server, compared to using Image Block Request command. In order to conserve battery life a device may use the Image Page Request command. Using the Image Page Request command eliminates the need for the client device to send Image Block Request command for every data block it needs; possibly saving the transmission of hundreds or thousands of messages depending on the image size.

The client keeps track of how much data it has received by keeping a cumulative count of each data size it has received in each Image Block Response. Once the count has reach the value of the page size requested, it shall repeat Image Page Requests until it has successfully obtained all pages. Note that the client may choose to switch between using Image Block Request and Image Page Request during the upgrade process. For example, if the client does not receive all data requested in one Image Page Request, the client may choose to request the missing block of data using Image Block Request command, instead of requesting the whole page again.

Since a single Image Page Request may result in multiple Image Block Response commands sent from the server, the client, especially ZED client, should make its best effort to ensure that all responses are received. A ZED client may select a small value for the response spacing and stay awake to receive all data blocks. Or it may choose a larger value and sleeps between receiving each data block.

Manufacturer code, image type and file version are included in all further queries regarding that image. The information eliminates the need for the server to remember which OTA Upgrade Image is being used for each download process.

6.10.7.4 Effect on Receipt

The server uses the file offset value to determine the location of the requested data within the OTA upgrade image. The server may respond to a single Image Page Request command with possibly multiple Image Block Response commands; depending on the value of page size. Each Image Block Response command sent as a result of Image Page Request command shall have increasing ZCL sequence number. Note that the sequence number may not be sequential (for example, if the server is also upgrading another client simultaneously); additionally ZCL sequence numbers are only 8-bit and may wrap.

In response to the Image Page Request, the server shall send Image Block Response commands with no APS retry to disable APS acknowledgement. The intention is to minimize the number of packets sent by the client in order to optimize the energy saving. APS acknowledgement is still used for Image Block Response sent in response to Image Block Request command.

Image Block Response message (in response to Image Page Request) only relies on network level retry. This may not be as reliable over multiple hops communication, however, the benefit of using Image Page Request is to save energy on the ZED client and using APS ack with the packet undermines that effort. ZED client needs to make the decision which request it uses. Image Page Request may speed up the upgrade process; the client transmits fewer packets, hence, less energy use but it may be less reliable. On the other hand, Image block request may slow down the upgrade process; the client is required to transmit more packets but it is also more predictable and reliable; it also allows the upgrade process to proceed at the client's pace.

6.10.7.5 Handling Error Cases

In most cases, the server sends Image Block Response command in response to the client’s Image Page Request command. However, with the exception of a few error cases described below that the server shall send default response command as a response.

6.10.7.5.1 Malformed Command

Upon reception a badly formatted Image Page Request command, for example, the command is missing one of the payload fields or the file offset value requested by the client is invalid. The server should ignore the message and it shall send default response command with MALFORMED_COMMAND status to the client.

6.10.7.5.2 No Image Available

If either manufacturer code or image type or file version information in the request command is invalid or the OTA upgrade file for the client for some reason has disappeared which result in the server no longer able to retrieve the file, it shall send default response command with NO_IMAGE_AVAILABLE status to the client. After three attempts, if the client keeps getting the default response with the same status, it should go back to sending Query Next Image Request periodically or waiting for next Image Notify command.

6.10.7.5.3 Command Not Supported

If the client sends Image Page Request command with field control value of 0x00 to request OTA upgrade image and the server does not support Image Page Request command, it shall send default response with UNSUP_CLUSTER_COMMAND status. Upon reception of such response, the client shall switch to using Image Block Request command instead to request OTA image data.

If the client sends image request command with field control value of 0x01 that indicates device specific file request and if the server does not support such request, it shall send default response with UNSUP_CLUSTER_COMMAND status. Upon reception of such response, the client should terminate the attempt to request the device specific file and it may try to query different server.

6.10.8 Image Block Response Command

6.10.8.1 Payload Format

Table 30 - Image Block Response Command Payload with SUCCESS status

Octets	**1**	**2**	**2**	**4**	**4**	**1**	**Variable**
Data Type	Unsigned 8-bit	Unsigned 16-bit	Unsigned 16-bit	Unsigned 32-bit	Unsigned 32-bit	Unsigned 8-bit	Octet
Field Name	Success status	Manufacturer code	Image type	File version	File offset	Data size	Image data

Table 31 - Image Block Response Command Payload with WAIT_FOR_DATA status

Octets	**1**	**4**	**4**
Data Type	Unsigned 8-bit	Unsigned 32-bit	Unsigned 32-bit
Field Name	Wait for data Status	Current time	Request time

Table 32 - Image Block Response Command Payload with ABORT status

Octets	**1**
Data Type	Unsigned 8-bit
Field Name	Abort Status

6.10.8.2 Payload Field Definitions

6.10.8.2.1 Image Block Response Status

The status in the Image Block Response command may be SUCCESS, ABORT or WAIT_FOR_DATA. If the status is ABORT then only the status field shall be included in the message, all other fields shall be omitted.

See section 6.10.2 for a complete list and description of OTA Cluster status codes.

6.10.8.2.2 Manufacturer Code

The value shall be the same as the one included in Image Block/Page Request command. See [R3] section 2.3.1.2 for detailed description.

6.10.8.2.3 Image Type

The value shall be the same as the one included in Image Block/Page Request command. See section 6.3.2.6 for detailed description.

6.10.8.2.4 File Version

The file version indicates the image version that the client is required to install. The version value may be lower than the current image version on the client if the server decides to perform a downgrade. The version value may be the same as the client's current version if the server decides to perform a reinstall. However, in general, the version value should be higher than the current image version on the client to indicate an upgrade. See section 6.3.2.7 for more description.

6.10.8.2.5 File Offset

The value represents the location of the data requested by the client. For most cases, the file offset value included in the (Image Block) response should be the same as the value requested by the client. For (unsolicited) Image Block responses generated as a result of Image Page Request, the file offset value shall be incremented to indicate the next data location.

6.10.8.2.6 Data Size

The value indicates the length of the image data (in bytes) that is being included in the command. The value may be equal or smaller than the maximum data size value requested by the client. See section 6.8.2 for more description.

6.10.8.2.7 Image Data

The actual OTA upgrade image data with the length equals to data size value. See section 6.8.3 for more description.

6.10.8.2.8 Current Time and Request Time

If status is WAIT_FOR_DATA, the payload then includes the server's current time and the request time that the client shall retry the request command. The client shall wait at least the request time value before trying again. In case of sleepy device, it may choose to wait longer than the specified time in order to not disrupt its sleeping cycle. If the current time value is zero that means the server does not support UTC time and the client shall treat the request time value as offset time. If neither time value is zero, and the client supports UTC time, it shall treat the request time value as UTC time. If the client does not support UTC time, it shall calculate the offset time from the difference between the two time values. The offset indicates the minimum amount of time to wait in seconds. The UTC time indicates the actual time moment that needs to pass before the client should try again.

See section 6.8.4 for more description.

6.10.8.3 When Generated

Upon receipt of an Image Block Request command the server shall generate an Image Block Response. If the server is able to retrieve the data for the client, it will respond with a status of SUCCESS and it will include all the fields in the payload. The use of file offset allows the server to send packets with variable data size during the upgrade process. This allows the server to support a case when the network topology of a client may change during the upgrade process, for example, mobile client may move around during the upgrade process. If the client has moved a few hops away, the data size shall be smaller. Moreover, using file offset eliminates the need for data padding since each Image Block Response command may contain different data size. A simple server implementation may choose to only support largest possible data size for the worst-case scenario in order to avoid supporting sending packets with variable data size.

The server shall respect the maximum data size value requested by the client and shall not send the data with length greater than that value. The server may send the data with length smaller than the value depending on the network topology of the client. For example, the client may be able to receive 100 bytes of data at once so it sends the request with 100 as maximum data size. But after considering all the security headers (perhaps from both APS and network levels) and source routing overhead (for example, the client is five hops away), the largest possible data size that the server can send to the client shall be smaller than 100 bytes.

If the server simply wants to cancel the download process, it shall respond with ABORT status. An example is while upgrading the client the server may receive newer image for that client. It may then choose to abort the current process so that the client may reinitiate a new upgrade process for the newer image.

If the server does not have the image block available for the client yet or it wants to slow down (pause) the download process, it shall send the response back with status WAIT_FOR_DATA and with RequestTime value that the client shall wait before resending the request.

6.10.8.4 Effect on Receipt

When the client receives the Image Block Response it shall examine the status field. If the value is SUCCESS it shall write the image data to its additional memory space. The client then shall continue to send Image Block Request commands with incrementing block numbers to request the remaining blocks of the OTA upgrade image. If the client has received the final block of the image, it shall generate an Upgrade End request command. In case of the client using Image Page Request, after receiving an Image Block Response, the client shall wait for response spacing time before expecting another Image Block Response from the server. A ZED client may go to sleep in between receiving Image Block Responses in order to save the energy.

If the client receives a response with ABORT status, it shall abort the upgrade process. It may retry the entire upgrade operation at a later point in time.

Upon receipt of WAIT_FOR_DATA status, the client shall wait at a minimum for the specified RequestTime and try to retrieve the image data again by resending Image Block Request or Image Page Request command with the same file offset value.

6.10.8.5 Handling Error Cases

If Image Block Response command is received successfully by the client, no default response will be generated if the disable default response bit is set in the ZCL header. However, a few error cases described below may cause the client to send default response to the server with an error code.

6.10.8.5.1 Malformed Command

Upon reception a badly formatted Image Block Response command, for example, the command is missing one of the payload field, the payload fields do not correspond to the status field, the request time value returned by the server is invalid, for example, the value is less than the client’s current time or the value is less than the server’s own current time, the data size value returned by the server is invalid, for example, the value is greater than the maximum data size specified by the client, or the value does not match the number of bytes of data actually included in the payload, or the value, when combined with file offset, is greater than the total image size or the file offset value returned by the server is invalid. The client should ignore the command and shall send default response command with MALFORMED_COMMAND status to the server.

6.10.9 Upgrade End Request Command

6.10.9.1 Payload Format

Table 33 - Format of Upgrade End Request Command Payload

Octets	**1**	**2**	**2**	**4**
Data Type	Unsigned 8-bit	Unsigned 16-bit	Unsigned 16-bit	Unsigned 32-bit
Field Name	Status	Manufacturer code	Image type	File version

6.10.9.2 Payload Field Definitions

6.10.9.2.1 Upgrade End Request Command Status

The status value of the Upgrade End Request command shall be SUCCESS, INVALID_IMAGE, REQUIRE_MORE_IMAGE, or ABORT. See section 6.10.2 for more description.

6.10.9.2.2 Manufacturer Code

The value shall be that of the client device assigned to each manufacturer by ZigBee. See [R3] section 2.3.1.2 for detailed description.

6.10.9.2.3 Image Type

The value shall be between 0x0000 - 0xffbf (manufacturer specific value range). See section 6.3.2.6 for detailed description.

6.10.9.2.4 File Version

The file version included in the payload represents the newly downloaded OTA upgrade image file version. See section 6.3.2.7 for more detailed description.

6.10.9.3 When Generated

Upon reception all the image data, the client should verify the image to ensure its integrity and validity. If the device requires signed images it shall examine the image and verify the signature as described in section 6.3.8.2. Clients may perform additional manufacturer specific integrity checks to validate the image, for example, CRC check on the actual file data.

If the image fails any integrity checks, the client shall send an Upgrade End Request command to the upgrade server with a status of INVALID_IMAGE. In this case, the client may reinitiate the upgrade process in order to obtain a valid OTA upgrade image. The client shall not upgrade to the bad image and shall discard the downloaded image data.

If the image passes all integrity checks and the client does not require additional OTA upgrade image file, it shall send back an Upgrade End Request with a status of SUCCESS. However, if the client requires multiple OTA upgrade image files before performing an upgrade, it shall send an Upgrade End

Request command with status REQUIRE_MORE_IMAGE. This shall indicate to the server that it cannot yet upgrade the image it received.

If the client decides to cancel the download process for any other reasons, it has the option of sending Upgrade End Request with status of ABORT at anytime during the download process. The client shall then try to reinitiate the download process again at a later time.

When a client finishes downloading a device specific file, it shall send Upgrade End Request command with status of SUCCESS to the server to indicate the end of the upgrade process.

6.10.9.4 Effect on Receipt

For manufacturer specific image type file download, upon receipt of a SUCCESS Upgrade End Request command the upgrade server shall reply with the Upgrade End Response indicating when the client shall upgrade to the newly retrieved image. For other status value received such as INVALID_IMAGE, REQUIRE_MORE_IMAGE, or ABORT, the upgrade server shall not send Upgrade End Response command but it shall send default response command with status of success and it shall wait for the client to reinitiate the upgrade process.

The server may utilize the Upgrade End Request command as a means to know when devices are done downloading a particular image. This helps the server manage the images and remove those that are no longer needed. However, the upgrade server should not rely on receiving the command and may impose upper limits on how long it will store a particular OTA upgrade image. The specific implementation of this is outside the scope of this document.

6.10.9.5 Handling Error Cases

Upgrade End Request command does not have disable default response bit set. Hence, in a case where the Upgrade End Request command has been received and the server does not send Upgrade End Response command in response, a default response command shall be sent with SUCCESS status. If the Upgrade End Request command has not been received, default response command with error status shall be sent as described below.

6.10.9.5.1 Malformed Command

Upon reception a badly formatted Upgrade End Request command, for example, the command is missing one of the payload fields. The server shall send default response command with MALFORMED_COMMAND status to the client.

6.10.10 Upgrade End Response Command

6.10.10.1 Payload Format

Table 34 - Format of Upgrade End Response Command Payload

Octets	**2**	**2**	**4**	**4**	**4**
Data Type	Unsigned 16-bit	Unsigned 16-bit	Unsigned 32-bit	Unsigned 32-bit	Unsigned 32-bit
Field Name	Manufacturer code	Image type	File version	Current time	Upgrade time

6.10.10.2 Payload Field Definitions

The ability to send the command with wild card values for manufacturer code, image type and file version is useful in this case because it eliminates the need for the server having to send the command multiple times for each manufacturer as well as having to keep track of all devices' manufacturers in the network.

6.10.10.2.1 Manufacturer Code

Manufacturer code may be sent using wildcard value of 0xffff in order to apply the command to all devices disregard of their manufacturers. See [R3] section 2.3.1.2 for detailed description.

6.10.10.2.2 Image Type

Image type may be sent using wildcard value of 0xffff in order to apply the command to all devices disregard of their manufacturers. See section 6.3.2.6 for detailed description.

6.10.10.2.3 File Version

The file version included in the payload represents the newly downloaded OTA upgrade image file version. The value shall match that included in the request. Alternatively, file version may be sent using wildcard value of 0xffffffff in order to apply the command to all devices disregard of their manufacturers. See section 6.3.2.7 for more detailed description.

6.10.10.2.4 Current Time and Upgrade Time

Current time and Upgrade time values are used by the client device to determine when to upgrade its running firmware image(s) with the newly downloaded one(s). See section 6.8.4 for more description.

6.10.10.3 When Generated

When an upgrade server receives an Upgrade End Request command with a status of INVALID_IMAGE, REQUIRE_MORE_IMAGE, or ABORT, no additional processing shall be done in its part. If the upgrade server receives an Upgrade End Request command with a status of SUCCESS, it shall generate an Upgrade End Response with the manufacturer code and image type received in the Upgrade End Request along with the times indicating when the device should upgrade to the new image.

The server may send an unsolicited Upgrade End Response command to the client. This may be used for example if the server wants to synchronize the upgrade on multiple clients simultaneously. For client devices, the upgrade server may unicast or broadcast Upgrade End Response command indicating a single client device or multiple client devices shall switch to using their new images. The command may not be reliably received by sleepy devices if it is sent unsolicited.

For device specific file download, the client should not always expect the server to respond back with Upgrade End Response command. For example, in a case of a client has just finished retrieving a log file from the server, the server may not need to send Upgrade End Response command. However, if the client has just retrieved a security credential or a configuration file, the server may send Upgrade End Response command to notify the client of when to apply the file. The decision of whether Upgrade End Response command should be sent for device specific file download is manufacturer specific.

6.10.10.4 Effect on Receipt.

The client shall examine the manufacturer code, image type and file version to verify that they match its own. If the received values do not match its own values or they are not wild card values, then it shall discard the command and no further processing shall continue. If all values match, the client shall examine the time values to determine the upgrade time. For more information on determining the time, please refer to section 6.8.4.

6.10.10.5 Handling Error Cases

If Upgrade End Response command is received successfully by the client or if it is sent as broadcast or multicast, no default response will be generated. However, a few error cases described below may cause the client to send default response to the server.

6.10.10.5.1 Malformed Command

Upon reception a badly formatted Upgrade End Response command, for example, the command is missing one of the payload field or the request time value returned by the server is invalid, for example, the value is less than the client's current time or the value is less than the server's own current time. The client should ignore the command and shall send default response command with MALFORMED_COMMAND status to the server.

6.10.11 Query Specific File Request Command

6.10.11.1 Payload Format

Table 35 - Format of Query Specific File Request Command Payload

Octets	**8**	**2**	**2**	**4**	**2**
Data Type	IEEE Address	Unsigned 16-bit	Unsigned 16-bit	Unsigned 32-bit	Unsigned 16-bit
Field Name	Request node address	Manufacturer code	Image type	File version	(Current) ZigBee stack version

6.10.11.2 Payload Field Definitions

6.10.11.2.1 Request Node Address

This is the IEEE address of the client device sending the request command. This indicates that the client is requesting a device specific file such as security credential, log or configuration; hence, the need to include the device's IEEE address in the image request command.

6.10.11.2.2 Manufacturer Code

The value shall be that of the client device assigned to each manufacturer by ZigBee. See [R3] section 2.3.1.2 for detailed description.

6.10.11.2.3 Image Type

The value of image type included in this case shall be one of the reserved values that are assigned to each specific file type. The value should be between 0xffc0 – 0xfffe, however, only 0xffc0 – 0xffc2 is being used currently. See section 6.3.2.6 for detailed description.

6.10.11.2.4 File Version

The value indicates the version of the device specific file being requested. See section 6.3.2.7 for more detailed description.

6.10.11.2.5 (current) ZigBee Stack Version

The value may represent the current running ZigBee stack version on the device or the ZigBee stack version of the OTA upgrade image being stored in additional memory space. The decision of which value to include depends on which device specific file being requested. For example, if the client is requesting a new security credential file in order to be able to run the newly downloaded image (ex. SE 2.0), then it should include the ZigBee stack version value of the new image.

6.10.11.3 When Generated

Client devices shall send a Query Specific File Request command to the server to request for a file that is specific and unique to it. Such file could contain non-firmware data such as security credential (needed for upgrading from Smart Energy 1.1 to Smart Energy 2.0), configuration or log. When the device decides to send the Query Specific File Request command is manufacturer specific. However, one example is during upgrading from SE 1.1 to 2.0 where the client may have already obtained new SE 2.0 image and now needs new SE 2.0 security credential data.

The fields included in the payload helps the upgrade server in obtaining or creating the right file for the client.

6.10.11.4 Effect on Receipt

The server takes the client's information in the command and either obtain the file via the backend system or create the file itself. Details of how the file is being obtained or created is manufacturer specific and outside the scope of this document. The device specific file shall follow OTA upgrade file format (section 6.3) and shall have Device Specific File bit set in OTA header field control. Moreover, the value of the Upgrade File Destination field in the OTA header shall match the Request node address value in the command's field.

6.10.11.5 Handling Error Cases

In most cases all error cases resulted from receiving Query Specific File Request command are handled by the corresponding Query Specific File Response command with the exception of a few error cases described below that are handled by default response command.

6.10.11.5.1 Malformed Command

Upon reception a badly formatted Query Specific File Request command, for example, the command is missing one of the payload fields; the server shall send default response command with MALFORMED_COMMAND status to the client and it shall not process the command further.

6.10.11.5.2 Command Not Supported

Certain server may not support transferring of device specific file and the implement of Query Specific File Request command; in this case the server shall send default response with UNSUP_CLUSTER_COMMAND status.

6.10.12 Query Specific File Response Command

6.10.12.1 Payload Format

Table 36 - Format of Query Specific File Response Command Payload

Octets	**1**	**0/2**	**0/2**	**0/4**	**0/4**
Data Type	Unsigned 8-bit	Unsigned 16-bit	Unsigned 16-bit	Unsigned 32-bit	Unsigned 32-bit
Field Name	Status	Manufacturer code	Image type	File version	Image size

6.10.12.2 Payload Field Definitions

6.10.12.2.1 Query Specific File Response Status

Only if the status is SUCCESS that other fields are included. For other (error) status values, only status field shall be present.

6.10.12.2.2 Manufacturer Code

The value shall be the one received by the server in the Query Specific File Request command. See [R3] section 2.3.1.2 for detailed description.

6.10.12.2.3 Image Type

The value shall be the one received by the server in the Query Specific File Request command. See section 6.3.2.6 for detailed description.

6.10.12.2.4 File Version

The file version indicates the image version that the client is required to download. The value shall be the same as the one included in the request. See section 6.3.2.7 for more description.

6.10.12.2.5 Image Size

The value represents the total size of the image (in bytes) including all sub-elements. See section 6.3.2.10 for more description.

6.10.12.3 When Generated

The server sends Query Specific File Response after receiving Query Specific File Request from a client. The server shall determine whether it first supports the Query Specific File Request command. Then it shall determine whether it has the specific file being requested by the client using all the information included in the request. The upgrade server sends a Query Specific File Response with one of the following status: SUCCESS, NO_IMAGE_AVAILABLE or NOT_AUTHORIZED.

A status of NO_IMAGE_AVAILABLE indicates that the server currently does not have the device specific file available for the client. A status of NOT_AUTHORIZED indicates the server is not authorized to send the file to the client.

6.10.12.4 Effect on Receipt

A status of SUCCESS in the Query Specific File response indicates to the client that the server has a specific file for it. The client shall begin requesting file data using the Image Block Request or Image Page Request command with a field control value set to 0x01 and include its IEEE address. A ZED client may choose to change its wake cycle to retrieve the file more quickly.

If the client receives the response with status of NOT_AUTHORIZED, it may perform discovery again to find another upgrade server. The client may implement an intelligence to avoid querying the same unauthorized server.

6.10.12.5 Handling Error Cases

Query Specific File Response command shall have disable default response bit set. Hence, if the command is received successfully, no default response command shall be generated. However, the default response shall be generated to indicate the error cases below.

6.10.12.5.1 Malformed Command

Upon reception a badly formatted Query Specific File Response command, for example, the command is missing one of the payload field, other payload fields are included when the status field is not SUCCESS, the manufacturer code included in the command does not match that of the device or the image type value included in the command does not match that of the device; the client should ignore the message and shall send default response command with MALFORMED_COMMAND status to the server.

6.11 OTA Upgrade Cluster Management

This section provides ways for the upgrade server to monitor and manage the network-wide OTA upgrade process. It is important to realize that the server cannot reliably query the upgrade status of the sleepy devices.

6.11.1 Query Upgrade Status

Server may send ZCL read attribute command for Image Upgrade Status attribute on the client devices. The attribute indicates the progress of the client's file download as well as its upgrade progress. The server may want to make sure that all clients have completely downloaded their new images prior to issuing the Upgrade End Response command.

A client shall only download a single file at a time. It shall not download a second file while the first file download is incomplete. This insures that the values in the client's attributes can be correlated to a single download instance.

6.11.2 Query Downloaded ZigBee Stack and File versions

The server may send ZCL read attribute command to a client to determine its downloaded ZigBee stack version and file version. The server should make sure that the client has downloaded the correct image prior to issuing the Upgrade End Response command.

6.12 OTA Upgrade Process

Once a device has completely downloaded the image and returned a status of SUCCESS in the Upgrade End Request, it shall obey the server's directive based on when it should upgrade. However there are many failure scenarios where this may not be possible. In such failure case, the device should attempt to contact the server and determine what should be done, but if that has failed as well, then it may apply its update without an explicit command by the server.

After receiving an Upgrade End Response from the server the client will apply the upgrade according to time values specified in the message. If the response directs the device to wait forever, it shall periodically query the server about when it should apply the new upgrade. This shall happen at a period no more often than once every 60 minutes. If the server is unreachable after 3 retries, the device may apply the upgrade.

The client does not need to persistently store the time indicating when to apply the upgrade. If the client feels that it has lost connection to the upgrade server, it shall first try to rediscover the upgrade server perhaps by rejoining to the network and performing network address discovery using the stored UpgradeServerID attribute. Once the server is found, the client shall resend an Upgrade End Request command with a status of SUCCESS to the server, including the relevant upgrade file information. The server shall send it a response again indicating when it should upgrade. If the device is unable to communicate to the upgrade server or it cannot synchronize the time, it may apply the upgrade anyway.

When the time comes for the client to upgrade, the device should begin the manufacturer specific method to upgrade its image. The upgrade may involve one or more hardware resets. Once the device has completed the upgrade it should be able to reinitialize itself and start communicating on the network again. Previous network information such as channel, power, short pan id, extended pan id should be preserved across the upgrade.

6.13 Application Profile Specific Decisions

Below are the decisions that each application profile needs to make in order to ensure successful OTA upgrade of devices in the network.

- The following are security considerations that should be taken into account when using this cluster.
    - Whether image signatures will be used to sign the OTA upgrade file. If a signature is used, what type of image signature will it be (example: ECDSA).
    - What encryption will be used during the transport of OTA data

- Whether to use offset or UTC time in Image Block Response and Upgrade End Response commands. Refer to section 6.8.4, 6.8.5, 6.10.7 and 6.10.9 for more details. If the application profile does not specify which type of (OTA upgrade) time to support, it is default to using the offset time since it does not require an implementation of ZCL time cluster. Once the application profile has decided which type of time to support, only that type of time shall be used consistently across the OTA upgrading. The profile shall avoid using both types of time simultaneously to avoid any confusion and inconsistency between the two time values.

Other application profile wide decisions that should be answered are:

- How often OTA client shall discovery OTA server until it finds one that is authorized to do the upgrade.

- How often the ZED client shall query OTA server for new OTA upgrade image.

- How often the ZED devices shall query for image data.

6.13.1 SE Profile: OTA Upgrade from SE 1.x to SE 2.0

The definition of SE Profile 2.0 is currently still being worked on by the ZigBee SE group. However, it is suggested that in order to successfully upgrade a device from SE 1.x to SE 2.0, such process may involve transferring new security data over-the-air from the server to the client device. OTA Upgrade cluster has provided a set of commands that may be used to obtain such security data. The security data will be requested separately by the client using Query Specific File Request command. The data is sent from the server to the client as an OTA upgrade file via similar set of commands used to request firmware image. This OTA security file will be specific to a particular client device.

A client should request new security data necessary for SE Profile 2.0 via Query Specific File Request command. After obtaining the security data file, the server will include the file information in the Query Specific File Response command in response to the client's request. Upon reception the response, the client then shall obtain the file via Image Block or Page Request command. Query Specific File Request and Response commands are described in section 6.10.11 and 6.10.12 respectively.

6.14 OTA Upgrade Recovery

Each manufacturer is encouraged to implement a recovery method that should be used to recover the node in a case when the OTA upgrade fails. The recovery method is particularly important in a case where the device may not be able to communicate to the server over-the-air. The actual recovery implementation is manufacturer specific, however, some of the options are discussed in this section.

One option for recovery method is the ability for the application bootloader to swap the images between its external flash and its internal flash, rather than just overwriting the internal with the external. A sample use case is where the upgraded device is functional enough to receive a message, but broken enough to not be able to initiate OTA upgrade process again. A manufacturer specific command may be sent from the server to notify the device to revert back to its previous image.

In a case where the device is no longer able to communicate to the server over-the-air; the application bootloader could revert to the previous image via a button press on power up.