**MASS SOFTWARE UPGRADE PROTOCOL SPECIFICATION**

Version 4.0

April 2012

|  |  |  |
| --- | --- | --- |
| Roles | Function | Name |
| Authors | Associate Lead – Software, AOCI | Chetan Ethapay |
| Engineer – Software, AOCI | Amulya Shekhar |
| Reviewers | Systems Architect, North Andover | David Kopp |
| Product Development Lead, North Andover | Ron Naismith |
| Technical Lead – Software, AOCI | Pawan Modi |
| Associate Lead – Software, AOCI | Shrikant Pawar |
| Technical Lead – Software, AOCI | Rajendra Alluri |
| Senior Engineer – Software, AOCI | Deepak Yadav |
| Approvers | Director, AOCI | Ramesh Phatak |

|  |  |  |  |
| --- | --- | --- | --- |
| Document Revision History | | | |
| Version | Date (yyyy/mm/dd) | Authors | Modifications Details |
| 1.0.1 | 2011/03/29 | Chetan Ethapay | Initial Creation |
| 1.0.2 | 2011/06/17 | Chetan Ethapay | Changed the format of the table headers in all the tables |
| 1.0.3 | 2011/08/18 | Chetan Ethapay | Added “data length” field in the upgrade message |
| 1.0.4 | 2011/11/25 | Chetan Ethapay | Added “MSU completed message” |
| 2.0 | 2011/12/12 | Chetan Ethapay | Added discovery messages “Who-Is” and “I-Am”, added “Status Message” from the clients |
| 2.1 | 2012/02/29 | Chetan Ethapay | Added abbreviations table, “Group Creation” and “Group Creation Acknowledge” messages, added “CCM completed” message and “Transfer Aborted” message |
| 3.0 | 2012/04/25 | Chetan Ethapay | Added SCM transfer completed message and description, added device parameter field in the I-Am message |
| 4.0 | 2014/08/19 | Amulya  Shekhar | Added authentication and made modifications to accommodate IpV6 support. |

**Abstract**

Mass Software Upgrade [MSU] is a reliable network protocol for carrying out simultaneous firmware/software upgrade process of a large number of systems. It uses a combination of multicast and unicast datagram protocols for data transfer. Reliability is achieved by selective re-transmission of the desired portion of the data upon request from the systems participating in the upgrade process.

1. **Abbreviations**

|  |  |
| --- | --- |
| **Abbreviation** | **Expansion** |
| MSU | Mass Software Upgrade |
| UDP | User Datagram Protocol |
| IGMP | Internet Group Management Protocol |
| FA | Firmware Archive |
| CN | Chunk Number |
| SN | Sequence Number |
| CCM | Chunk Complain Mode |
| SCM | Sequence Complain Mode |
| CM | Complain Mode |
| GM | Group Message |
| IM | Individual Message |

1. **Introduction & Overview**

The objectives of Mass Software Upgrade process are: 1) to simultaneously upgrade firmware/software on many systems, 2) to encourage indirect or implicit (via programs) use of remote computers, 3) to transfer data reliably and efficiently even on high traffic network. 4) Automatic recovery of upgrade process even in cases of network re-establishment.

This paper assumes knowledge of the User Datagram Protocol (UDP), Unicast and Multicast (IGMP) data transmission over the Ethernet.

1. **Terminology**
2. Mass Software Upgrade [MSU] – The process of carrying out simultaneous upgrade process of a large number of clients which require the same set of firmware package and/or configuration files.
3. Firmware Archive [FA] – Refers to a single binary file or a number of firmware files archived to form a single package. The files mentioned here may also refer to configuration files containing plain text. The terms FA and File are used interchangeably in this document.
4. Server System(s) – The system(s) which is the source of the firmware archive. Note: Although a number of server systems can be present on the same network, at a particular instant of MSU process, only one system is expected to be participating. The term server used in this document refers to the Server System.
5. Client System(s) – The system(s) in which the firmware needs to be updated. Note: The term client(s) used in this document refers to the Client System(s).
6. Chunk – Binary file or firmware archive broken down into a number of pieces of equal and predefined size. Note: The last chunk of the file may be of a different length when compared to the other chunks. Each chunk is numbered incrementally starting from one. This number is referred to as the chunk number [CN].
7. Sequence Number [SN] – Each chunk is further divided into a predefined number of units referred to with numbers starting from one.
8. Complain Mode – Selective re-transmission of a part of the firmware archive.
9. **Summary of Operation**

MSU process uses a combination of Multicast and Unicast datagram protocol for communication over the Ethernet for data transfer. It involves reliable data transfer using a mode of operation referred to as the complain mode for selective re-transmission of a part of the firmware archive.

shows an example setup for MSU process.

Server

Client 1

Client 5

Client 3

Client 2

Client 6

Ethernet Switch

Client 4

Client

7

Figure 1 - Example of MSU Process Setup

1. **Design Goals & Constraints**

MSU process has been designed to make use of the advantages of multicast datagram protocol to upgrade systems on a large scale simultaneously in order to save time and increase the network’s efficiency to a large extent.

1. **Reliability**

Reliable data delivery over the unreliable network is achieved by using the complain mode in the MSU process. The MSU process enters into complain mode after the transfer of each of the chunks hereby referred to as sequence complain mode [SCM] for re-transmission of a selected part of the chunk indicated by the SN.

By the end of the complete FA transfer, the MSU process enters into complain mode for re-transmission of the selected chunk indicated by the CN. This process is hereby referred to as the chunk complain mode [CCM].

1. **Precondition**

* + 1. Each of the client systems must have a unique MAC address.
    2. Each of the client systems must have unique IP address.

1. **Protocol Data Format**

All the data in the MSU protocol definitions are represented in the **Big-Endian** format.

1. **Device Discovery Messages**

The server sends the '**Who-Is'** multicast message on the default multicast IP address and port number to discover all the clients on the network. The clients reply with the ‘**I-Am’** unicast message.

* + - * 1. **Who-Is**

|  |  |  |  |
| --- | --- | --- | --- |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 |
| Opcode (4) | Subcode (4) | IsRange | Msg\_type | IP Version (1) | Protocol Version (3) | reserved (4) |
| Start Range/Device ID [0] | | | |
| Start Range/Device ID [1] | | | |
| Start Range/Device ID [2] | | | |
| Start Range/Device ID [3] | | | |
| Start Range/Device ID [4] | | | |
| Start Range/Device ID [5] | | | |
| Start Range/Device ID [6] | | | |
| Start Range/Device ID [7] | | | |
| Start Range/Device ID [8] | | | |
| Start Range/Device ID [9] | | | |
| End Range/Device ID[0] | | | |
| End Range/Device ID[1] | | | |
| End Range/Device ID[2] | | | |
| End Range/Device ID[3] | | | |
| End Range/Device ID[4] | | | |
| End Range/Device ID[5] | | | |
| End Range/Device ID[6] | | | |
| End Range/Device ID[7] | | | |
| End Range/Device ID[8] | | | |
| End Range/Device ID[9] | | | |

Table 1 - Who-Is Discovery Message Format (GM)

|  |  |
| --- | --- |
|  | Optional Field |
|  | Fixed Length Field |
|  | Variable Length Field |

* Opcode - MSU\_DEVICE\_DISCOVER (2)
* Subcode – WHO\_IS (1)

* IsRange - TRUE (1) or FALSE (0). If TRUE, the Start Range and End Range should be provided. If FALSE, Start Range is interpreted as the Device ID and the End Range is filled with zeros/null character. These fields must be 40 byte character arrays respectively, to extend IPv6 support.
* Msg\_type – This field contains different flag definitions.

|  |  |  |
| --- | --- | --- |
| ID | Flag\_Name(bit position) | Description |
| 1 | Client/Server(7) | Device is Client or Server. |
| 2 | Master (6) | Device is acting as the master |
| 3 | Authentication(5) | Authentication must be performed |
| 4 | Authentication\_level(4) | When set authentication must be performed the group level.  Else it must be performed for every device individually. |
| 5 | Reserved (3-0) |  |

Table 2 – Msg\_Type

**For the discovery request, only flags 1 and 2 are applicable. Rests are not used and hence their values must be set to zero.**

* IP header field

|  |  |  |
| --- | --- | --- |
| ID | Field Name(bit position) | Description |
| 1 | IP version(7) | This value set implies IPv4  Otherwise IPv6 support |
| 2 | Protocol Version(6-4) | This field should be set to 2. |
| 3 | Reserved field(3-0) |  |

Table 3 – IP header field

* Start Range – Starting IP address in the discovery process. This is applicable if IsRange is TRUE.
* Device ID – IP address of the device, applicable if IsRange is FALSE.
* End Range – End IP address in the discovery process. This is applicable if IsRange is TRUE.
  + - * 1. **I-Am**

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte-0** | **Byte-1** | **Byte-2** | **Byte-3** |
| Opcode (4) | Subcode (4) | GroupID | Msg\_type | IP Version (1) | Protocol Version (3) | reserved (4) |
| Device ID[0] | | | |
| Device ID[1] | | | |
| Device ID[2] | | | |
| Device ID[3] | | | |
| Device ID[4] | | | |
| Device ID[5] | | | |
| Device ID[6] | | | |
| Device ID[7] | | | |
| Device ID[8] | | | |
| Device ID[9] | | | |
| T/F | HW\_ID\_Len | T/F | Product\_ID\_Len | T/F | Product\_Name\_Len | T/F | Model\_Name\_Len |
| T/F | Vendor\_ID\_Len | T/F | FW\_SW\_Ver\_Len | T/F | Major\_Minor\_Revison\_Len | T/F | Device\_Loc\_Len |
| T/F | MSU\_Comm\_Param\_Len | T/F | Dev\_Param\_Len | Reserved | |
| Reserved for user specific field | | | |
| Variable Length Data | | | |
|
|
|

Table 2 - I-Am Discovery Message Format (IM)

* Opcode - MSU\_DEVICE\_DISCOVER (2)
* Subcode – I\_AM (2)
* GroupID – Group number to which this device belongs. The GroupID is supplied by the server during the Group Creation process. If the device is not yet assigned to any Group, by default, the GroupID is set to zero.
* Msg\_type – This field contains different flag definitions.

Client/Server(7)-This bit indicates if the device serves as a client or slave.

Master(6)-The device serves as a Master in the two level architecture.

Authentication(5)-When set, authentication must be performed.

Authentication\_level(4)- When set, the server must perform authentication for every individual device.

0 flag value implies that the devices can be authenticated at the group level.

Reserved(3-0)-

* IP Version – Refer to Table 3 description.
* Protocol Version (3) – Refer to description.
* DeviceID – Unique representation of the device such as the IP address for Ethernet devices. This field is again a 40 byte character array, to extend support for IPv6.
* T/F – Bit to suggest if the field exists or not.
* HW\_ID\_Len – Length of the Hardware Identification string.
* Product\_ID\_Len – Length of the Product Identification string.
* Product\_Name\_Len – Length of the Product Name string.
* Model\_Name\_Len – Length of the Model Name string.
* Vendor\_ID\_Len – Length of the Vendor Identification string.
* FW\_SW\_Ver\_Len – Length of the Firmware Software Version string.
* Major\_Minor\_Revison\_Len – Length of the Major and Minor Revision string.
* Device\_Loc\_Len – Length of the Device Location string.
* MSU\_Comm\_Param\_Len – Length of the MSU Communication Parameter string.
* Dev\_Param\_Len – Length of the Device Parameter field. This field is as shown in
* Table 3. The server, having collected the information from all the clients has to use the largest of all the values in the corresponding fields. This is in order to support even the slowest of all the clients.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Size in Bytes** | **Units** | **Description** |
| Sequence\_Delay | 2 | milliseconds | Delay between consecutive data packets |
| SCM\_Delay | 2 | milliseconds | Delay between consecutive SCMs |
| CCM\_Delay | 2 | milliseconds | Delay between consecutive CCMs |
| CCM\_Retry | 1 | number (0, 1,…255) | Number of CCM retries |
| SCM\_Retry | 1 | number (0, 1,…255) | Number of SCM retries |
| Timeout\_Val | 1 | seconds | Timeout value between any two consecutive response from the MSU server |

Table 3 - Device Parameter Field

* User specific Data. – User can define own identity to recognize the device. User can append the data at the end of the defined variable length data.
* Variable Length Data – String of variable length, the length is specified by the previous field. For example, File Name Length field specifies the length of this field.

1. **Group Creation [De-scoped]**

The server sends Group Creation unicast message to the individual clients with the GroupID information.

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte-0** | **Byte-1** | **Byte-2** | **Byte-3** |
| Opcode (4) | Subcode (4) | GroupID | Reserved | IP Version (1) | Protocol Version (3) | reserved (4) |

Table 4 – Group Creation (IM)

* Opcode – MSU\_DEVICE\_GROUP (3)
* Subcode – SET\_GROUP\_ID (1)

RESET\_GROUP\_ID (2)

* GroupID – Group number is specified by the server in order to group the devices. By default, the GroupID is zero. Using the RESET\_GROUP\_ID Subcode, the server can reset the GroupID to zero.
* IP Version – Refer to description.
* Protocol Version (3) – Refer to description.

1. **Group Creation Acknowledge**

Client Systems need to send ACK message to the server once they handle the group create message from server. If server does not receive the ACK message from the devices, it will resend the group create message. Number of retries is subjected to the implementation.

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte-0** | **Byte-1** | **Byte-2** | **Byte-3** |
| Opcode (4) | Subcode (4) | GroupID | Reserved | IP Version (1) | Protocol Version (3) | reserved (4) |

Table 5 – Group Creation Acknowledge (IM)

* Opcode – MSU\_DEVICE\_GROUP (3)
* Subcode – CREATE\_GROUP\_ACK (3)
* GroupID – device Group id (optional).
* IP Version – Refer to description.
* Protocol Version (3) – Refer to description.

1. **Connect(Authentication)**

Keeping compliance with the security standards, MSU performs authentication (connection permissibility) at three levels

* 1. Basic authentication – Verification of Username and Password alone.
  2. IP white list – The leech device accepts data packets from those servers whose address falls in the IP white list.

Packets from any other server must not be consumed.

The IP white list is fed into the device through a binary file at the time of commissioning.

This level of authentication also includes basic authentication.

* 1. MAC – This is the superset, and it further incorporates server MAC address authentication at the device level.

This information is again present with the device through the configuration file.

To add another layer to device safety, the correctness of the hardware and the image to be transferred is cross- verified at the device level.

If the authentication fails at any level, the device returns an error code defining the cause of the failure and disengages from the upgrade process.

The protocol also allows explicit disconnect. On receiving the disconnect message, the client must disengage from the Upgrade process.

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte-0** | **Byte-1** | **Byte-2** | **Byte-3** |
| opcode(4)|subcode(4) | Reserved | Msg\_type | IP version(1)| Protocol Version(3)|reserved(4) |
| Server Ip[0] | | | |
| Server Ip[1] | | | |
| Server Ip[2] | | | |
| Server Ip[3] | | | |
| Server Ip[4] | | | |
| Server Ip[5] | | | |
| Server Ip[6] | | | |
| Server Ip[7] | | | |
| Server Ip[8] | | | |
| Server Ip[9] | | | |
| Transaction ID(encrypt) | | | |
| MAC[0] | MAC[1] | MAC[2] | MAC[3] |
| MAC[4] | MAC[5] | T/F|Filename\_len | T/F|HardwareId\_len |
| T/F|Product\_Id\_len | T/F|ModelName\_len | T/F|Password\_len | T/F|Username\_len(encryption) |
| T/F|F/w-Ver\_Len | T/F|S/w-Ver\_Len | T/F|Vendor\_Id\_len | T/F|Product\_Name\_len |
| Variable Length Data | | | |

Table 6 **–** Connect Request Message Format (IM)

* Opcode - MSU\_DEVICE\_AUTHENTICATION (4)
* Subcode – REQ (1)
* Msg\_type – Refer to for description.
* IP Version – Refer to description.
* Protocol Version (3) – Refer to description.
* Server IP – IP address of the server.
* Transaction ID – Randomly generated number which uniquely represents one MSU cycle
* MAC ID – MAC address of the server.
* T/F – Bit to suggest if the field exists or not.
* Filename\_len – Length of the image filename.
* HW\_ID\_Len – Length of the Hardware Identification string.
* Product\_ID\_Len – Length of the Product Identification string.
* Model\_Name\_Len – Length of the Model Name string.
* Password\_len – Length of the password.
* Username\_len – Length of the username.
* FW\_SW\_Ver\_Len – Length of the Firmware Software Version string.
* Product\_Name\_Len – Length of the Product Name string.
* Variable Length Data – String of variable length, the length is specified by the previous field. For example, File Name Length field specifies the length of this field.

|  |  |  |  |
| --- | --- | --- | --- |
| Byte-0 | Byte-1 | Byte-2 | Byte-3 |
| opcode(4)|subcode(4) | Errorcode | Error Subcode | ` IP version(1)| Protocol Version(3)|reserved(4) |
| Append more data (only 4 bytes as of now) | | | |

Table 7 **–** Connect response Message Format (IM)

* Opcode - MSU\_DEVICE\_AUTHENTICATION
* Subcode – RES (2)
* Errorcode – 0 value implies the absence of error.

1 implies the presence of error.

* Error Subcode – The value of this field defines the reason of the failure.
* IP Version – IPV4 (0)

IPV6 (1) – reserved

* Protocol Version (3) – This field should be set to 2.

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte-0** | **Byte-1** | **Byte-2** | **Byte-3** |
| opcode(4)|subcode(4) | Reserved | Msg\_type | IP version(1)| Protocol Version(3)|reserved(4) |
| Server Ip[0] | | | |
| Server Ip[1] | | | |
| Server Ip[2] | | | |
| Server Ip[3] | | | |
| Server Ip[4] | | | |
| Server Ip[5] | | | |
| Server Ip[6] | | | |
| Server Ip[7] | | | |
| Server Ip[8] | | | |
| Server Ip[9] | | | |
| Transaction ID(encrypt) | | | |

Table 8 **–** Disconnect Message Format (IM)

* Opcode - MSU\_DEVICE\_AUTHENTICATION (4)
* Subcode – DISCONNECT (3)
* Msg\_type – Refer to for description.
* IP Version – Refer to for description.
* Protocol Version (3) – Refer to description.
* Server IP – IP address of the server.
* Transaction ID – Randomly generated number which uniquely represents one MSU cycle

1. **Update Process**

Server initiates the MSU process by multicasting notification message. The default multicast IP address, port number, the number of times this message is sent and the delay between each of the notification messages are implementation specific. The Notification message packet contents are shown in .

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte-0** | **Byte-1** | **Byte-2** | **Byte-3** |
| Opcode (4) | Subcode (4) | File Number | | IP Version (1) | Protocol Version (3) | reserved (4) |
| File Size (in bytes) | | | |
| Number of Chunks | | | |
| Sequence Number Limit | | Sequence Size Limit (in bytes) | |
| Multicast Address[0] | | | |
| Multicast Address[1] | | | |
| Multicast Address[2] | | | |
| Multicast Address[3] | | | |
| Multicast Address[4] | | | |
| Multicast Address[5] | | | |
| Multicast Address[6] | | | |
| Multicast Address[7] | | | |
| Multicast Address[8] | | | |
| Multicast Address[9] | | | |
| CM Multicast Address[0] | | | |
| CM Multicast Address[1] | | | |
| CM Multicast Address[2] | | | |
| CM Multicast Address[3] | | | |
| CM Multicast Address[4] | | | |
| CM Multicast Address[5] | | | |
| CM Multicast Address[6] | | | |
| CM Multicast Address[7] | | | |
| CM Multicast Address[8] | | | |
| CM Multicast Address[9] | | | |
| Port Number | | CM Port Number | |
| TransactionID[encryption] | | | |
| File CRC | | | |
| T/F | File\_Name\_Len | T/F | Dest\_Path\_Name\_Len | T/F | GroupId | Update Timeout |
| Variable Length Data | | | |
|
|
|

Table 8 – Notification Message Format (GM)

* Opcode – MSU\_DEVICE\_UPDATE (1)
* Subcode – UPGRADE (1)

DOWNGRADE (2)

FORCE\_UPGRADE (3)

* File Number – In case of upgrade process involving multiple files, this number represents the index of the file that is being transferred. This number starts from 1.
* File Size – Size of the file in bytes that is being transferred.
* Number of Chunks – Total number of chunks in the current file transfer.
* Sequence Number Limit – Maximum number of sequences in the first chunk up to the last but one chunk. Note that the last chunk may have a smaller number of sequences depending on the file size. This limit should not be more than 32.
* Sequence Size Limit – Maximum size in bytes of the data payload of a single packet sequence. Note that the last sequence of the last chunk may have a smaller size depending on the file size.
* Multicast Address – The client systems that are interested in participating in the MSU process need to join this multicast address.
* TransactionID – Randomly generated number which uniquely represents one MSU cycle.
* Port Number – The client systems that are interested in participating in the MSU process need to listen on this port to receive multicast messages.
* CM Multicast Address – The client systems which are interested in participating in the CM either in SCM or CCM need to join this multicast address.
* CM Port Number – The client systems which are interested in participating in the CM either in SCM or CCM need to listen on this port to receive CM multicast messages.
* File CRC – 32-bit CRC of the file.
* T/F – Bit to suggest if the field exists or not.

1 – Field exists.

0 – Field does not exist.

* File Name Len. – Length of the file name.
* Dest. Path Name Len. – Length of the destination path of the file under transfer.
* GroupID – The T/F bit suggests whether the GroupID is valid or not. If valid, then this can be any number from 1 to 128 (7-bit). GroupID 0 is the default group.
* Update Timeout - On Communication failure for a period greater than this timeout value, the Client MUST exit from the current process and become available for next update cycle without participating for the rest of the current update process. The status of the client will remain 'FAIL' until the beginning of the next update cycle. Refer to Appendix A for default UPDATE\_TIMEOUT value.
* Variable Length Data – String of variable length, the length is specified by the previous field. For example, File Name Length field specifies the length of this field.
  1. Using the GroupID field, the interested clients having the same GroupID need to join the multicast group as specified in the notification message packet. Please refer to for the message format.
  2. Server can force the clients to upgrade their firmware by sending the sub code FORCE\_UPGRADE. Upon receiving this message from the server, the clients compulsorily need to participate in the MSU process.
  3. The file [FA] is divided into smaller units called Chunks.
  4. The chunk is divided into smaller units called SN.

CHUNK 1

CN 1

CN 2

CN 3

.

.

.

CN n

SN 1

SN 2

SN 3

SN n

FILE

.

.

.

Figure 2 File – Chunk - Sequence Structure

* 1. Server prepares the chunks of the file which it needs to send over the network to the clients. Refer .
  2. The chunks are further divided into a number of parts depending on the maximum datagram payload size, optimization requirements and datagram fragmentation. These numbers are the sequence numbers [SN]. Refer . Please refer to Number of Chunks, Sequence Number Limit and Size of Sequence fields of .
  3. Server waits for a predefined JOIN\_MSU\_WAIT\_TIME time (refer Appendix A) for the clients to join the multicast group.
  4. Server starts sending the data packet from the first chunk to the end of the chunk.
  5. While sending each chunk of data, the server inserts the SN in the message. The data transfer message format is shown in the .

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte-0** | **Byte-1** | **Byte-2** | **Byte-3** |
| Opcode (4) | Subcode (4) | File Number | | IP Version (1) | Protocol Version (3) | reserved (4) |
| Chunk Number [CN] | | | |
| Sequence Number (Min 1 - Max 32) | Data Length | | File/Chunk Transfer States (2)| reserved (6) |
| Variable Length Data | | | |
|

Table 9 - Data Transfer Message Format (GM)

* Opcode – MSU\_DEVICE\_UPDATE (1)
* Subcode – DATA\_TRANSFER (4)
* File Number - Refer description.
* IP Version – Refer to description.
* Protocol Version (3) – Refer to description.
* CN – Refers to the chunk number that is being transferred. It starts from 1.
* SN – Refers to the sequence number that is sent in this message. It starts from 1, maximum being Sequence Number Limit. Please refer for Sequence Number Limit.
* Data Length – Length of the variable length data in bytes.
* File/Chunk Transfer States – FILE\_START [0]/END[1] – BIT 0

CHUNK\_START [0]/END [1] – BIT 1

* Variable Length Data – The actual FA data that needs to be transferred.
  1. The client builds a database of all the frames received with CN and SN information.
  2. After sending one complete chunk, the Server waits for a predefined SCM\_WAIT\_TIME time (refer Appendix A) for the SCM. In this mode, the clients request the server all the frames that were missed. These missed frames are indicated by the SN. The server re-transmits all the missed frames on the Multicast Address (optionally, if CM Multicast Address is supported, then, this address will be used). The Unicast SCM Message Format is shown in .

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte-0** | **Byte-1** | **Byte-2** | **Byte-3** |
| Opcode (4) | Subcode (4) | File Number | | IP Version (1) | Protocol Version (3) | reserved (4) |
| Chunk Number [CN] | | | |
| Number of Sequences | Reserved | | |
| Bit Map of Sequences | | | |

Table 10 - SCM Message Format (IM)

* Opcode – MSU\_DEVICE\_UPDATE (1)
* Subcode – SEQUENCE\_COMPLAIN\_MODE (5)
* File Number - Refers to the number of the file to which this SCM belongs.
* IP Version – Refer for description.
* Protocol Version- Refer for description.
* CN – Refers to the number of the chunk to which this SCM belongs.
* Number of Sequences – The total number of missed frames.
* Bit Map of Sequences – The missed sequence numbers are represented by bits. The 0th bit indicates SN=1 and so on.
  1. After the SCM\_WAIT\_TIME has elapsed, the server sends the SCM completed message for the current chunk which it has just transferred. The message format is shown in . If the SCM\_Retry\_Flag field is non-zero, then the client has another time interval of SCM\_WAIT\_TIME to form and send SCM message for those sequences it still has not received. This process of SCM retries continue until the server sends a SCM completed message with SCM\_Retry\_Flag field set to zero after which any further SCM messages from the client will be ignored by the server for the current chunk.

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte-0** | **Byte-1** | **Byte-2** | **Byte-3** |
| Opcode (4) | Subcode (4) | File Number | | IP Version (1) | Protocol Version (3) | reserved (4) |
| SCM\_Retry\_Flag | Reserved | | |

Table 7 - SCM Completed Message Format (GM)

* Opcode – MSU\_DEVICE\_UPDATE (1)
* Subcode – SCM\_TRANSFER\_COMPLETED (10)
* File Number – Refers to the index of the file that was transferred.
* IP Version – Refer to description.
* Protocol Version (3) – Refer to description.
  1. Server re-transmits all the SCM requested packets. The clients receive this re-transmitted data and recover the malformed chunk. If there are no packets requested by any of the clients, the server proceeds with the transfer of the next chunk.
  2. Server repeats the same procedure until all the chunks are sent, including the last chunk.
  3. After the SCM process for the last chunk, the server must send the Transfer Completed Message with the Subcode set to TRANSFER\_COMPLETED. Setting File/Chunk Transfer States field to FILE\_COMPLETED is OPTIONAL. The Multicast Transfer Completed Message Format is shown in .
  4. The number of times the server sends the Transfer Completed message is implementation specific.

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte-0** | **Byte-1** | **Byte-2** | **Byte-3** |
| Opcode (4) | Subcode (4) | File Number | | IP Version (1) | Protocol Version (3) | reserved (4) |

Table 8 - Transfer Completed Message Format (GM)

* Opcode – MSU\_DEVICE\_UPDATE (1)
* Subcode – TRANSFER\_COMPLETED (7)
* File Number – Refers to the index of the file that was transferred.
* IP Version – Refer to description.
* Protocol Version (3) – Refer to description.
  1. After the complete file transfer including the SCM for the last chunk and after the server sends the Transfer Completed Message, the MSU process goes into CCM. In this mode, the clients request for complete chunk re-transmission by sending the missed chunk number using the CCM Message. The server gathers the information from the clients within a window of time meant for CCM. The Unicast CCM Message Format is shown in .

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte-0** | **Byte-1** | **Byte-2** | **Byte-3** |
| Opcode (4) | Subcode (4) | File Number | | IP Version (1) | Protocol Version (3) | reserved (4) |
| Number of Chunks | | | |
| Chunk Number [CN] | | | |
| … | | | |
| Variable Length CNs | | | |

Table 9 - CCM Message Format (IM)

* Opcode - MSU\_DEVICE\_UPDATE (1)
* Subcode – CHUNK\_COMPLAIN\_MODE (6)
* File Number - Refers to the number of the file to which this CCM belongs.
* IP Version – Refer to description.
* Protocol Version (3) – Refer to description.
* Number of Chunks – This is the number of missed/malformed chunks.
* CN – Refers to the missed chunk number that the client wants the server to re-transmit. Each chunk number takes four bytes in the message format stacked one below the other as indicated by Variable Length CNs.
* Variable Length CNs – Refer to CN field.
  1. Server waits for a predefined CCM\_WAIT\_TIME time (refer Appendix A) allocated for the CCM. After this time, the server re-transmits the requested chunks one by one in ascending order until all the chunks are transmitted. The frame format and the procedure for sending the chunks remains same as before including the SCM states.
  2. After receiving Transfer completed message and the CCM, the clients verify the sanity of the received file using CRC. Optionally, the clients can verify CRC immediately after receiving the last chunk of the file if they do not need to participate in the CCM. This step is implementation specific.
  3. The server sends the CCM completed message after every iteration of the CCM. The number of iterations of the CCM is implementation specific. This number is predefined and known to the server and the clients. The format of this message is shown in . The CCM cycle is carried out more than once to increase the success rate of file transfer.

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte-0** | **Byte-1** | **Byte-2** | **Byte-3** |
| Opcode (4) | Subcode (4) | File Number | | IP Version (1) | Protocol Version (3) | reserved (4) |

Table 10 - CCM Completed Message Format (GM)

* Opcode – MSU\_DEVICE\_UPDATE (1)
* Subcode – CCM\_TRANSFER\_COMPLETED (9)
* File Number – Refers to the index of the file that was transferred.
* IP Version – Refer to description.
* Protocol Version (3) – Refer to description.
  1. At any point of time during the MSU process, the Server can send MSU abort message to all the participating clients to exit from the current MSU process by sending the TRANSFER\_ABORTED message as shown in .

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte-0** | **Byte-1** | **Byte-2** | **Byte-3** |
| Opcode (4) | Subcode (4) | File Number | | IP Version (1) | Protocol Version (3) | reserved (4) |

Table 11 - Transfer Aborted Message Format (GM)

* Opcode – MSU\_DEVICE\_UPDATE (1)
* Subcode – TRANSFER\_ABORTED (8)
* File Number – Refers to the index of the file that was transferred.
* IP Version – Refer to description.
* Protocol Version (3) – Refer to description.
  1. If the clients receive MSU abort, they have to undo all the MSU process which is subject to a clients’ local matter.
  2. Status Message – This message is sent (unicast) by the clients to the server with the Subcode set to CLIENT\_STATUS\_UPDATE\_RESPONSE. Two packets (number of packets is implementation specific) are sent in succession with a delay of STATUS\_MESSAGE\_GAP after the completion of CRC calculation.

The server can explicitly request for this message from the clients by sending (unicast) with the Subcode CLIENT\_STATUS\_UPDATE\_REQUEST. Refer for the message format.

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte-0** | **Byte-1** | **Byte-2** | **Byte-3** |
| Opcode (4) | Subcode (4) | Status | Error Code | IP Version (1) | Protocol Version (3) | reserved (4) |
| TransactionID | | | |
| DeviceID | | | |
| CCM Retry | Reserved | | |

Table 12 – Status Message Format (IM)

* Opcode - MSU\_DEVICE\_UPGRADE (1)
* Subcode – CLIENT\_STATUS\_UPDATE\_REQUEST (11)

CLIENT\_STATUS\_UPDATE\_RESPONSE (12)

* Status - PASS (0)

FAIL (1)

IN\_PROGRESS (3)

* Error Code - Failure code (0 if there is no failure)
* IP Version – Refer to description.
* Protocol Version (3) – Refer to description.
* TransactionID – The latest TransactionID stored by the client for the current MSU cycle.
* Device ID – Device Identification
* CCM Retry – Number of times the client had participated in CCM processes.

1. **Note**
   1. Upon the completion of each chunk, only those clients go into the SCM that has missed any packet in that chunk sent by the server.
   2. Upon the completion of all the chunks, only those clients go into the CCM that has missed any chunk sent by the server.
   3. Clients that have received complete chunk must ignore the SCM/CCM transactions.
   4. Server collects the packet numbers from the clients during the SCM and CCM and form a sequence of all the missed packets/chunks. Forming of sequence is done to avoid the overlapping of missed packets/chunks of multiple clients. This avoids the multiple re-transmission of the same packet/chunk to more than one client.
   5. During CCM, the server sends the requested malformed chunks to clients, one chunk at a time until all the requested chunks get transferred.
   6. The rules of SCM and CCM apply as it is for all the chunks irrespective of whether the chunk is in the normal MSU process or in the malformed chunk re-transmission process**.**
   7. The clients which are participating in the malformed chunk re-transmission processhave to analyze by the end of each re-transmitted chunk if they have any more malformed chunksstill left.
   8. Clients must discard duplicate packets received on the network.
   9. Clients must handle out of sequence reception of packets based on the SN.
   10. This protocol is not restricted to firmware. It may be used for any file on the client. That is, the configuration file, file system, special files etc.
2. **Limitations of the MSU Protocol**
   1. Supports only IPV4
   2. Server and Client devices must support file system

**Appendix A**

|  |  |  |
| --- | --- | --- |
| **NAME** | **DEFAULT VALUES\*** | **UNITS** |
| JOIN\_MSU\_WAIT\_TIME | 5 | seconds |
| SCM\_WAIT\_TIME | 3 | seconds |
| CCM\_WAIT\_TIME | 5 | Seconds |
| STATUS\_MESSAGE\_GAP | 10 | Milliseconds |
| UPDATE\_TIMEOUT | 10 | seconds |

\*The values indicated are configurable and are a local matter of the products that are participating in the MSU process.

Table 13 – Default Values

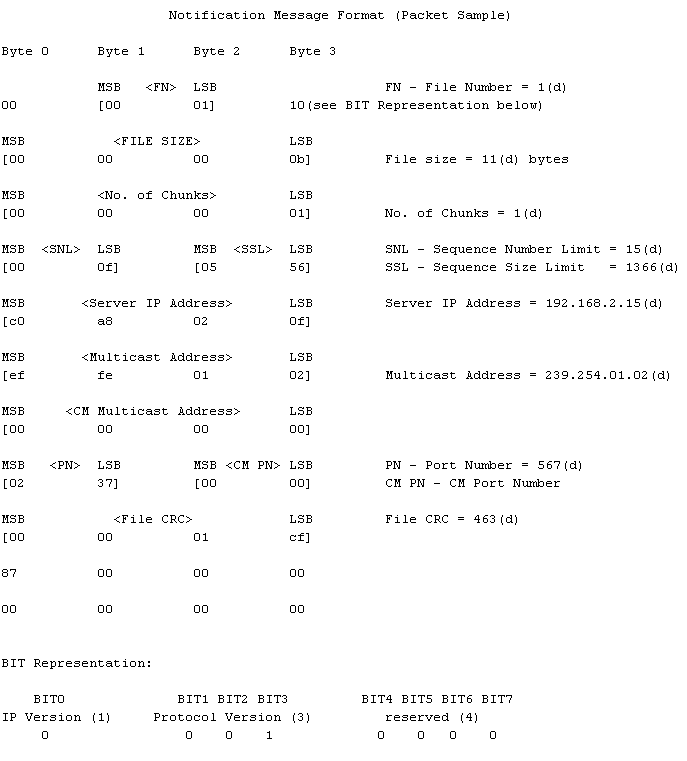
|  |  |  |
| --- | --- | --- |
| **No.** | **OpCode** | **SubCode** |
| 1 | MSU\_DEVICE\_CODE\_UNDEF (0) | NA |
| 2 | MSU\_DEVICE\_UPGRADE (1) | UPGRADE (1) |
| DOWNGRADE (2) |
| FORCE\_UPGRADE (3) |
| DATA\_TRANSFER (4) |
| SEQUENCE\_COMPLAIN\_MODE (5) |
| CHUNK\_COMPLAIN\_MODE (6) |
| TRANSFER\_COMPLETED (7) |
| TRANSFER\_ABORTED (8) |
| CCM\_TRANSFER\_COMPLETED (9) |
| SCM\_TRANSFER\_COMPLETED (10) |
| CLIENT\_STATUS\_UPDATE\_REQUEST (11) |
| CLIENT\_STATUS\_UPDATE\_RESPONSE (12) |
| 3 | MSU\_DEVICE\_DISCOVER (2) | WHO-IS (1) |
| I-AM (2) |
| 4 | MSU\_DEVICE\_GROUP (3) | SET\_GROUP\_ID (1) |
| RESET\_GROUP\_ID (2) |
| CREATE\_GROUP\_ACK (3) |
| 5 | MSU\_USER\_CMD (15) | START (1) |
| STOP (2) |
| INIT (3) |
| 6 | MSU\_DEVICE\_AUTHENTICATION(4) | REQ (1) |
| RES(2) |

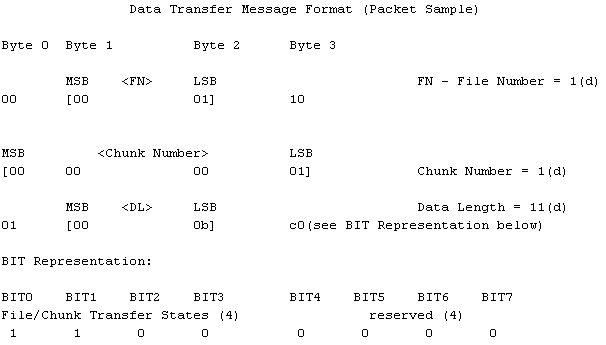
Table 14 – Command List

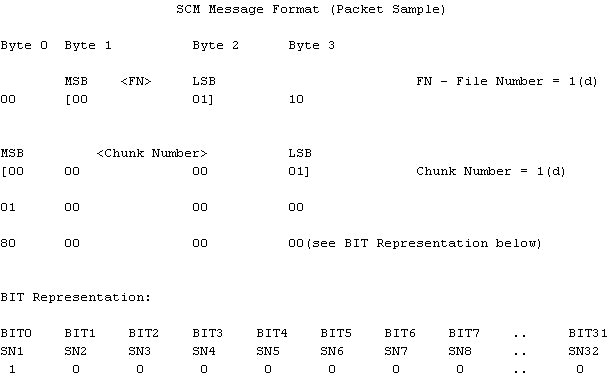
|  |  |
| --- | --- |
| **Name** | **Value** |
| **MSU\_CLIENT\_STATUS** | **1** |
| **MSU\_MASTER\_STATUS** | **0** |
| **MSU\_AUTHENTICATION\_TAG** | **1** |
| **MSU\_AUTHETICATION\_LEVEL\_TAG** | **1** |
| **MSU\_MAX\_IP\_FIELDS** | **1** |
| **MSU\_MAX\_FILES\_SUPPORTED** | **3** |
| **MSU\_DEFAULT\_SERVER\_IP** | **3232236043** |
| **MSU\_DEFAULT\_USERNAME** | **“admin”** |
| **MSU\_DEFAULT\_PASSWORD** | **“admin”** |
| **MSU\_DEFAULT\_FILENAME** | **“App2.out”** |
| **MSU\_DEFAULT\_FILENAME1** | **“text1.txt”** |
| **MSU\_DEFAULT\_FILENAME2** | **“text2.txt”** |

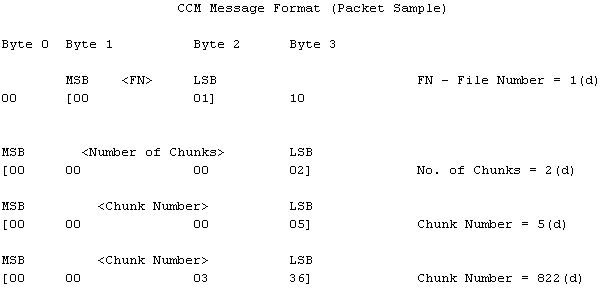
**Appendix B**

**Appendix B represents samples of Notification Message Format as per ; Data Transfer Message Format as per ; SCM Message Format as per ; CCM Message Format as per . These formats are illustrated as seen on the wire.**





****



**Appendix C**



Figure 3 - Client Flow Chart

**Init** – Initialization operation

**Config** – Configuration operation

**Listening** – Mode of listening to Notification message

**Update Mode** – Main mode of MSU operation

**Explicit Commands** – Who-Is, I-am, Status commands, etc.

**CCM** - Chunk Complain Mode of operation

**SCM** - Sequence Complain Mode of operation

**Abort** - Abort operation handling

**Finalization** - CRC calculation, status determination