Mesh Firmware Update Proposal

***Bluetooth*® Specification**

- **Revision:** d05r05
- **Revision Date:** 2017-Feb-28
- **Group Prepared By:** Mesh Working Group
- **Feedback Email:** mesh-main@bluetooth.org

Abstract:

This Bluetooth specification defines fundamental requirements to enable the update of firmware over a Bluetooth mesh network for use with Mesh Profile Specification v1.0 using Mesh Object Transport

Revision History

Revision Number	Date	Comments
d05r00	11/30/2017	Initial revision
d05r01	2018-Jan-09	Initial Draft. Mesh Object Transfer not yet extracted
d05r02	2018-Jan-15	Implement changes agreed during CC
d05r03	2018-Jan-19	Move Image Size field to Object Transfer Start message, make Block Size field optional in Object Block Transfer Start, add Firmware Update Prepare message in model definition. Added Firmware Update Abort message. Added missing opcodes values. Renamed Object Transfer Stop to Object Transfer Abort. Change Chunk Size Log to Chunk Size. Added new status codes and two one-byte opcodes. Added Object ID to Object Block Get. Added Object ID field to Object Block Transfer Start. Changed Firmware Update Start: add Company ID and Firmware ID, remove Object ID.
d05r04	2018-Jan-19	All changes accepted
d05r05	2018-Feb-28	Merged the flow diagrams, texts and command codes into the single document

Contributors

Name	Company
Krzysztof Tkaczenko	Silvair, Inc
Piotr Winiarczyk	Silvair, Inc
Jori Rintahaka	Silicon Laboratories, Inc.
Alok Mittal	STMicroelectronics

Use of this specification is your acknowledgement that you agree to and will comply with the following notices and disclaimers. You are advised to seek appropriate legal, engineering, and other professional advice regarding the use, interpretation, and effect of this specification.

Use of Bluetooth specifications by members of Bluetooth SIG is governed by the membership and other related agreements between Bluetooth SIG and its members, including those agreements posted on Bluetooth SIG's website located at www.bluetooth.com. Any use of this specification by a member that is not in compliance with the applicable membership and other related agreements is prohibited and, among other things, may result in (i) termination of the applicable agreements and (ii) liability for infringement of the intellectual property rights of Bluetooth SIG and its members.

Use of this specification by anyone who is not a member of Bluetooth SIG is prohibited and is an infringement of the intellectual property rights of Bluetooth SIG and its members. The furnishing of this specification does not grant any license to any intellectual property of Bluetooth SIG or its members. THIS SPECIFICATION IS PROVIDED "AS IS" AND BLUETOOTH SIG, ITS MEMBERS AND THEIR AFFILIATES MAKE NO REPRESENTATIONS OR WARRANTIES AND DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, TITLE, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR THAT THE CONTENT OF THIS SPECIFICATION IS FREE OF ERRORS. For the avoidance of doubt, Bluetooth SIG has not made any search or investigation as to third parties that may claim rights in or to any specifications or any intellectual property that may be required to implement any specifications and it disclaims any obligation or duty to do so.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, BLUETOOTH SIG, ITS MEMBERS AND THEIR AFFILIATES DISCLAIM ALL LIABILITY ARISING OUT OF OR RELATING TO USE OF THIS SPECIFICATION AND ANY INFORMATION CONTAINED IN THIS SPECIFICATION, INCLUDING LOST REVENUE, PROFITS, DATA OR PROGRAMS, OR BUSINESS INTERRUPTION, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, AND EVEN IF BLUETOOTH SIG, ITS MEMBERS OR THEIR AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF THE DAMAGES.

If this specification is a prototyping specification, it is solely for the purpose of developing and using prototypes to verify the prototyping specifications at Bluetooth SIG sponsored IOP events. Prototyping Specifications cannot be used to develop products for sale or distribution and prototypes cannot be qualified for distribution.

Products equipped with Bluetooth wireless technology ("Bluetooth Products") and their combination, operation, use, implementation, and distribution may be subject to regulatory controls under the laws and regulations of numerous countries that regulate products that use wireless non-licensed spectrum. Examples include airline regulations, telecommunications regulations, technology transfer controls and health and safety regulations. You are solely responsible for complying with all applicable laws and regulations and for obtaining any and all required authorizations, permits, or licenses in connection with your use of this specification and development, manufacture, and distribution of Bluetooth Products. Nothing in this specification provides any information or assistance in connection with complying with applicable laws or regulations or obtaining required authorizations, permits, or licenses.

Bluetooth SIG is not required to adopt any specification or portion thereof. If this specification is not the final version adopted by Bluetooth SIG's Board of Directors, it may not be adopted. Any specification adopted by Bluetooth SIG's Board of Directors may be withdrawn, replaced, or modified at any time. Bluetooth SIG reserves the right to change or alter final specifications in accordance with its membership and operating agreements.

Contents

1 Introduction

This Bluetooth specification defines fundamental requirements to enable an update of the firmware over the Mesh network for use with Mesh Profile Specification v1.0 [1]. The Mesh Profile specification defines the usage of the embedded devices in the Mesh network running a firmware. This specification defines the process required to update the firmware on the embedded devices in the network.

The firmware update can be for complete solution or part of it like adding or updating a specific model support, the Mesh core libraries, parameters of the interface or any partial update.

1.1 Conformance

If conformance to this specification is claimed, all capabilities indicated as mandatory for this specification shall be supported in the specified manner (process-mandatory). This also applies for all optional and conditional capabilities for which support is indicated.

1.2 Bluetooth specification release compatibility

This specification shall be used with the Bluetooth Mesh Profile Specification Version 1.0 [1].

1.3 Language

1.3.1 Language conventions

The Bluetooth SIG has established the following conventions for use of the words ***shall***, ***must***, ***will***, ***should***, ***may***, ***can***, ***is***, and ***note*** in the development of specifications:

shall	is required to – used to define requirements
must	is a natural consequence of – used only to describe unavoidable situations
will	it is true that – only used in statements of fact
should	is recommended that – used to indicate that among several possibilities one is recommended as particularly suitable, but not required
may	is permitted to – used to allow options
can	is able to – used to relate statements in a causal manner
is	is defined as – used to further explain elements that are previously required or allowed
note	Used to indicate text that is included for informational purposes only and is not required in order to implement the specification. Informative text in a note continues to the end of the paragraph.

For clarity of the definition of those terms, see Core Specification Volume 1, Part E, Section 1.

1.3.1.1 Reserved for Future Use

Where a field in a packet, Protocol Data Unit (PDU), or other data structure is described as "Reserved for Future Use" (irrespective of whether in uppercase or lowercase), the device creating the structure shall set its value to zero unless otherwise specified. Any device receiving or interpreting the structure shall ignore that field; in particular, it shall not reject the structure because of the value of the field.

Where a field, parameter, or other variable object can take a range of values and some values are described as "Reserved for Future Use," a device sending the object shall not set the object to those values. A device receiving an object with such a value should reject it, and any data structure containing it, as being erroneous; however, this does not apply in a context where the object is described as being ignored or it is specified to ignore unrecognized values.

When a field value is a bit field, unassigned bits can be marked as Reserved for Future Use and shall be set to 0. Implementations that receive a message that contains a Reserved for Future Use bit that is set to 1 shall process the message as if that bit was set to 0, except where specified otherwise.

The acronym RFU is equivalent to "Reserved for Future Use."

1.3.1.2 Prohibited

When a field value is an enumeration, unassigned values can be marked as “Prohibited.” These values shall never be used by an implementation, and any message received that includes a Prohibited value shall be ignored and shall not be processed and shall not be responded to.

Where a field, parameter, or other variable object can take a range of values, and some values are described as “Prohibited,” devices shall not set the object to any of those Prohibited values. A device receiving an object with such a value should reject it, and any data structure containing it, as being erroneous.

“Prohibited” is never abbreviated.

1.3.2 Acronyms and abbreviations

Acronym or Abbreviation	Meaning
BIC	Block Integrity Check
CID	Company Identifier
DFU	Device firmware update
FOTA	Firmware update Over The Air
FWB	Firmware Block: Part of the firmware
FIC	Firmware Integrity Check
ID	Identifier
IVI	Initialization Vector Index
MCU	Micro-controller unit
OTA	Over The Air Update (Same as FOTA)

Acronym or Abbreviation	Meaning
OTD	Over The Air Download (Same as OTA)
PID	Product Identifier
RFU	Reserved for Future Use
RPL	Replay Protection List
SIG	Special Interest Group
SoC	System on Chip
TTL	Time To Live
UUID	Universally Unique Identifier
VID	Version Identifier

Table 1.1: Acronyms and abbreviations

1.3.3 Terminology

The terminology used throughout this specification is defined in Mesh Profile Specification Version 1.0 [1] unless otherwise stated.

In addition, following terminology is used in the document

- Initiating node: The node that determines that a firmware update is required by caching and comparing metadata, and that transfers the appropriate firmware image to the appropriate mesh devices (either the Updating node or an intermediate Distributor node) through one or more hops and monitors the progress of the transfer.
- Distributor node: An optional intermediate node that receives the firmware download on behalf of an Initiating node or other Distributor nodes and transfers the firmware image to either another Distributor node or the Updating node and monitors and reports progress.
- Provider node: The node which provides the firmware update to the updating node. It is either of the Initiating node or the Distributor node.
- Updating node: The node that requires and receives an updated firmware image and reports progress.

2 Mesh Firmware Update actors

The mesh firmware update is done through sequence of commands from the Provider node (Initiating node or Distributor node) to the Updating nodes. The Provider node provides the new firmware update image to the Updating node and sends the confirmation command to switch to the updated firmware.

2.1 Firmware provider

The firmware needs to be updated from a firmware update provider. This Initiating node can be the smart-phone or a Gateway device which might periodically check on the product website if a new firmware update is available.

The Provider node updates the latest binary of the firmware that needs to be deployed in some or all nodes of the network depending on the capability of the nodes. It is possible that some nodes can be updated and others cannot be updated.

2.2 Check for firmware update capability

The Provider node needs to know if the nodes in the network have the firmware update capability. It is possible that some nodes in the network have this capability and some other nodes do not have the capability and they continue to work on the pre-programmed firmware.

This information has to be read from the mesh node. The support for the firmware update “model” can be known from the composition data of the node.

Having firmware update as a Model helps firmware provider to send messages in seam-less manner keeping the other functionalities of the node to work at the same time. For example, a Lighting node can keep on working for its intended use while the firmware provider is downloading the new update to the node.

2.3 Check for the current version

The firmware provider needs to know the manufacturer identifier, the product identifier, and the firmware version running on the product. If a new firmware version is available for a combination of Company Identifier (CID) and Product Identifier (PID), the firmware provider compares it with the current firmware version (VID) running on the product. To deploy the firmware, the available VID shall be a later version than the VID currently running on the product.

For reference, the information is available from the Composition Data Page0. This composition information contains the following:

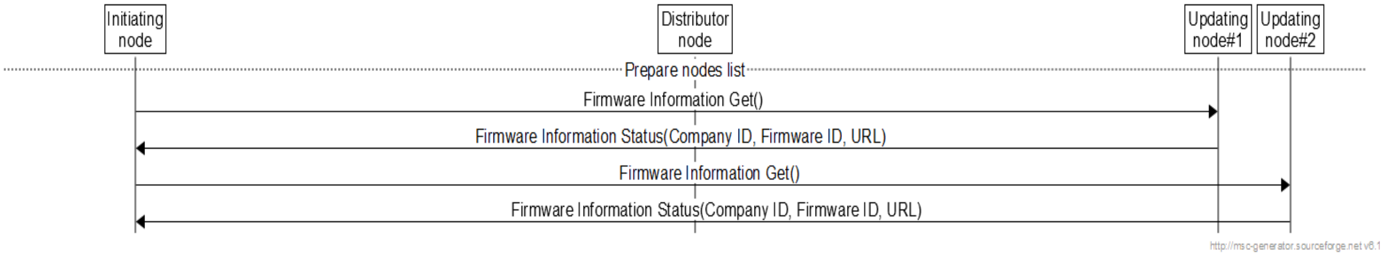
- CID
- PID
- VID

The CID contains a 16-bit Company Identifier assigned by the Bluetooth SIG [3].

The PID contains a 16-bit vendor-assigned Product Identifier.

The VID property contains a 16-bit vendor-assigned firmware version identifier.

The firmware information can be received by polling the nodes to be updated. The nodes can provide the URL for the firmware binary image which can be checked by the Initiating or distributor node to check if an updated version is available


Initiating node
Distributor node
Updating node#1
Updating node#2
Prepare nodes list
Firmware Information Get()
Firmware Information Status(Company ID, Firmware ID, URL)
Firmware Information Get()
Firmware Information Status(Company ID, Firmware ID, URL)
http://msc-generator.sourceforge.net v6.1

Note: The diagram may need an update for the VID information reply from the updating node. To be discussed.

2.4 Internal Memory Management of Mesh Node

The Bluetooth Mesh nodes may be running on SoC, microcontrollers or any computing unit. They will have their own memory management depending on the architecture of the computing unit for storing the firmware.

The executable file format will depend on the internal architecture of the node device. The format of executable will depend on the manufacturer’s and product’s identifiers.

In addition there can be external memories which can store the firmware during the download. This temporary storage can then be verified for validating the correct download by verifying the firmware integrity check of the binary. Once the binary download is verified, the downloaded firmware can be programmed into the Bluetooth Mesh device.

The internal architecture of the mesh node and the way of addressing the memory are out of scope of this specification. However, the start memory address and the size of the firmware to download are critical for the download process.

If there are multiple memories in a mesh node, the firmware of the node need to ensure that all memories are assigned unique non-overlapping addresses

2.5 Format of firmware binaries

The format of firmware binaries is out of scope of this document. It can be any binary format useful to carry information of the firmware program depending on the SoC or microcontroller which is supporting the Bluetooth radio communication

The firmware update binary which is sent over the mesh is dependent on the vendor and the product type. It can be raw data or compressed data which the mesh node is able to receive and update for DFU.

It is also possible that the node can accept multiple formats of the firmware download binary. For example it can accept a specific format for programming the execution memory or a different format for other updates like model library or some other parameters as for configuration of sensor calibration values available on the Mesh node.

The format of binaries can be of any type, for example the following ones:

a. Hex format

b. S19 format

c. Raw binary

2.6 Partial or full firmware update

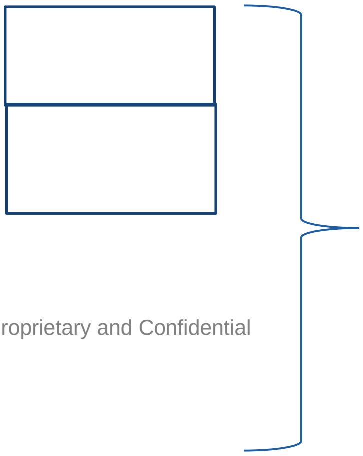
It is possible that the Bluetooth Mesh node do not need a complete update of the firmware. Maybe in certain cases only certain blocks of the firmware of the Mesh node need an update.

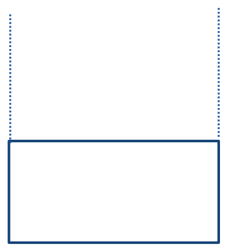
For example, it can be some calibration information of the sensor nodes which needs an update. Also, a bug seen in any sensor software driver is discovered and the vendor prefers updating only portions of the firmware associated with that driver.

Such updates can be for re-writing some parameters inside the memory (EEPROM/Flash) which store the configuration of certain parameters.

In many cases, the complete firmware update may be needed for the Mesh node. In this case, the complete binary will be sent by the firmware provider to the node.

The binary which needs to be updated in the nodes will be split into smaller blocks of update. Each block of update will contain the start address and size information for that block of update





Complete Firmware image divided into blocks

Figure: Firmware is divided into several blocks

Each block of the firmware has the following information:

a. Block number
b. Start address
c. Size of the block
d. Block integrity check value

The DFU or blocks update from initiating or distributor nodes will use Mesh Object Transfer.for sending firmware-binary object.

2.7 Unicast or Group subscription of nodes under update

The firmware update for a single node can be done by sending the Mesh Firmware update messages to the addressed node. This node address is a unicast address when only one node is under update.

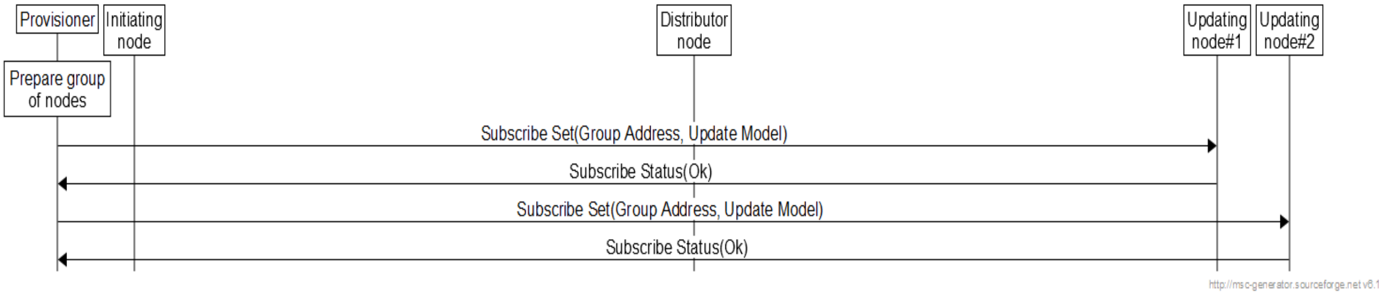
The mesh network may have several nodes from different manufacturers due to interoperability. However, as discussed in the previous sections, the new firmware update can be applied only to the applicable nodes which have the same manufacturer identifier (CID) and product identifier (PID).

For the mesh network, when more than one nodes from the same manufacturer (CID) and having the same Product identifier (PID) need to be updated to a new version of the firmware, the mesh network capabilities need to be used. This can be done by grouping these nodes together for the firmware update. This will help to address only the nodes under firmware update by using the same group subscription address.

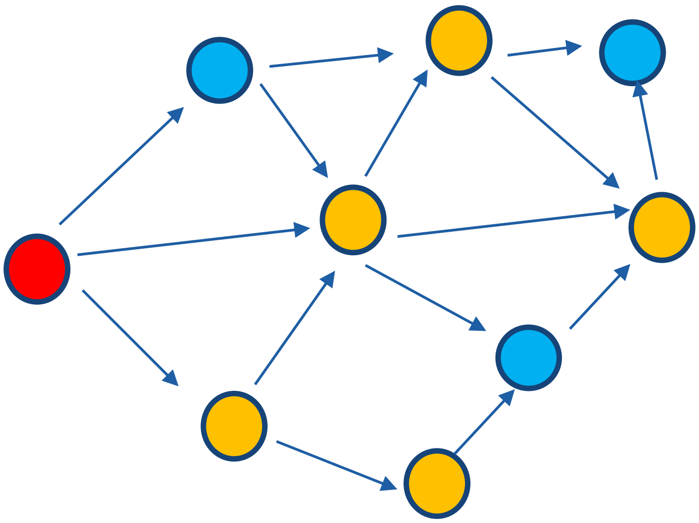
The efficiency of this kind of firmware update is always higher than updating each node separately.

A node checks if a DFU message received is directed to a group address it subscribed to at the access layer, after the message is decrypted and authenticated using the application key. The nodes which are not under update will not be part of the group subscription under firmware update

For a network which has all nodes from the same manufacturer and with the same product identifier, the firmware update can still be done by creating the group for all the nodes or using already available groups where all nodes to be updated are the members.



Following diagram shows an illustration of the network which may have few nodes grouped together for the firmware update



🔴 Provider Node

🟡 Node with same CID, PID of type X design

🔵 Node with same CID, PID of type Y design. These nodes will be grouped together to do the firmware update

2.8 Agreement of firmware update

The firmware provider and the target node of the firmware update process need to agree on the firmware update phase. This involves sending commands to the node for the firmware update request. The command for update will have information on CID, PID, and new VID.

The node can reject the firmware update if the CID, PID is not matching the already running firmware. Please note that the CID and PID cannot be changed in the firmware update. Only the VID can be updated after the firmware update.

The node can agree on the firmware upgrade request if it is ready to receive the new firmware from the provider for the CID, PID and new VID.

The provider needs to send the start address and size of the complete download. The message will also contain the information on the format of the downloaded binary. The node will verify it is able to accept the upgrade request for the address and the size in the message. All these messages are acknowledged messages. The firmware download may be split into a few blocks of the download.

The block size of the firmware download will be agreed between the Provider node and the Updating node. This block size may be equal to the maximum size of a segmented packet in the mesh network or a smaller size depending on the internal memory organization of the updating nodes. The Updating node will provide the maximum size of the block it supports. The provider will send the firmware blocks which are less than or equal to the maximum block size supported by the Updating node. In all cases, the Provider node knows in advance the block size of the Updating node corresponding to the CID and PID of the node.

If the Updating node does not have any space to receive the new firmware image blocks, it rejects the firmware update request.

If the addressed Updating node is a LPN, a Friend node may respond to the firmware update request from the Provider node on behalf of the befriended LPN.

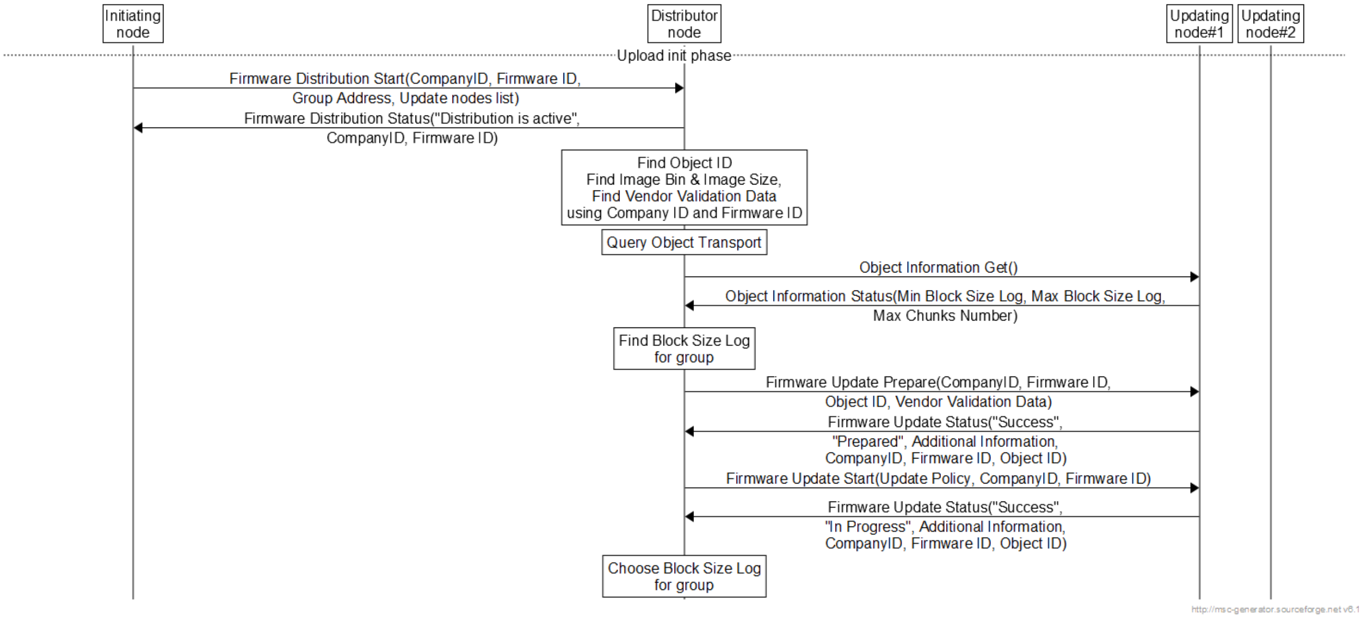
Firmware update process is based on the Mesh Object Transfer feature.

Firmware Distribution Start; Prepare and Start phases (shown only for one node for clarity)
It is assumed that distributor has firmware binary stored

Following diagram shows the communication between Initiating node, distributor node and Updating node.

The Initiating node and Distributor node communicate for the start of DFU process.

The Distributor node gets the information on the Block size support from the Updating node. Once the information is available, the Distributor node sends the Prepare command to the nodes followed by the DFU Start command. Each command is duly acknowledged.


Initiating node
Distributor node
Updating node#1
Updating node#2
Upload init phase
Firmware Distribution Start(CompanyID, Firmware ID, Group Address, Update nodes list)
Firmware Distribution Status("Distribution is active", CompanyID, Firmware ID)
Find Object ID
Find Image Bin & Image Size,
Find Vendor Validation Data
using Company ID and Firmware ID
Query Object Transport
Object Information Get()
Object Information Status(Min Block Size Log, Max Block Size Log, Max Chunks Number)
Find Block Size Log for group
Firmware Update Prepare(CompanyID, Firmware ID, Object ID, Vendor Validation Data)
Firmware Update Status("Success", "Prepared", Additional Information, CompanyID, Firmware ID, Object ID)
Firmware Update Start(Update Policy, CompanyID, Firmware ID)
Firmware Update Status("Success", "In Progress", Additional Information, CompanyID, Firmware ID, Object ID)
Choose Block Size Log for group
http://msc-generator.sourceforge.net v6.1

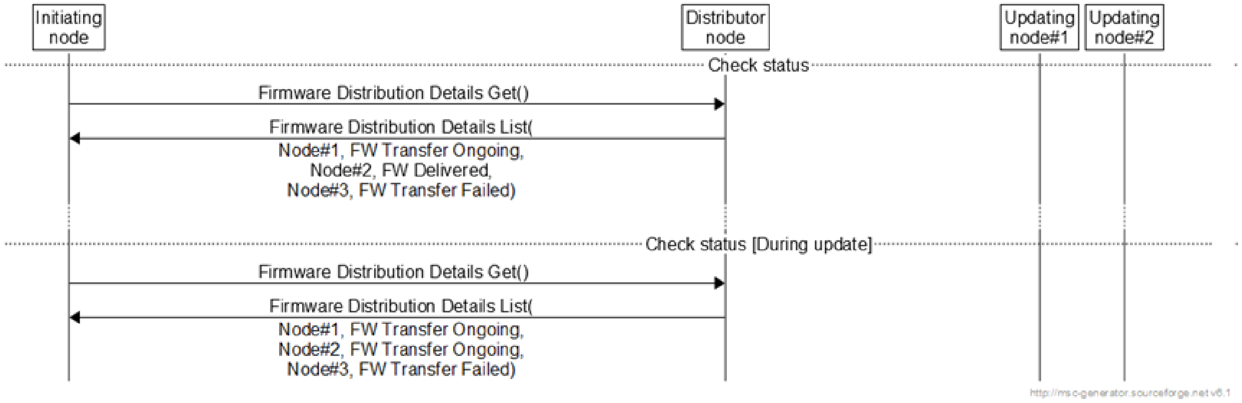
2.9 Check for download progress status

When the firmware download is ongoing, the Distributor node verifies the status of the download of the firmware to the Updating node by checking the status of the block downloads.

This check can be done after every block of new binary (FWB) download. This can be done by sending a message to the node to verify the firmware download integrity check value (FIC) and the firmware block integrity check value (BIC).

The Provider node can query the Distributor node about the status of the download process and progress of this download. This may be required to show the progress on a user interface or smart-phone application.

The Distributor node can provide the summary of the progress of each node to the Initiating node

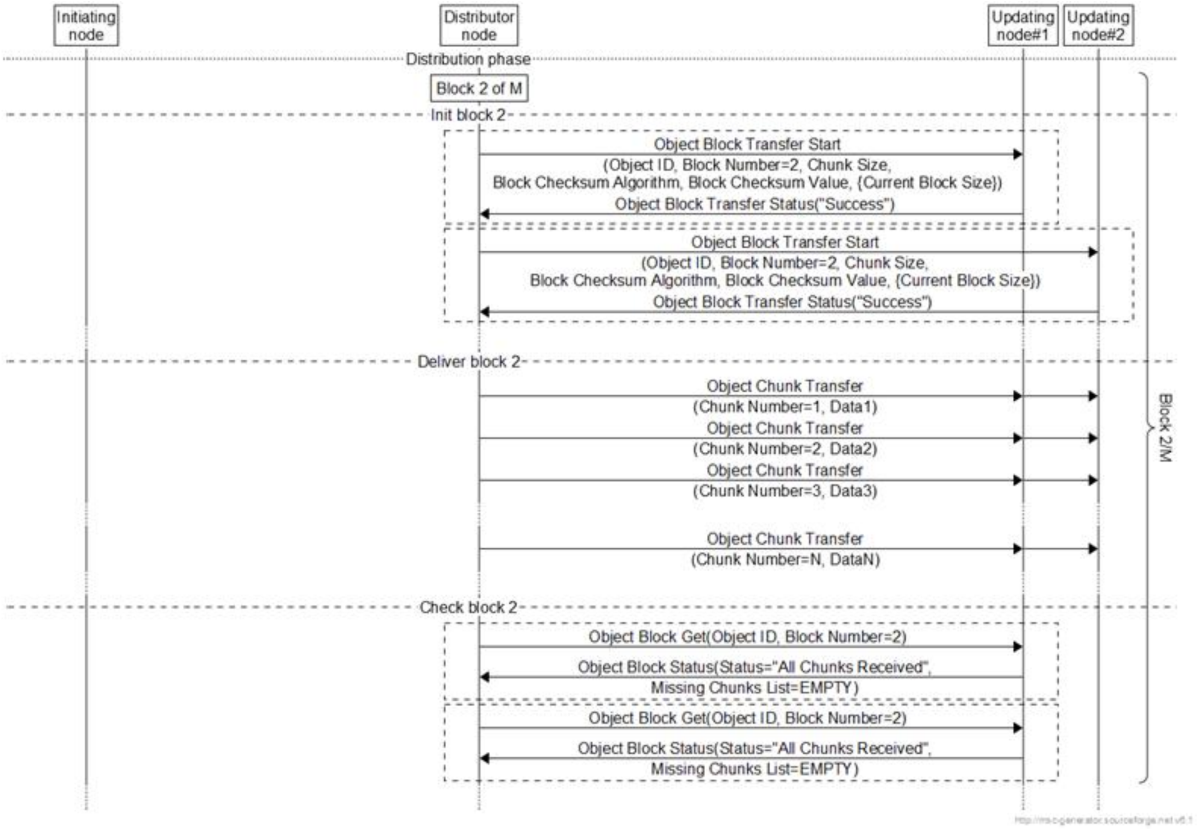

Initiating node
Distributor node
Updating node#1
Updating node#2
Check status
Firmware Distribution Details Get()
Firmware Distribution Details List(
Node#1, FW Transfer Ongoing,
Node#2, FW Delivered,
Node#3, FW Transfer Failed)
Check status [During update]
Firmware Distribution Details Get()
Firmware Distribution Details List(
Node#1, FW Transfer Ongoing,
Node#2, FW Transfer Ongoing,
Node#3, FW Transfer Failed)
http://msc-generator.sourceforge.net v6.1

The Updating node will compute the Block Integrity check (BIC) of the downloaded firmware blocks and reply back to the Distributor node when the status is requested. The Distributor node can then verify the correctness of the download process. If the BIC is matching the firmware provider’s one, the downloaded segment will be treated as successful and the Distributor node can continue to send the other blocks of the download.

Each block may consist of several chunks of the data. Several chunks of data will be updated from Distributor node to the Updating node. Once a block is sent from the Distributor, the status command can be issued to the nodes in unicast addressing. The updating node will provide the information on which chunks are correctly received by it and which are missing. Based on missing chunks information, the Distributor node may send the missed chunks again to the nodes. This cycle may continue till all chunks of the block is successfully updated in the node.

The check of block update helps to handle the partial download which may be interrupted due to either the updating node going out of range or accidently powered off. The check of blocks transfer helps to not start the firmware upgrade process from the beginning but to continue from the last successful download.

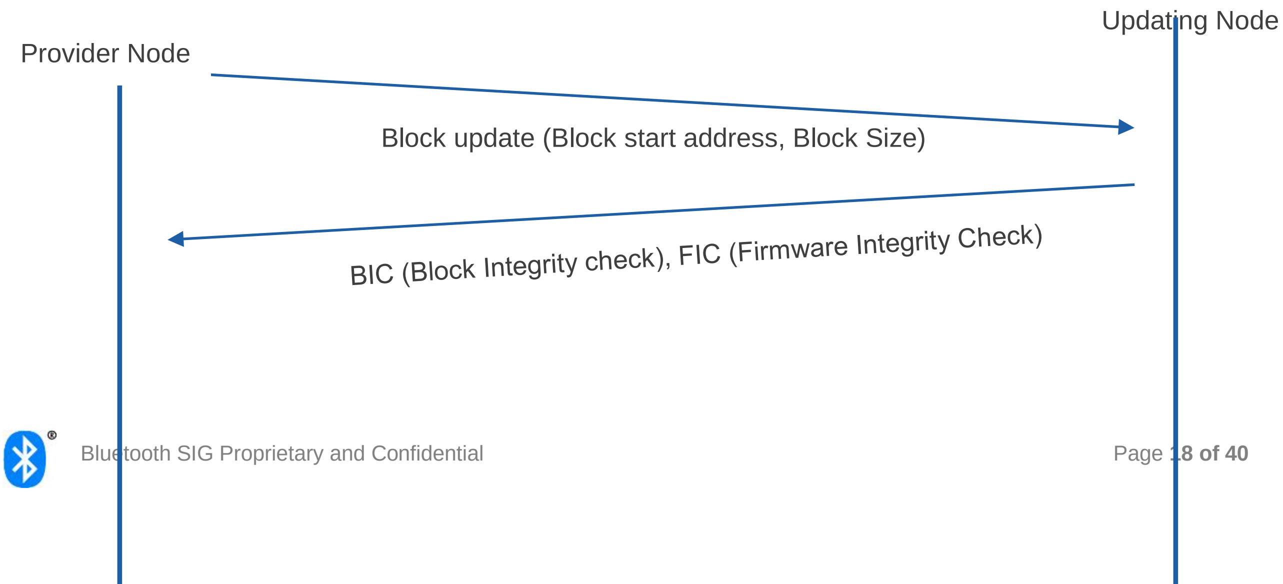
The firmware provider node will wait for the feedback from the Updating node for certain number of retrials. If the Updating node does not provide the feedback for certain number of enquiries from the Provider node, the update to that node will be aborted. The Provider node will continue to provide the newer firmware blocks to other nodes which correctly provide the feedbacks to the Provider node.


Initiating node
Distributor node
Updating node#1
Updating node#2
Distribution phase
Block 2 of M
Init block 2
Object Block Transfer Start
(Object ID, Block Number=2, Chunk Size,
Block Checksum Algorithm, Block Checksum Value, {Current Block Size})
Object Block Transfer Status("Success")
Object Block Transfer Start
(Object ID, Block Number=2, Chunk Size,
Block Checksum Algorithm, Block Checksum Value, {Current Block Size})
Object Block Transfer Status("Success")
Deliver block 2
Object Chunk Transfer
(Chunk Number=1, Data1)
Object Chunk Transfer
(Chunk Number=2, Data2)
Object Chunk Transfer
(Chunk Number=3, Data3)
Object Chunk Transfer
(Chunk Number=N, DataN)
Check block 2
Object Block Get(Object ID, Block Number=2)
Object Block Status(Status="All Chunks Received",
Missing Chunks List=EMPTY)
Object Block Get(Object ID, Block Number=2)
Object Block Status(Status="All Chunks Received",
Missing Chunks List=EMPTY)
Block 2/M
http://msc-generator.sourceforge.net v6.1

2.10 Check for correct download

All mesh messages are protected by the message integrity check. However, it is required by the Provider node to verify the firmware download is ongoing correctly. This check can be done after every block of new binary (FWB) download as explained in previous sections. For the complete firmware download, there is a Firmware integrity check value (FIC).

The FIC may be required after all blocks are successfully downloaded. So, in addition to the block integrity checks, the FIC helps to verify the completeness of the firmware update.

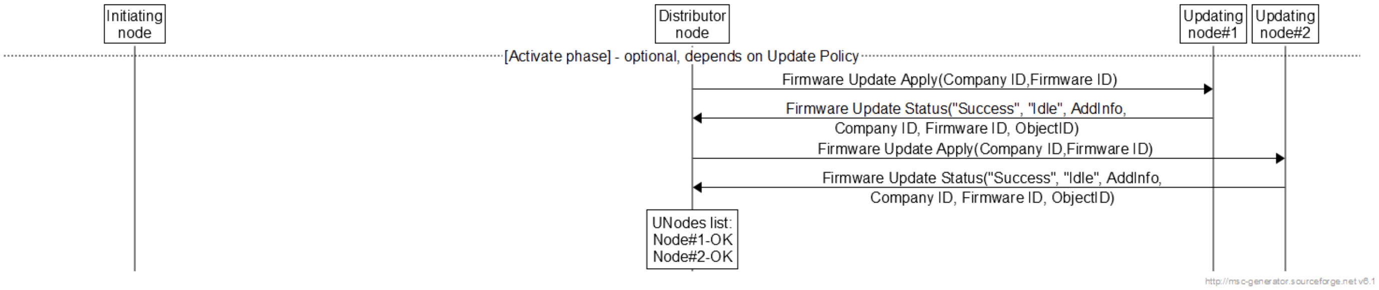

Updating Node
Provider Node
Block update (Block start address, Block Size)
BIC (Block Integrity check), FIC (Firmware Integrity Check)

2.11 Switching to updated firmware

The switching to the updated firmware can be done by the firmware update client by issuing a message to the nodes under update after confirmation of correct download of the firmware. This can be done using “Firmware Update Apply” command

Once the Distributor node confirms for each Updating node the firmware update integrity check, the Distributor node can issue unicast command to each Updating to let them upgrade to the new binary downloaded. The updating node can confirm the status of firmware upgrade back to the Distributor node.

The Distributor node maintains the DFU status of each node and can provide this status back to Initiating/ Provider node when required.


Initiating node
Distributor node
Updating node#1
Updating node#2
[Activate phase] - optional, depends on Update Policy
Firmware Update Apply(Company ID,Firmware ID)
Firmware Update Status("Success", "Idle", AddInfo, Company ID, Firmware ID, ObjectID)
Firmware Update Apply(Company ID,Firmware ID)
Firmware Update Status("Success", "Idle", AddInfo, Company ID, Firmware ID, ObjectID)
UNodes list: Node#1-OK Node#2-OK
http://msc-generator.sourceforge.net v6.1

3 Firmware Update Server model

3.1 Numerical summary of opcodes

3.1.1 Firmware Update Messages

Message name	Opcode
Firmware Information Get	0xB6 0x01
Firmware Information Status	0xB6 0x02
Firmware Update Get	0xB6 0x03
Firmware Update Prepare	0xB6 0x04
Firmware Update Start	0xB6 0x05
Firmware Update Abort	0xB6 0x06
Firmware Update Apply	0xB6 0x07
Firmware Update Status	0xB6 0x08
Firmware Distribution Get	0xB6 0x09
Firmware Distribution Start	0xB6 0x0A
Firmware Distribution Stop	0xB6 0x0B
Firmware Distribution Status	0xB6 0x0C
Firmware Distribution Details Get	0xB6 0x0D
Firmware Distribution Details List	0xB6 0x0E

3.1.2 Object Transfer Messages

Message name	Opcode
Object Transfer Get	0xB7 0x01
Object Transfer Start	0xB7 0x02

Object Transfer Abort	0xB7 0x03
Object Transfer Status	0xB7 0x04
Object Block Transfer Start	0xB7 0x05
Object Block Transfer Status	0xB7 0x06
Object Chunk Transfer	0x7D
Object Block Get	0x7E
Object Block Status	0xB7 0x09
Object Information Get	0xB7 0x0A
Object Information Status	0xB7 0x0B

3.2 Firmware Update Messages

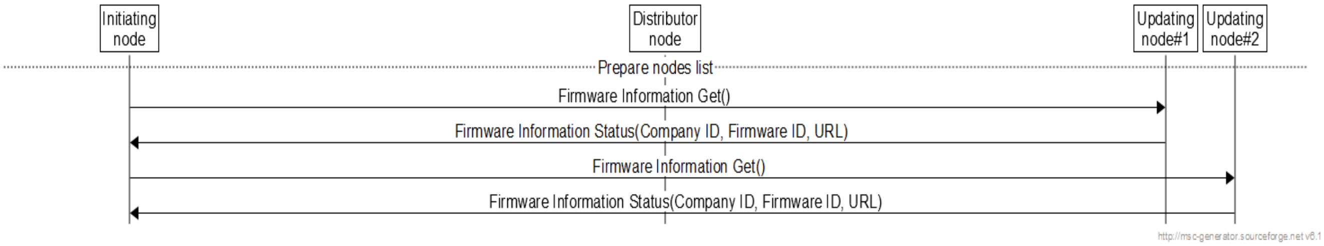
3.2.1 Firmware Information Get

The Firmware Information Get is an acknowledged message used to get the current firmware information of a node. The response to the Firmware Information Get message is a Firmware Information Status message. There are no parameters for this message.

3.2.2 Firmware Information Status

The Firmware Information Status is an unacknowledged message used to report the current firmware information of a node. The structure of the Firmware Information Status message is defined in the table below.

Field	Size (bytes)	Notes
Company ID	2	Company identifier
Firmware ID	1+N	Unique firmware identifier
Update URL	N	URL for update source (optional)


Initiating node
Distributor node
Updating node#1
Updating node#2
Prepare nodes list
Firmware Information Get()
Firmware Information Status(Company ID, Firmware ID, URL)
Firmware Information Get()
Firmware Information Status(Company ID, Firmware ID, URL)
http://msc-generator.sourceforge.net v6.1

3.2.3 Firmware Distribution Get

The Firmware Distribution Get is an acknowledged message used to get the state of the firmware distribution process of the firmware distributor node for a given firmware. The response to the Firmware Distribution Get message is a Firmware Distribution Status message. The structure of the Firmware Distribution Get message is defined in the table below.

Field	Size (bytes)	Notes
Company ID	2	Company identifier
Firmware ID	1+N	Unique firmware identifier

3.2.4 Firmware Distribution Start

The Firmware Distribution Start is an acknowledged message used to start the firmware distribution to the group of the nodes. The response to the Firmware Distribution Start message is a Firmware Distribution Status message. The structure of the Firmware Distribution Start message is defined in the table below.

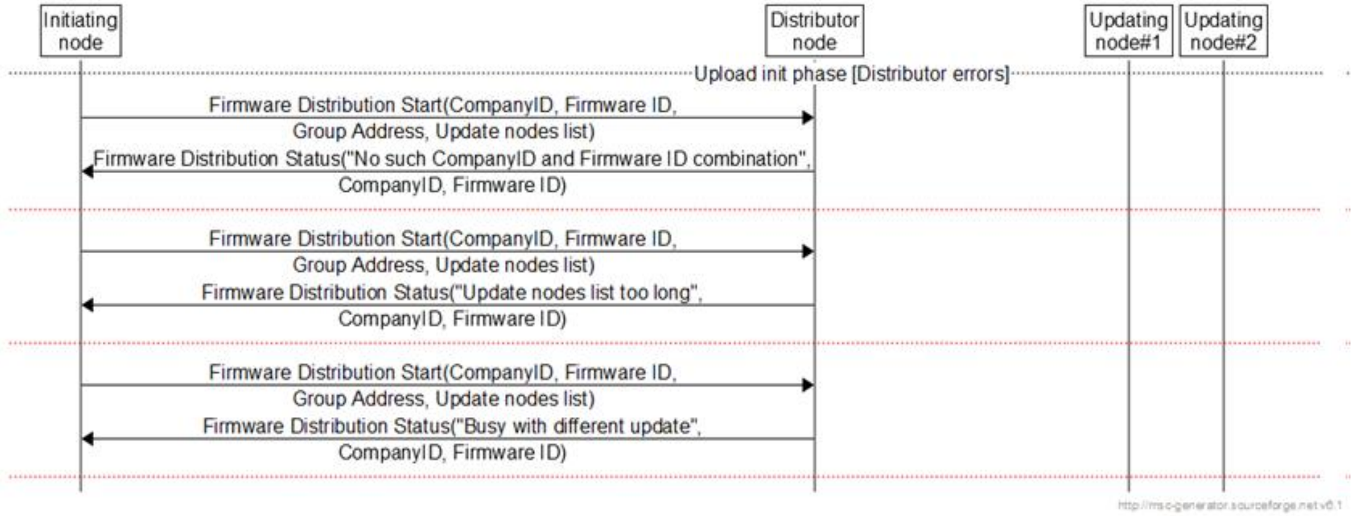
Field	Size (bytes)	Notes
Company ID	2	Company identifier
Firmware ID	1+N	Unique firmware identifier
Group address	2	Group address for multicast update mode
Update nodes list	2 * update nodes list size	Update nodes list

Error handling

It is possible that there is some error in the firmware update process. When an Initiating node starts the firmware update process, the following error conditions may occur:

a. The Initiating Node requests a firmware update with a CID that does not match the Updating nodes' one.

b. The Updating nodes are already under update.


Initiating node
Distributor node
Updating node#1
Updating node#2
Upload init phase [Distributor errors]
Firmware Distribution Start(CompanyID, Firmware ID, Group Address, Update nodes list)
Firmware Distribution Status("No such CompanyID and Firmware ID combination", CompanyID, Firmware ID)
Firmware Distribution Start(CompanyID, Firmware ID, Group Address, Update nodes list)
Firmware Distribution Status("Update nodes list too long", CompanyID, Firmware ID)
Firmware Distribution Start(CompanyID, Firmware ID, Group Address, Update nodes list)
Firmware Distribution Status("Busy with different update", CompanyID, Firmware ID)
http://msc-generator.sourceforge.net v6.1

3.2.5 Firmware Distribution Stop

The Firmware Distribution Stop is an acknowledged message used to stop the firmware distribution. The response to the Firmware Distribution Stop message is a Firmware Distribution Status message. The structure of the Firmware Distribution Stop message is defined in the table below.

Field	Size (bytes)	Notes
Company ID	2	Company identifier
Firmware ID	1+N	Unique firmware identifier

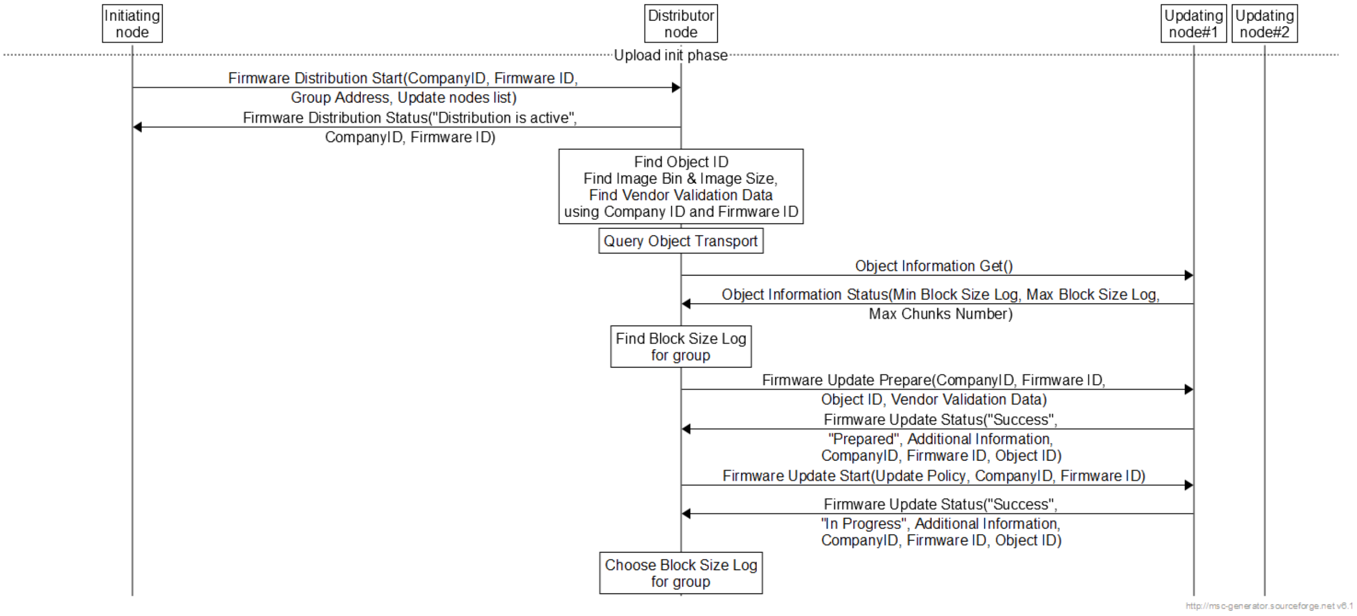
3.2.6 Firmware Distribution Status

The Firmware Distribution Status is an unacknowledged message used to report the state of the firmware distribution process of the firmware distributor node for a given firmware. The structure of the Firmware Distribution Status message is defined in the table below.

Field	Size (bytes)	Notes
Status	1	Status code
Company ID	2	Company identifier
Firmware ID	1+N	Unique firmware identifier

Status values:

- 0x00 - ready, distribution is not active
- 0x01 - distribution is active
- 0x02 - no such Company ID and Firmware ID combination
- 0x03 - busy with different distribution
- 0x04 - update nodes list is too long


Initiating node
Distributor node
Updating node#1
Updating node#2
Upload init phase
Firmware Distribution Start(CompanyID, Firmware ID, Group Address, Update nodes list)
Firmware Distribution Status("Distribution is active", CompanyID, Firmware ID)
Find Object ID Find Image Bin & Image Size, Find Vendor Validation Data using Company ID and Firmware ID
Query Object Transport
Object Information Get()
Object Information Status(Min Block Size Log, Max Block Size Log, Max Chunks Number)
Find Block Size Log for group
Firmware Update Prepare(CompanyID, Firmware ID, Object ID, Vendor Validation Data)
Firmware Update Status("Success", "Prepared", Additional Information, CompanyID, Firmware ID, Object ID)
Firmware Update Start(Update Policy, CompanyID, Firmware ID)
Firmware Update Status("Success", "In Progress", Additional Information, CompanyID, Firmware ID, Object ID)
Choose Block Size Log for group
http://msc-generator.sourceforge.net v6.1

3.2.7 Firmware Distribution Details Get

The Firmware Distribution Details Get is an acknowledged message used to get the current status of the firmware update node list. The response to the Firmware Distribution Details Get message is a Firmware

Distribution Details Status message. The structure of the Firmware Distribution Details Get message is defined in the table below.

Field	Size (bytes)	Notes
Status	1	Status code
Company ID	2	Company identifier
Firmware ID	1+N	Unique firmware identifier

3.2.8 Firmware Distribution Details List

The Firmware Distribution Details List is an unacknowledged message used to report the current status of the firmware update node list. The structure of the Firmware Distribution Details List message is defined in the table below.

Status	1	Notes
Node #1 address	2	Node #1 unicast address
Node #1 update status	1	Node #1 update status code
..	..	..
Node #N address	2	Node #N unicast address
Node #N update status	1	Node #N update status code

Update status code values:

- 0x00 - successfully updated
- 0x01 - in progress
- 0x02 - canceled

3.2.9 Firmware Update Get

The Firmware Update Get is an acknowledged message used to get the current status of the firmware update process. The response to the Firmware Update Get message is a Firmware Update Status message. The structure of the Firmware Update Get message is defined in the table below.

Field	Size (bytes)	Notes
Company ID	2	Company identifier
Firmware ID	1+N	Unique firmware identifier

3.2.10 Firmware Update Prepare

The Firmware Update Prepare is an acknowledged message used to start the firmware update process. The response to the Firmware Update Prepare message is a Firmware Update Status message. The structure of the Firmware Update Prepare message is defined in the table below.

Field	Size (bytes)	Notes
Company ID	2	Company identifier
Firmware ID	1+N	Unique firmware identifier
Object ID	8	Unique object identifier
Vendor validation data	0...256	Vendor specific validation data for update (optional)

3.2.11 Firmware Update Start

The Firmware Update Start is an acknowledged message used to start the firmware update process. The response to the Firmware Update Start message is a Firmware Update Status message. The structure of the Firmware Update Start message is defined in the table below.

Field	Size (bytes)	Notes
Update Policy	1	Firmware update policy
Company ID	2	Company identifier
Firmware ID	1+N	Unique firmware identifier

The Update Policy field values are defined in table below.

Value	Update Policy	Notes
0x00	None	Do not apply new firmware when Object transfer is completed.

0x01	Auto Update	Apply new firmware when Object transfer is completed.
0x02- 0xFF	RFU	Reserved for Future Use

3.2.12 Firmware Update Abort

The Firmware Update Abort is an acknowledged message used to abort the firmware update and delete any stored information about the update. The response to the Firmware Update Abort message is a Firmware Update Status message. The structure of the Firmware Update Abort message is defined in the table below.

Field	Size (bytes)	Notes
Company ID	2	Company identifier
Firmware ID	1+N	Unique firmware identifier

3.2.13 Firmware Update Apply

The Firmware Update Apply is an acknowledged message used to apply the new firmware stored in the device. The response to the Firmware Update Apply message is a Firmware Update Status message. The structure of the Firmware Update Apply message is defined in the table below.

Field	Size (bytes)	Notes
Company ID	2	Company identifier
Firmware ID	1+N	Unique firmware identifier

3.2.14 Firmware Update Status

The Firmware Update Status is an unacknowledged message used to report the current state of the firmware update process. The structure of the Firmware Update Status message is defined in the table below.

Field	Size (bits)	Notes
Status	8	Status code of the operation
Phase	3	Phase of the update

Additional Information	5	Bitfield with additional information
Company ID	16	Company identifier
Firmware ID	8+N*8	Unique firmware identifier
Object ID	64	Unique object identifier (optional)

The Phase field values are defined in table below:

Phase Code	Phase	Notes
0x00	Idle	No DFU update in progress
0x01	Prepared	DFU update is prepared and awaiting start
0x02	In Progress	DFU update is in progress
0x03	DFU Ready	DFU upload is finished and waiting to be applied
0x4-0x7	RFU	Reserved for Future Use

The Additional Information bitfield is defined in table below:

Bit	Information	Notes
0	ProvisioningNeeded	When set to 1 the device be in unprovisioned state after the new FW is applied (possibly due to new models added).
1-4	RFU	Reserved for Future Use

The Status filed values are defined below:

- 0x00 - success
- 0x01 - wrong Company ID and Firmware ID combination
- 0x02 - different object transfer already ongoing

- 0x03 - Company ID and Firmware ID combination apply failed
- 0x04 - Company ID and Firmware ID combination permanently rejected, newer firmware version present
- 0x05 - Company ID and Firmware ID combination temporary rejected, node is not able to accept new firmware now, try again later

Object ID present only when firmware object transfers just started or is ongoing.

3.3 Object Transfer Messages

3.3.1 Object Transfer Get

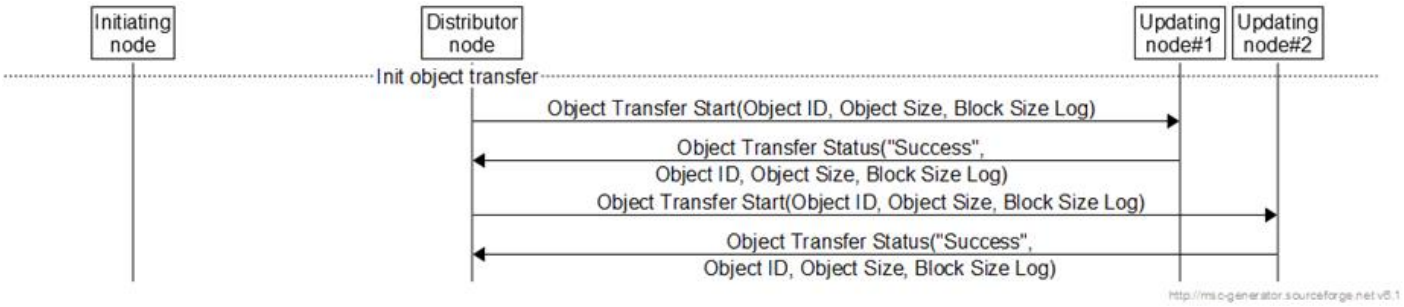
The Object Transfer Get is an acknowledged message used to report the current state of the object transfer process. The response to the Object Transfer Get message is an Object Transfer Status message. The structure of the Object Transfer Get message is defined in the table below.

Field	Size (bytes)	Notes
Object ID	8	Unique object identifier

3.3.2 Object Transfer Start

The Object Transfer Start is an acknowledged message used to start new object transfer process. The response to the Object Transfer Start message is an Object Transfer Status message. The structure of the Object Transfer Start message is defined in the table below.

Field	Size (bytes)	Notes
Object ID	8	Unique object identifier
Object Size	4	Object size in bytes
Block Size Log	1	Size of the block during this transfer


Initiating node
Distributor node
Updating node#1
Updating node#2
Init object transfer
Object Transfer Start(Object ID, Object Size, Block Size Log)
Object Transfer Status("Success", Object ID, Object Size, Block Size Log)
Object Transfer Start(Object ID, Object Size, Block Size Log)
Object Transfer Status("Success", Object ID, Object Size, Block Size Log)
http://msc-generator.sourceforge.net v6.1

3.3.3 Object Transfer Abort

The Object Transfer Abort is an acknowledged message used to abort the ongoing object transfer process. The response to the Object Transfer Abort message is an Object Transfer Status message. The structure of the Object Transfer Abort message is defined in the table below.

Field	Size (bytes)	Notes
Object ID	8	Unique object identifier

3.3.4 Object Transfer Status

The Object Transfer Status is an unacknowledged message used to report the state of the object transfer process. The structure of the Object Transfer Status message is defined in the table below.

Field	Size (bytes)	Notes
Status	1	Status of operation
Object ID	8	Unique object identifier
Object Size	4	Object size in bytes
Block Size Log	1	Size of the block during this transfer

Status possible values:

- 0x00 - ready, object transfer is not active
- 0x01 - busy, object transfer is active
- 0x02 - busy, with different transfer
- 0x03 – object is too big to be stored

3.3.5 Object Block Transfer Start

The Object Block Transfer Start is an acknowledged message used to start block transfer to the node.

The response to the Object Block Transfer Start message is an Object Block Transfer Status message.

The structure of the Object Block Transfer Start message is defined in the table below.

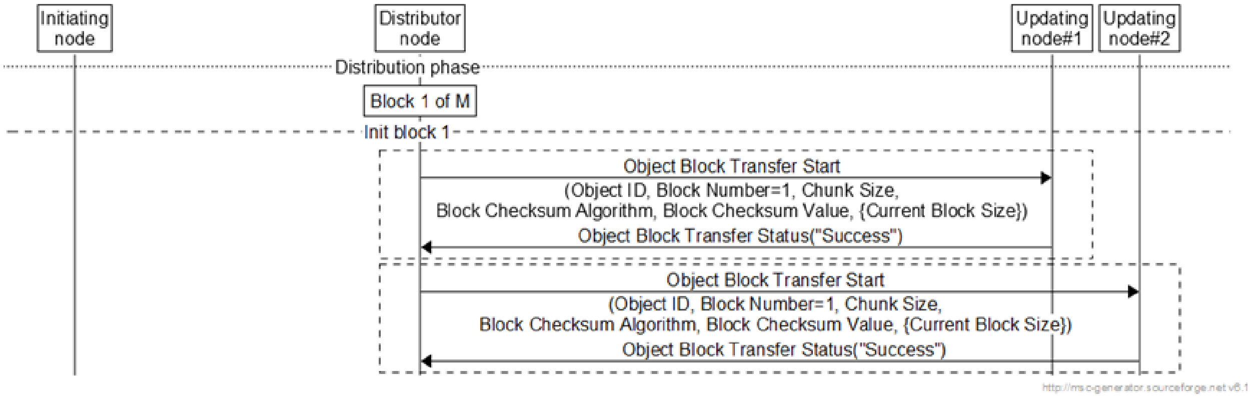
Field	Size (bytes)	Notes
Object ID	8	Unique object identifier
Block Number	2	Block number
Chunk Size	2	Chunk size bytes for this block
Block Checksum Algorithm	1	Checksum type
Block Checksum Value	variable	Checksum for image block
Current Block Size	2	Block size in bytes (optional)

Current Block Size is optional when equal to Block Size [Object Transfer Start] and mandatory when currently transferred block size is different than Block Size (could only happen in last block).

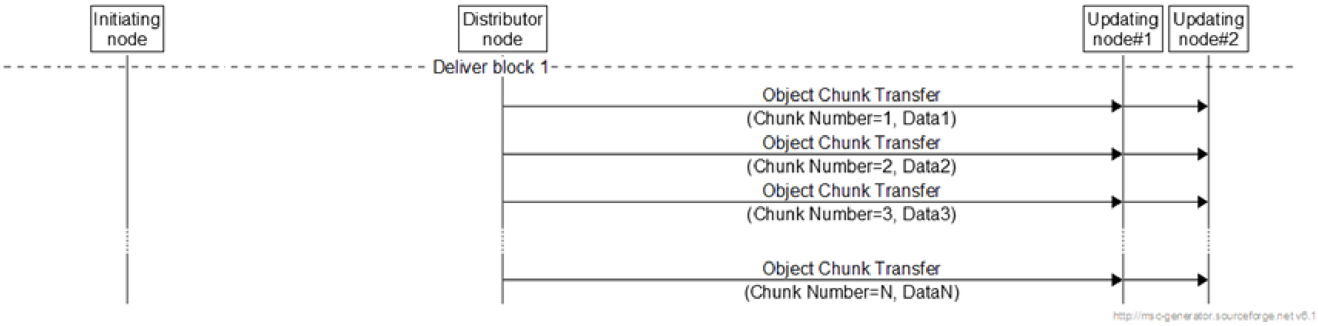
Note: Current Block Size is needed here to support last block padding.

The Block Checksum Algorithm values are defined in table below.

Value	Name	Block Checksum Value len	Notes
0x00	CRC32	4	Details TBD
0x01-0xFF	RFU	-	Reserved for Future Use


Initiating node
Distributor node
Updating node#1
Updating node#2
Distribution phase
Block 1 of M
Init block 1
Object Block Transfer Start
(Object ID, Block Number=1, Chunk Size,
Block Checksum Algorithm, Block Checksum Value, {Current Block Size})
Object Block Transfer Status("Success")
Object Block Transfer Start
(Object ID, Block Number=1, Chunk Size,
Block Checksum Algorithm, Block Checksum Value, {Current Block Size})
Object Block Transfer Status("Success")
http://msc-generator.sourceforge.net v6.1

The Block is actually composed of several chunks of the data


Initiating node
Distributor node
Updating node#1
Updating node#2
Deliver block 1
Object Chunk Transfer (Chunk Number=1, Data1)
Object Chunk Transfer (Chunk Number=2, Data2)
Object Chunk Transfer (Chunk Number=3, Data3)
Object Chunk Transfer (Chunk Number=N, DataN)
http://msc-generator.sourceforge.net v6.1

3.3.6 Object Block Transfer Status

The Object Block Transfer Status is an unacknowledged message used to report start of the block transfer to the node.

The structure of the Object Block Transfer Status message is defined in the table below.

Field	Size (bytes)	Notes
Status	1	Status of operation

Status field possible values:

- 0x00 - block transfer accepted
- 0x01 - block already transferred
- 0x02 - invalid block number, no previous block
- 0x03 - wrong current block size - bigger then Block Size Log [Object Transfer Start]
- 0x04 - wrong Chunk Size - bigger then Block Size divided by Max Chunks Number [Object Information Status]
- 0x05 – unknown checksum algorithm
- 0x0F - block transfer rejected

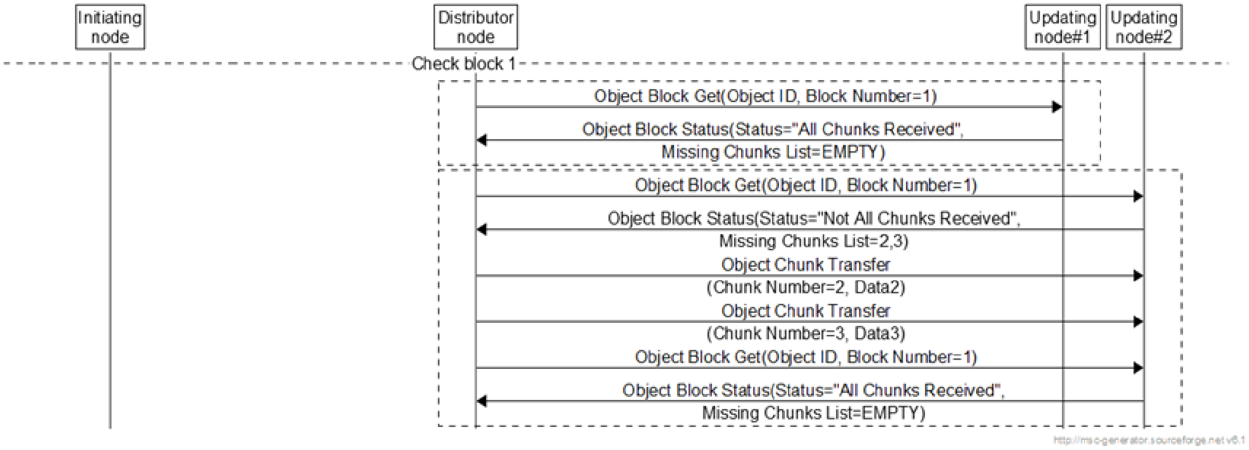
3.3.7 Object Chunk Transfer

The Object Chunk Transfer is an unacknowledged message used deliver chunk of a current block to the node or to the group of nodes.

The structure of the Object Chunk Transfer message is defined in the table below.

Field	Size (bytes)	Notes
Chunk Number	2	Chunk number
Firmware Image Data	1...256	Chunk data

The diagram below shows Block delivery check (unicast) and missing records delivery delivered via Object chunk transfer


Initiating node
Distributor node
Updating node#1
Updating node#2
Check block 1
Object Block Get(Object ID, Block Number=1)
Object Block Status(Status="All Chunks Received", Missing Chunks List=EMPTY)
Object Block Get(Object ID, Block Number=1)
Object Block Status(Status="Not All Chunks Received", Missing Chunks List=2,3)
Object Chunk Transfer (Chunk Number=2, Data2)
Object Chunk Transfer (Chunk Number=3, Data3)
Object Block Get(Object ID, Block Number=1)
Object Block Status(Status="All Chunks Received", Missing Chunks List=EMPTY)
http://msc-generator.sourceforge.net v6.1

3.3.8 Object Block Get

The Object Block Get is an acknowledged message used to get status of the current block transfer.

The response to the Object Block Get message is an Object Block Status message.

The structure of the Object Transfer Status message is defined in the table below.

Field	Size (bytes)	Notes
Object ID	8	Unique object identifier
Block Number	2	Block number

3.3.9 Object Block Status

The Object Block Status is an unacknowledged message used to report status of the current chunk transfer.

The structure of the Object Transfer Status message is defined in the table below.

Field	Size (bytes)	Notes

Status	1	
Missing Chunks List	2 * M	Missing chunks list (optional)

The values of the Status field are defined in table below.

Status Code	Status	Notes
0x00	All Chunks Received	All chunks received, checksum is valid, ready for the next block
0x01	Not All Chunks Received	Not all chunks received, checksum is not computed
0x02	Wrong Checksum	All chunks received, computed checksum value is not equal to expected value
0x03	Wrong Object ID	Requested Object ID not active
0x04	Wrong Block	Requested block not active
0x05-0xFF	RFU	Reserved for Future Use

The Missing Chunks List field is present only when Status field is equal to Not All Chunks Received

3.3.10 Object Information Get

The Object Information Get is an acknowledged message used to get object transfer capabilities of the node.

The response to the Object Information Get message is an Object Information Status message.

There are no parameters for this message.

3.3.11 Object Information Status

The Object Information Status is an unacknowledged message used to report object transfer capabilities of the node.

The structure of the Object Information Status message is defined in the table below.

Field	Size (bytes)	Notes

Min Block Size Log	1	Minimum block size: 2 ^ Max Block Size Log
Max Block Size Log	1	Maximum block size: 2 ^ Max Block Size Log
Max Chunks Number	2	Supported maximum number of chunks in block

3.4 Models definitions

Roles to models mapping.

Role	Models	Notes
Initiating node	Firmware Distribution Client Firmware Update Client	
Distributor node	Firmware Distribution Server Firmware Update Client Object Transfer Client	
Updating node	Firmware Update Server Object Transfer Server	

3.4.1 Firmware Update Server

Element	SIG Model ID	States	Messages	Rx	Tx
Main	0xFE00	FW Information	Firmware Information Get	M	
			Firmware Information Status		M
		FW Update Process	Firmware Update Get	M	
			Firmware Update Prepare	M	
			Firmware Update Start	M	
			Firmware Update Abort	M	
			Firmware Update Apply	M	
			Firmware Update Status		M

3.4.2 Firmware Distribution Server

Element	SIG Model ID	States	Messages	Rx	Tx
Main	0xFE02	FW Distribution Control	Firmware Distribution Get	M	
			Firmware Distribution Start	M	
			Firmware Distribution Stop	M	
			Firmware Distribution Status		M
		FW Distribution	Firmware Distribution Details Get	M	
			Firmware Distribution Details List		M

3.4.3 Object Transfer Server

Element	SIG Model ID	States	Messages	Rx	Tx
Main	0xFF00	Object	Object Transfer Get	M	
			Object Transfer Start	M	
			Object Transfer Abort	M	
			Object Transfer Status		M
		Block	Object Block Transfer Start	M	
			Object Block Transfer Status		M
			Object Block Get	M	
			Object Block Status		M
		Chunk	Object Chunk Transfer		M
		Capabilities	Object Information Get	M	
			Object Information Status		M

3.4.4 Firmware Update Client

Element	SIG Model ID	Procedure	Messages	Rx	Tx
Main	0xFE01	FW Information	Firmware Information Get		O
			Firmware Information Status	C.1	
		FW Update Process	Firmware Update Get		O
			Firmware Update Prepare		O
			Firmware Update Start		O

			Firmware Update Abort		O
			Firmware Update Apply		O
			Firmware Update Status	C.2	

C.1: If Firmware Information Get message is supported, the Firmware Information Status message shall also be supported; otherwise support for the Firmware Information Status message is optional.

C.2: If any of the messages: Firmware Update Get, Firmware Update Prepare, Firmware Update Start, Firmware Update Abort, Firmware Update Apply are supported, the Firmware Update Status message shall also be supported; otherwise support for the Firmware Update Status message is optional.

3.4.5 Firmware Distribution Client

Element	SIG Model ID	Procedure	Messages	Rx	Tx
Main	0xFE03	FW Distribution Control	Firmware Distribution Get		O
			Firmware Distribution Start		O
			Firmware Distribution Stop		O
			Firmware Distribution Status	C.1	
		FW Distribution	Firmware Distribution Details Get		O
			Firmware Distribution Details List	C.2	

C.1: If any of the messages: Firmware Distribution Get, Firmware Distribution Start, Firmware Distribution Stop are supported, the Firmware Distribution Status message shall also be supported; otherwise support for the Firmware Distribution Status message is optional.

C.2: If Firmware Distribution Details Get message is supported, the Firmware Distribution Details Status message shall also be supported; otherwise support for the Firmware Distribution Details Status message is optional.

3.4.6 Object Transfer Client

The following table illustrates the complete structure of elements, procedures, and messages used by the model. At least one message listed in the table shall be supported by this model.

Element	SIG Model ID	Procedure	Messages	Rx	Tx
Main	0xFF01	Object	Object Transfer Get		O
			Object Transfer Start		O
			Object Transfer Abort		O
			Object Transfer Status	C.1	
		Block	Object Block Transfer Start		O
			Object Block Transfer Status	C.2	
			Object Block Get		O

			Object Block Status	C.3	
		Chunk	Object Chunk Transfer	O	
		Capabilities	Object Information Get		O
			Object Information Status	C.4	

C.1: If any of the messages: Object Transfer Get, Object Transfer Start, Object Transfer Abort are supported, the Object Transfer Status message shall also be supported; otherwise support for the Object Transfer Status message is optional.

C.2: If Object Block Transfer Start message is supported, the Object Block Transfer Status message shall also be supported; otherwise support for the Object Block Transfer Status message is optional.

C.3: If Object Block Get message is supported, the Object Block Status message shall also be supported; otherwise support for the Object Block Status message is optional.

C.4: If Object Information Get message is supported, the Object Information Status message shall also be supported; otherwise support for the Object Information Status message is optional.

4 Security considerations

The security of the mesh firmware update is critical.

Besides the level of security provided by the BT Mesh Profile, the firmware can be sent encrypted by the Initiating node and decrypted by the Updating node. This kind of vendor-specific encryption is out-of-scope for this document.

5 References

[1] Bluetooth Mesh Profile Specification, Version 1.0

[2] Bluetooth SIG Assigned Numbers (http://www.bluetooth.com/specifications/assigned-numbers)

[3] Bluetooth SIG Company Identifiers (https://www.bluetooth.com/specifications/assigned-numbers/company-Identifiers)

[4] Bluetooth Mesh Model Specification, Version 1.0