



Software Design Specification

Z-Wave Management Command Class Specification

Document No.:	SDS13782
Version:	
Description:	The document describes the Z-Wave Command Classes and associated Commands used by Z-Wave enabled products at the Operation and Management level.
Written By:	JFR;NOBRIOT;BBR
Date:	
Reviewed By:	BBR;JFR;NOBRIOT
Restrictions:	Public

Approved by:

This document is the property of Silicon Labs. The data contained herein, in whole or in part, may not be duplicated, used or disclosed outside the recipient for any purpose. This restriction does not limit the recipient's right to use information contained in the data if it is obtained from another source without restriction.



REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20170102	NOBRIOT	ALL 4.6 to 4.8 4.25 & 4.26 4.30 4.33& 4.34 4.37	Transferred content from [2] and [3] Integrated approved content from Open Review 2016C and 2016D <ul style="list-style-type: none"> Clarified the AGI Command Class Clarified the Multi Channel Association Command Class Clarified Version Command Class, version 1 regarding how to treat supported and controlled Command Classes. Clarified Wake Up Command Class regarding how to handle unsupported interval values and unsolicited communication. Added minor adjustments to the Z-Wave Plus Info Command Class
2	20170403	NOBRIOT	4.3, 4.4, 4.19, 4.25 & 4.26 4.30 & 4.31 4.13 4.33 & 4.34 4.9 4.32	Integrated approved content from Open Review 2017A: <ul style="list-style-type: none"> Clarified Associations, IP Association and Multi Channel Association: association should be made only if the 2 nodes are bootstrapped on the same Security level. Clarified Version Command Class and the use of "Command Class version" and "Hardware version" fields. Added minor adjustments to the Firmware Update Meta Data, version 3 Refactored the Wake-Up Command Class, version 1 and 2 Refactored the Battery Command Class, version 1. Added Version Command Class, version 3
3	20170702	NOBRIOT	4.3, 4.4, 4.19, 4.25 & 4.26 4.27 4.6 4.35 4.19 & 4.35	Integrated approved content from Open Review 2017B: <ul style="list-style-type: none"> Clarified Association, IP Association and Multi Channel Association regarding the expected behavior of a controlling node that associates two other nodes Refactored the Node Naming and Location Command Class, version 1 Added minor adjustments to AGI Command Class, version 1 Added minor adjustments to the Z/IP Naming and Location Command Class, version 1 Moved IP Association and Z/IP Naming and Location from [17]
4	20171002	NOBRIOT	4.17 4.10 4.2 4.3, 4.4, 4.25 & 4.26	Integrated approved content from Open Review 2017C: <ul style="list-style-type: none"> Obsoleted Hail Command Class Clarified the allowed Multi Channel encapsulation for Device Reset Locally Notification Changed Application Status Command Class to be always advertised in the NIF (i.e. supported non-securely) Adjusted requirement level to Z-Wave Plus for Association Command Class and Multi Channel Association Command Class.
5	20180110	NOBRIOT	4.30, 4.22 & 4.23	Integrated approved content from Open Review 2017D: <ul style="list-style-type: none"> Version Command Class and Manufacturer Specific Command Class security considerations are overruled by S2 requirements
6	20180305	BBR	All	Added Silicon Labs template
7	20180405	NOBRIOT	4.18, 4.19 & 4.20 4.22 & 4.23 4.1 4.37	Integrated approved content from Open Review 2018A: <ul style="list-style-type: none"> Added Indicator Command Class, version 1-3 Mandated S0 bootstrapped node to answer to non-secure Manufacturer Specific Get Commands. Obsoleted the Application Capability Command Class Added a new value for the Z-Wave Plus Info Report

Table of Contents

1	ABBREVIATIONS.....	1
2	INTRODUCTION.....	2
2.1	Precedence of definitions	2
2.2	Terms used in this document	2
3	COMMAND CLASS OVERVIEW.....	3
4	COMMAND CLASS DEFINITIONS	4
4.1	Application Capability Command Class, version 1 [OBSOLETE]	5
4.1.1	Not Supported Command Class Command.....	5
4.2	Application Status Command Class, version 1	7
4.2.1	Compatibility Considerations.....	7
4.2.1.1	Node Information Frame (NIF)	7
4.2.2	Application Busy Command	7
4.2.3	Application Rejected Request Command	8
4.3	Association Command Class, version 1	9
4.3.1	Compatibility considerations	9
4.3.2	Z-Wave Plus considerations	9
4.3.3	Security considerations	9
4.3.4	Association Set Command.....	10
4.3.5	Association Get Command	11
4.3.6	Association Report Command	12
4.3.7	Association Remove Command.....	13
4.3.8	Association Supported Groupings Get Command	14
4.3.9	Association Supported Groupings Report Command.....	14
4.4	Association Command Class, version 2.....	15
4.4.1	Compatibility considerations	15
4.4.2	Z-Wave Plus considerations	15
4.4.3	Security considerations	15
4.4.4	Association Remove Command.....	16
4.4.5	Association Specific Group Get Command.....	17
4.4.6	Association Specific Group Report Command.....	18
4.5	Association Command Configuration Command Class, version 1	19
4.5.1	Command Records Supported Get Command	20
4.5.2	Command Records Supported Report Command	20
4.5.3	Command Configuration Set Command	22
4.5.4	Command Configuration Get Command.....	23
4.5.5	Command Configuration Report Command.....	24
4.6	Association Group Information (AGI) command class, version 1	26
4.6.1	Compatibility considerations	26
4.6.1.1	Multi Channel considerations	26
4.6.2	Association Principles	26
4.6.3	The Association Group Information	26
4.6.3.1	Group identifier	27
4.6.3.2	Profile	27
4.6.3.3	Command Class and Command	31
4.6.3.4	Association group name	32
4.6.4	Association Group Name Get	32
4.6.5	Association Group Name Report	33
4.6.6	Association Group Info Get	33
4.6.7	Association Group Info Report	35

4.6.8	Association Group Command List Get.....	37
4.6.9	Association Group Command List Report.....	38
4.7	Association Group Information (AGI) command class, version 2	39
4.7.1	Compatibility considerations	39
4.7.2	The Association Group Information	39
4.7.2.1	Profile	39
4.8	Association Group Information (AGI) command class, version 3	42
4.8.1	Compatibility considerations	42
4.8.2	The Association Group Information	42
4.8.2.1	Profile	42
4.9	Battery Command Class, version 1	45
4.9.1	Battery Level Get Command.....	45
4.9.2	Battery Level Report Command.....	45
4.10	Device Reset Locally Command Class, version 1	46
4.10.1	Device Reset Locally Notification Command	46
4.11	Firmware Update Meta Data Command Class, version 1 [DEPRECATED].....	47
4.11.1	Firmware Meta Data Get Command	47
4.11.2	Firmware Meta Data Report Command	48
4.11.3	Firmware Update Meta Data Request Get Command	49
4.11.4	Firmware Update Meta Data Request Report Command	50
4.11.5	Firmware Update Meta Data Get Command	51
4.11.6	Firmware Update Meta Data Report Command	52
4.11.7	Firmware Update Meta Data Status Report Command	53
4.11.8	Examples.....	54
4.12	Firmware Update Meta Data Command Class, version 2.....	55
4.12.1	Compatibility considerations	55
4.12.2	Interoperability considerations	55
4.12.2.1	Interoperability with v1 devices	55
4.12.2.2	Checksum calculation	56
4.12.3	Firmware Update Meta Data Report Command	56
4.13	Firmware Update Meta Data Command Class, version 3.....	58
4.13.1	Compatibility considerations	58
4.13.1.1	New fields and values	58
4.13.2	Interoperability considerations	58
4.13.2.1	Interoperability with v1 devices	59
4.13.2.2	Checksum calculation	59
4.13.3	Firmware Meta Data Report Command	60
4.13.4	Firmware Update Meta Data Request Get Command	61
4.13.5	Firmware Update Meta Data Request Report Command	63
4.13.6	Firmware Update Meta Data Get Command	64
4.13.7	Firmware Update Meta Data Report Command	66
4.13.8	Firmware Update Meta Data Status Report Command	67
4.14	Firmware Update Meta Data Command Class, version 4.....	69
4.14.1	Compatibility considerations	69
4.14.1.1	New commands, fields and values.....	69
4.14.2	Interoperability considerations	69
4.14.2.1	Interoperability with v1 devices	69
4.14.2.2	Checksum calculation	70
4.14.3	Firmware Update Meta Data Request Get Command	71
4.14.4	Firmware Update Meta Data Status Report Command	72
4.14.5	Firmware Update Activation Set Command.....	73
4.14.6	Firmware Update Activation Status Report.....	74
4.15	Firmware Update Meta Data Command Class, version 5.....	75
4.15.1	Compatibility considerations	75
4.15.1.1	New commands, fields and values.....	75
4.15.2	Interoperability Considerations.....	76
4.15.2.1	Interoperability with v1 devices	76

4.15.2.2	Checksum calculation	76
4.15.3	Firmware Meta Data Report Command	77
4.15.4	Firmware Update Meta Data Request Get Command	78
4.15.5	Firmware Update Meta Data Request Report Command	79
4.15.6	Firmware Update Meta Data Status Report Command	80
4.15.7	Firmware Update Activation Set Command	82
4.15.8	Firmware Update Activation Status Report Command	83
4.15.9	Firmware Update Meta Data Prepare Get Command	85
4.15.10	Firmware Update Meta Data Prepare Report Command	86
4.15.11	Examples	87
4.15.11.1	Identifying firmware revisions	87
4.15.11.2	Firmware download	88
4.16	Grouping Name Command Class, version 1 [DEPRECATED]	89
4.16.1	Grouping Name Set Command	89
4.16.2	Grouping Name Get Command	90
4.16.3	Group Name Report Command	91
4.17	Hail Command Class, version 1 [OBSOLETE]	92
4.17.1	Hail Command	92
4.18	Indicator Command Class, version 1	93
4.18.1	Interoperability considerations	93
4.18.2	Indicator Set Command	93
4.18.3	Indicator Get Command	93
4.18.4	Indicator Report Command	94
4.19	Indicator Command Class, version 2	95
4.19.1	Compatibility Considerations	95
4.19.2	Interoperability considerations	95
4.19.3	Indicator Set Command	96
4.19.4	Indicator Get Command	97
4.19.5	Indicator Report Command	98
4.19.6	Indicator Supported Get Command	99
4.19.7	Indicator Supported Report Command	100
4.20	Indicator Command Class, version 3	102
4.20.1	Compatibility Considerations	102
4.21	IP Association Command Class, version 1	103
4.21.1	Compatibility considerations	103
4.21.2	Terminology	103
4.21.3	Z-Wave Multi Channel compatibility considerations	104
4.21.4	Advertising the IP Association Command Class	105
4.21.5	IP Association Set Command	106
4.21.6	IP Association Get Command	107
4.21.7	IP Association Report Command	108
4.21.8	IP Association Remove Command	109
4.22	Manufacturer Specific Command Class, version 1	111
4.22.1	Security Considerations	111
4.22.2	Manufacturer Specific Get Command	111
4.22.3	Manufacturer Specific Report Command	112
4.23	Manufacturer Specific Command Class, version 2	113
4.23.1	Security Considerations	113
4.23.2	Device Specific Get Command	113
4.23.3	Device Specific Report Command	114
4.24	Multi Channel Association Command Class, version 1 [OBSOLETE]	115
4.25	Multi Channel Association Command Class, version 2	116
4.25.1	Compatibility considerations	116
4.25.2	Z-Wave Plus considerations	117
4.25.3	Security considerations	117
4.25.4	Multi Channel Association Set Command	118
4.25.5	Multi Channel Association Get Command	120

4.25.6	Multi Channel Association Report Command	121
4.25.7	Multi Channel Association Remove Command	122
4.25.7.1	Examples	124
4.25.8	Multi Channel Association Supported Groupings Get Command	129
4.25.9	Multi Channel Association Supported Groupings Report Command	130
4.26	Multi Channel Association Command Class, version 3	131
4.26.1	Compatibility considerations	131
4.26.2	Z-Wave Plus considerations	131
4.26.3	Security considerations	132
4.26.4	Multi Channel Association Set Command	133
4.27	Node Naming and Location Command Class, version 1	135
4.27.1	Node Name Set Command	135
4.27.2	Node Name Get Command	136
4.27.3	Node Name Report Command	136
4.27.4	Node Location Set Command	137
4.27.5	Node Location Get Command	138
4.27.6	Node Location Report Command	138
4.28	Remote Association Activation Command Class, version 1 [OBSOLETE]	139
4.28.1	Remote Association Activate Command	140
4.29	Remote Association Configuration Command Class, version 1 [OBSOLETE]	141
4.29.1	Remote Association Configuration Set Command	142
4.29.2	Remote Association Configuration Get Command	143
4.29.3	Remote Association Configuration Report Command	144
4.30	Version Command Class, version 1	145
4.30.1	Compatibility Considerations	145
4.30.2	Security Considerations	145
4.30.3	Version Get Command	146
4.30.4	Version Report Command	146
4.30.5	Version Command Class Get Command	148
4.30.6	Version Command Class Report Command	148
4.31	Version Command Class, version 2	149
4.31.1	Version Report Command	149
4.32	Version Command Class, version 3	153
4.32.1	Compatibility considerations	153
4.32.2	Version Capabilities Get Command	153
4.32.3	Version Capabilities Report Command	154
4.32.4	Version Z-Wave Software Get Command	154
4.32.5	Version Z-Wave Software Report Command	155
4.32.5.1	Version fields	157
4.33	Wake Up Command Class, version 1	158
4.33.1	Terminology	158
4.33.2	Interoperability considerations	158
4.33.3	Wake Up Interval Set Command	159
4.33.4	Wake Up Interval Get Command	159
4.33.5	Wake Up Interval Report Command	160
4.33.6	Wake Up Notification Command	160
4.33.7	Wake Up No More Information Command	161
4.34	Wake Up Command Class, version 2	162
4.34.1	Compatibility considerations	162
4.34.2	Wake Up Interval Set Command	162
4.34.3	Wake Up Interval Capabilities Get Command	163
4.34.4	Wake Up Interval Capabilities Report Command	164
4.35	Z/IP Naming and Location Command Class, version 1	167
4.35.1	Compatibility Considerations	167
4.35.2	Z/IP Name Set Command	167
4.35.3	Z/IP Name Get Command	168
4.35.4	Z/IP Name Report Command	168

4.35.5	Z/IP Location Set Command	169
4.35.6	Z/IP Location Get Command.....	169
4.35.7	Z/IP Location Report Command.....	170
4.36	Z-Wave Plus Info Command Class, version 1 [OBSOLETE]	171
4.37	Z-Wave Plus Info Command Class, version 2	172
4.37.1	Compatibility Considerations.....	172
4.37.1.1	Node Information Frame (NIF)	172
4.37.2	Multi Channel considerations	172
4.37.3	Z-Wave Plus Info Get Command	172
4.37.4	Z-Wave Plus Info Report Command	173
REFERENCES		179

Table of Figures

Figure 1, Light control transmitter (example).....	26
Figure 2, Lamp module (example).....	26
Figure 3, Requesting Manufacturer ID and Firmware ID of a node	54
Figure 4, Transferring a firmware image to a device	54
Figure 5, Identifying compatible firmware images	88
Figure 6, Firmware download flow diagram	88
Figure 7, IP Association example	104
Figure 8, Remote Association Activation Command Class	139
Figure 9, Remote Association Configuration Command Class	141
Figure 10, Version Report::Firmware numbering	150
Figure 11, Version Report::Firmware numbering (2)	150
Figure 12, Wake Up sequence	158

Table of Tables

Table 1, Association Remove, V1::Parameter interpretation	13
Table 2, Association Remove, V2::Parameter interpretation	16
Table 3, Layout of one line of the Association Group Information table	27
Table 4, Example: AGI tables for two-button light control transmitter	28
Table 5, Example: AGI tables for two-function sensor	30
Table 6, Example: AGI table for one-function alarm device	31
Table 7, Example: AGI tables for three-function meter	40
Table 8, AGI Profiles.....	43
Table 9, Firmware Update Meta Data Request Report::Status encoding	79
Table 10, Firmware Update Meta Data Status Report::Status encoding	81
Table 11, Firmware Update Activation Status Report::Firmware Update Status encoding	84
Table 12, Firmware Update Meta Data Prepare Report::Status encoding.....	86
Table 13, Labelling different Firmware revisions	87
Table 14, IP association mapping to Z Wave association	105
Table 15, IP Association Remove, V1 :: Parameter interpretation	110
Table 17, V2 Associations and the transmissions they may trigger	117
Table 18, Multi Channel Association Remove, V2 :: Parameter interpretation	123
Table 19, V3 Associations and the transmissions they may trigger	131
Table 20, Node Name Set::Char. Presentation encoding	136
Table 21, Version Command Class support.....	145
Table 22, Z-Wave Library Type	147
Table 23, Wake Up Interval Capabilities Report::Maximum Wake Up Interval Seconds encoding	165
Table 24, Wake Up Interval Capabilities Report::Wake Up Interval Step Seconds encoding	166

Table 25, Node Type identifiers.....	173
Table 26, The standard ASCII Table	175
Table 27, OEM Extended ASCII Table.....	176
Table 28, Players Table.....	176

1 ABBREVIATIONS

Abbreviation	Explanation
AGI	Association Group Information
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange. An ASCII code is the numerical representation of a character.
CRC	Cyclic Redundancy Check
DHCP	Dynamic Host Configuration Protocol.
DNS	Dynamic Host Service
ID	Identifier
IP	Internet Protocol
IPV4	Internet Protocol version 4
IPV6	Internet Protocol version 6
LF	Linefeed character.
LSB	Less Significant Bit
MSB	Most Significant Bit
NIF	Node Information Frame
PIR	Pyroelectric Infrared
SUC	Static Update Controller
Unicode	Unicode is a standard for encoding of characters. For more information, visit http://www.unicode.org/
Z/IP	Z-Wave for IP

2 INTRODUCTION

Commands classes are divided in four categories:

- Application Command Classes [16]
- Management Command Classes
- Transport-Encapsulation Command Classes [15]
- Network-Protocol Command Classes [17]

The list of defined Command Classes with their associated category is available in [14].

This document describes the Command Classes designed for node management specific purposes. Management command classes are used for service and capabilities discovery, as well as battery management or node associations.

2.1 Precedence of definitions

Device Class, Device Type and Command Class Specifications approved as final version during the Device Class, Device Type and Command Class Open Review process have precedence over this document until integrated into this document.

2.2 Terms used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document MUST be interpreted as described in IETF RFC 2119 [6].

Statements containing the IETF RFC 2119 [6] key words are at times marked with unique requirement numbers in the margin. The requirements numbers have the following syntax: CC:xxxx.xx.xx.xxx with each x being an hexadecimal digit.

This document defines functionality as deprecated or obsoleted.

The term "obsolete" means that the functionality MUST NOT be supported in new implementations applying for certification.

A controller SHOULD provide controlling capabilities of the actual functionality for backwards compatibility with legacy devices.

The term "deprecated" also indicates an obsolete definition, but it permits new implementations applying for certification.

Thus, the term "deprecated" means that the functionality SHOULD NOT be supported in new implementations applying for certification. Often, another substitute functionality is REQUIRED if the deprecated functionality is implemented.

A controller SHOULD provide controlling capabilities of the actual functionality for backwards compatibility with legacy devices.

3 COMMAND CLASS OVERVIEW

General command class overview and rules are described in the Z-Wave Application Command Class Specification [16] and are valid for the Command Classes presented in this document.

No additional considerations apply for the Management Command Classes.

4 COMMAND CLASS DEFINITIONS

The following subchapters contain definitions of Management Command Classes.

4.1 Application Capability Command Class, version 1 [OBSOLETE]

THIS COMMAND CLASS HAS BEEN OBSOLETE

New implementations **MUST NOT** support this Command Class.

New implementations **MUST** use the Supervision Command Class to report support or no support of a given Command Class. Refer to [15].

The Application Capability Command Class comprises commands for handling issues related to dynamic support for command classes.

Examples include nodes, which only support certain command classes when included securely, and controllers that only support command classes for primary controllers when they are actually operating as primary controllers.

The intention of the Application Capability Command Class is to provide tools for handling exceptions. Controlling nodes **SHOULD** maintain a local representation of supported command classes by destination nodes rather than relying on destination nodes returning Application Capability messages.

Destination nodes capable of returning Application Capability messages **MUST** list Application Capability Command Class as supported only.

4.1.1 Not Supported Command Class Command

The Not Supported Command Class Command is used by a node to indicate to a requesting node that the requested command class is not supported. The offending command and command class are encapsulated.

A node **SHOULD** use this command to indicate to other nodes that a command is not supported.

A node receiving this command **SHOULD** use the information to clean up internal states by requesting up-to-date information on command class support from the sending node using Node Information frame. This includes adjusting user interface details accordingly.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_APPLICATION_CAPABILITY							
Command = COMMAND_COMMAND_CLASS_NOT_SUPPORTED							
Dyna- mic	Reserved						
Offending Command Class (0x20 – 0xEE)							
Offending Command							

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_APPLICATION_CAPABILITY							
Command = COMMAND_COMMAND_CLASS_NOT_SUPPORTED							
Dyna- mic	Reserved						
Offending Command Class MSB (0xF1 – 0xFF)							
Offending Command Class LSB (0x00 – 0xFF)							
Offending Command							

Payload data following the offending command is omitted.

Dynamic (1 bit)

If the “Dynamic” flag is set (‘1’), the sending node has been designed to support this command class but the current operational conditions prevent the node from supporting this command class. One example is the AddNode command used for network management. This command is only supported when a controller is operating as primary controller.

Value	Meaning
‘0’	Command not supported (Permanently)
‘1’	Dynamic support. (Currently no support)

Reserved

This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Offending Command Class (1 or 2 bytes)

This field indicates the offending command class received by the destination node. The size of the command class field depends on the value of the first byte.

Offending Command (1 byte)

This field indicates the offending command received by the destination node.

4.2 Application Status Command Class, version 1

This command class contains commands that are not directly related to a specific functionality in the application, but are useful for maintaining an optimal Z-Wave system.

4.2.1 Compatibility Considerations

4.2.1.1 Node Information Frame (NIF)

A supporting node **MUST** always advertise the Application Status Command Class in its NIF, regardless of the inclusion status and security bootstrapping outcome.

4.2.2 Application Busy Command

The Application Busy Command used to instruct a node that the node that it is trying to communicate with is busy and is unable to service the request right now.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_APPLICATION_STATUS							
Command = APPLICATION_BUSY							
Status							
Wait Time							

Status (8 bits)

The status field can have the following values

Status	Description
0	Try again later
1	Try again in Wait Time seconds
2	Request queued, executed later

Wait Time (8 bits)

The wait time field indicates the number of seconds a node should wait before retrying the request.

4.2.3 Application Rejected Request Command

All supported commands are typically executed unconditionally and the only handshake is acknowledgement on the protocol level. Some applications can however be in a state where the application rejects to execute a supported command. The Application Rejected Request Command used to instruct a node that the command was rejected by the application in the receiving node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_APPLICATION_STATUS							
Command = APPLICATION_REJECTED_REQUEST							
Status							

Status (8 bits)

This field MUST be set to 0. The value 0 MUST indicate that the received (supported) command has been rejected by the application at the receiving node

4.3 Association Command Class, version 1

The Association Command Class is used to manage associations to NodeID destinations. A NodeID destination may be a simple device or the Root Device of a Multi Channel device.

An association group sends an unsolicited command to the configured destinations when triggered by an event. The parameters of the command may be dynamic, e.g. the temperature of a sensor reading or the light level for a dimmer.

4.3.1 Compatibility considerations

CC:0085.01.00.22.001 The Association Group Information (AGI) Command Class SHOULD be supported to enable automated discovery of association group properties.

CC:0085.01.00.22.002 It is RECOMMENDED that a node which implements the Association Command Class also implements the Multi Channel Association Command Class for compatibility with End Point destinations. For instance, a wall switch may be configured to control one specific outlet of a power strip if the wall switch supports Multi Channel Association.

4.3.2 Z-Wave Plus considerations

CC:0085.01.00.11.001 The Z-Wave Plus certification program mandates that association group 1 is reserved for the Lifeline association group. Association group 1 MUST NOT be assigned to any other use than the Lifeline group.

CC:0085.01.00.13.001 The actual Device Type of a given product specifies mandatory commands which the device must be able to send to a lifeline group destination. A manufacturer MAY add additional commands to the Lifeline group.

The Z-Wave Plus certification program mandates support for the Association Group Information (AGI) Command Class if a device supports the Association Command Class.

The Z-Wave Plus certification program recommends that a composite device is designed as a Multi Channel device.

4.3.3 Security considerations

CC:0085.01.00.41.001 A node which has been S0/S2 bootstrapped MUST NOT accept Association commands unless the commands are received via the highest security key granted to the node during bootstrapping.

4.3.4 Association Set Command

This command is used to add destinations to a given association group.

CC:0085.01.01.12.001 The receiving node **SHOULD** add the specified NodeID destinations to the specified association group.
 CC:0085.01.01.13.001 This command **MAY** be ignored if the association group is already full.

CC:0085.01.01.11.001 Routing slaves **MUST** have return routes assigned to all association destinations.

CC:0085.01.01.51.001 Unless the association destination is a gateway, a controlling node **MUST NOT** create an association if the association destination node does not support the controlling commands (Set/Get types) that the actual association group will be sending. The AGI Command Class **MUST** be used to probe the commands that a given association group will be sending.
 CC:0085.01.01.51.002
 CC:0085.01.01.52.001 A controlling node **SHOULD NOT** create an association if the source and destination nodes are bootstrapped with different security levels.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION							
Command = ASSOCIATION_SET							
Grouping Identifier							
NodeID 1							
..							
NodeID N							

Grouping Identifier (8 bits)

CC:0085.01.01.11.002 This field is used to specify the actual association group. Grouping Identifiers **MUST** be assigned in a consecutive range starting from 1.

CC:0085.01.01.11.003 A receiving node **MUST** ignore an unsupported Grouping Identifier.

NodeID (N bytes)

This field specifies a list of NodeIDs that are to be added to the specified association group.

4.3.5 Association Get Command

This command is used to request the current destinations of a given association group.

CC:0085.01.02.11.001 The Association Report Command **MUST** be returned in response to this command.

CC:0085.01.02.11.002 This command **MUST NOT** be issued via multicast addressing.

CC:0085.01.02.11.003 A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION							
Command = ASSOCIATION_GET							
Grouping Identifier							

Grouping Identifier (8 bits)

CC:0085.01.02.11.004 This field is used to specify the actual association group. Grouping Identifiers **MUST** be assigned in a consecutive range starting from 1.

CC:0085.01.02.12.001 A node that receives an unsupported Grouping Identifier **SHOULD** return information relating to Grouping Identifier 1.

4.3.6 Association Report Command

This command is used to advertise the current destinations of a given association group.

If the node supports the Multi Channel Association Command Class,

- the node **MUST** advertise the same Node ID destinations in this report as in the Multi Channel Association Report Command
- the node **MUST NOT** advertise the NodeID of End Point destinations in this report.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION							
Command = ASSOCIATION_REPORT							
Grouping Identifier							
Max Nodes Supported							
Reports to Follow							
NodeID 1							
...							
NodeID N							

Grouping Identifier (8 bits)

This field is used to advertise the actual association group. Grouping Identifiers **MUST** be assigned in a consecutive range starting from 1.

Max Nodes Supported (8 bits)

The maximum number of destinations supported by the advertised association group. Each destination **MAY** be a NodeID destination or an End Point destination (if the node supports the Multi Channel Association Command Class).

Reports to Follow (8 bits)

The entire list destinations of the advertised association group may be too long for one command. This field **MUST** advertise how many report frames will follow this report.

NodeID (N bytes)

This field advertises a list of NodeID destinations of the advertised association group.

The list of NodeIDs **MUST** be empty if there are no NodeID destinations configured for the advertised association group.

4.3.7 Association Remove Command

This command is used to remove destinations from a given association group.

CC:0085.01.04.11.001

If the node supports the Multi Channel Association Command Class the node **MUST NOT** remove End Point destinations in response to this command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION							
Command = ASSOCIATION_REMOVE							
Grouping Identifier							
NodeID 1							
...							
NodeID N							

Grouping Identifier (8 bits)

This field is used to specify from which association group the specified NodeID destinations should be removed.

CC:0085.01.04.11.002

Grouping Identifiers **MUST** be assigned in a consecutive range starting from 1.

CC:0085.01.04.11.003

This field **MUST** be interpreted in combination with the NodeID field.

CC:0085.01.04.11.004

A node that receives an unsupported Grouping Identifier **MUST** ignore this command.

NodeID (N bytes)

This field specifies a list of NodeID destinations that are to be removed from the specified association group.

CC:0085.01.04.11.005

The Grouping Identifier and NodeID fields **MUST** be interpreted as indicated in Table 1.

Table 1, Association Remove, V1::Parameter interpretation

Grouping identifier	Number of NodeIDs	Interpretation
> 0	> 0	Remove specified NodeIDs from the specified association group (MANDATORY V1)
> 0	= 0	Remove all NodeIDs from the specified association group (RECOMMENDED V1)
= 0	≥ 0	(Reserved V1)

CC:0085.01.04.11.006

A sending node **MUST NOT** send reserved parameter combinations and a receiving node **MUST** ignore reserved parameter combinations.

4.3.8 Association Supported Groupings Get Command

This command is used to request the number of association groups that this node supports.

CC:0085.01.05.11.001

The Association Supported Groupings Report Command MUST be returned in response to this command.

CC:0085.01.05.11.002

This command MUST NOT be issued via multicast addressing.

CC:0085.01.05.11.003

A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION							
Command = ASSOCIATION_GROUPINGS_GET							

4.3.9 Association Supported Groupings Report Command

This command is used to advertise the maximum number of association groups implemented by this node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION							
Command = ASSOCIATION_GROUPINGS_REPORT							
Supported Groupings							

Supported Groupings (8 bits)

This field is used to advertise the number of association groups that this node supports.

CC:0085.01.06.11.001

Grouping Identifiers MUST be assigned in a consecutive range starting from 1.

4.4 Association Command Class, version 2

The Association Command Class is used to manage associations to NodeID destinations. A NodeID destination may be a simple device or the Root Device of a Multi Channel device.

The following sections specify commands which were extended or added in version 2.

4.4.1 Compatibility considerations

CC:0085.02.00.21.001

A device supporting this command class version **MUST** also support the Association Command Class, version 1.

This version introduces

- New methods for the removal of associations via the Association Remove Command
- New commands
 - Association Specific Group Get Command
 - Association Specific Group Report Command

The considerations of section 4.3.1 also apply to this version.

4.4.2 Z-Wave Plus considerations

The considerations of section 4.3.2 also apply to this version.

4.4.3 Security considerations

The considerations of section 4.3.3 also apply to this version.

4.4.4 Association Remove Command

The Association Remove Command is used to remove NodeID destinations from a given association group.

CC:0085.02.04.11.001

If the node supports the Multi Channel Association Command Class the node MUST NOT remove End Point destinations in response to this command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION							
Command = ASSOCIATION_REMOVE							
Grouping Identifier							
NodeID 1							
...							
NodeID N							

Grouping identifier (8 bits)

This field is used to specify from which association group the specified NodeID destinations are to be removed.

CC:0085.02.04.11.002

This field MUST be interpreted in combination with the NodeID field.

CC:0085.02.04.11.003

A node that receives an unsupported Grouping Identifier MUST ignore this command; with the exception of Grouping Identifier 0, which MUST be accepted. Refer to Table 2.

NodeID (N bytes)

CC:0085.02.04.11.004

This field specifies a list of NodeID destinations that are to be removed. The Grouping Identifier and NodeID fields MUST be interpreted as indicated in Table 2:

Table 2, Association Remove, V2::Parameter interpretation

Grouping identifier	Number of NodeIDs	Interpretation
> 0	> 0	Remove destination NodeIDs from association group (MANDATORY V1, MANDATORY V2)
> 0	= 0	Remove all destination NodeIDs from association group (RECOMMENDED V1, MANDATORY V2)
= 0	> 0	Remove destination NodeIDs from all association groups (Reserved V1, MANDATORY V2)
= 0	= 0	Remove all destination NodeIDs from all association groups (Reserved V1, MANDATORY V2)

4.4.5 Association Specific Group Get Command

This command allows a portable controller to interactively create associations from a multi-button device to a destination that is out of direct range.

CC:0085.02.0B.12.001
CC:0085.02.0B.11.001
CC:0085.02.0B.12.002

It is **OPTIONAL** for a device to support this functionality. However, a receiving device **MUST** always return the Association Specific Group Report Command in response to this command.
If the device does not support this functionality, a Group parameter value of 0 **SHOULD** be advertised in the Association Specific Group Report Command that is returned in response to this command.

This functionality allows a supporting multi-button device to detect a key press and subsequently advertise the identity of the key. The following sequence of events takes place:

- The user activates a special identification sequence and pushes the button to be identified
- The device issues a Node Information frame (NIF)
- The NIF allows the portable controller to determine the NodeID of the multi-button device
- The portable controller issues an Association Specific Group Get Command to the multi-button device
- The multi-button device returns an Association Specific Group Report Command that advertises the association group that represents the most recently detected button

CC:0085.02.0B.12.003

The Association Group Information (AGI) Command Class provides a centralised alternative for the discovery of available association groups and the capabilities of these groups. A device supporting this functionality **SHOULD** also support the Association Group Information (AGI) Command Class.

CC:0085.02.0B.11.002
CC:0085.02.0B.11.003

This command **MUST NOT** be issued via multicast addressing.
A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION							
Command = ASSOCIATION_SPECIFIC_GROUP_GET							

CC:0085.02.0B.12.004

Clarification: Previous text revisions of this specification presented conflicting guidelines for the Association Specific Group Get Command when received by non-supporting devices. The values 0 and 1 were both suggested for the Group field of the Association Specific Group Report Command. It is **RECOMMENDED** that the value 0 is returned by non-supporting devices.

4.4.6 Association Specific Group Report Command

This command is used to advertise the association group that represents the most recently detected button.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION							
Command = ASSOCIATION_SPECIFIC_GROUP_REPORT							
Group							

Group (8 bits)

This field is used to advertise the association group that represents the most recently detected button.

CC:0085.02.0C.11.001 The value of this field **MUST** be in the range 0..255.

CC:0085.02.0C.11.002 If a supporting device implements a multi-button device, the Group field **MUST** advertise an association group which represents the most recently activated button. The actual association group **SHOULD** be able to control an actuator device, e.g. via the Basic Command Class.

CC:0085.02.0C.12.001

CC:0085.02.0C.12.002 This field **SHOULD** be set to 0 if the functionality is not supported or if the most recent button event does not map to an association group.

Clarification: Previous text revisions of this specification presented conflicting guidelines for the Association Specific Group Get Command when received by non-supporting devices.

CC:0085.02.0C.12.003 The values 0 and 1 were both suggested for the Group field of the Association Specific Group Report Command. It is **RECOMMENDED** that the value 0 is returned by non-supporting devices.

4.5 Association Command Configuration Command Class, version 1

The Association Command Configuration Command Class defines the commands necessary for a 2nd node to add and delete commands to NodeIDs in a group as defined in the Association Command Class in a 1st node.

The device MUST implement the Association Command Class as 'supported'

The Association Command Class and the Command Configuration Command Class MUST be linked through the following dependencies:

- Nodes added to an association through an Association Set or Association Composite Set MUST be reported in Command Configuration Reports with the Command Class and command identifiers transmitted to the nodes.
- Nodes added to an association through a Command Configuration Set MUST also be reported in an Association Report and Association Composite Report
- All commands associated to a grouping identifier//NodeID pair will be removed as a result of an Association Remove. The related command records will be released
- Command(s) associated to a grouping identifier/NodeID/endpoint pair will be removed as a result of an Association Composite remove. The related command(s) record will be released.

The memory consumption of supporting full command sizes in all combinations of groupings identifiers and Node IDs is very extensive; hence the command class supports a memory flexible implementation.

The Command Class allows a device to support a number of command records. A command record consists of the grouping identifier, the Node ID and the command. The size of the command MAY be restricted by the device through the Max command length field. The command MUST be the complete command needed (i.e. All relevant encapsulations MUST be included in the command).

When no command records are free (all has been used), new commands MUST NOT be allocated to Node IDs before one or more command records have been freed up (through the Association Remove Command)

If the 2nd node runs out of free command records before it has finalized its command configurations, it MUST accept that the application on the device has full control of the remaining Node IDs. Alternatively the 2nd node can abort the command configuration process. In this case it is RECOMMENDED that the 2nd node free up the used command records in the aborted command configuration attempt.

A device can report the maximum number of command records, the number of free command records, and the max command length supported through the Command Records Supported Report.

In order to support sharing knowledge of how a device controls nodes without using extensive memory resources a Configurable Cmd field is supported. When configurable Cmd= 0x0 then a 2nd node can only monitor the commands. It cannot control them.

The V/C field allows a device to decrease the memory utilization to a minimum. When a device reports V/C=0x01, then the Command Configuration Set and Command Configuration Report MUST always use command class identifier and command identifier equal to a Basic Set Command Records Supported Get. This allows the device to only store the value field and thereby save memory resources.

4.5.1 Command Records Supported Get Command

The Command Records Supported Get Command is used to request the number of free command records available (grouping Identifier, Node ID, command), the maximum command records supported in the device and information regarding the maximum command length supported in the device.

The maximum number of groupings the given node supports is available through the Association Command Class.

The Command Records Supported Report Command **MUST** be returned in response to this command.

This command **MUST NOT** be issued via multicast addressing.

A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION_COMMAND_CONFIGURATION							
Command = COMMAND_RECORDS_SUPPORTED_GET							

4.5.2 Command Records Supported Report Command

The Command Records Supported Report Command used to report information regarding the Command records.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION_COMMAND_CONFIGURATION							
Command = COMMAND_RECORDS_SUPPORTED_REPORT							
Max command length						V/C	Conf. Cmd
Free Command records 1							
Free Command records 2							
Max Command records 1							
Max Command records 2							

Configurable Cmd (1 bit)

Configurable Cmd	Functionality
0	The local application has full control of the commands associated with the grouping. The commands can be monitored from a 2 nd network node.
1	The specific commands associated with the grouping can be controlled and monitored from a 2 nd network node. This option includes also the level field used by the command <i>Transfer Scene</i> in the <i>Controller Replication Command Class</i> . In this case the level value is transferred via the command <i>Basic Set</i> in the <i>Basic Command Class</i> .

V/C (1 bit)

V/C	Functionality
0	Command type. A Z-Wave command can be added to the node
1	Value type. A Value field is specified for a Node ID using the command <i>Basic Set</i> in the <i>Basic Command Class</i> . Level field originates from the command <i>Transfer Scene</i> in the <i>Controller Replication Command Class</i> .

Max command length (6 bits)

If Configurable Cmd is equal to 0x0, the field SHOULD return 0x0.

If Configurable Cmd is equal to 0x1 the field SHOULD return the maximum length of a command which can be associated to a node in a grouping. The minimum max command length allowed is 0x03.

Example:

A product reports a Max command length = 0x03. In this case the product can be programmed with a Multilevel Switch Set, but not a Switch Multilevel Start Level Change.

Multilevel Switch Set has a command length of 3

Multilevel Switch Start Level Change has a command length of 4

Free Command Records (16 bits)

The field specifies the current number of free Command Records which can be configured in the device through the Command Configuration Set Command.

Max Command records (16 bits)

The field specifies the maximum number of Command Records which can be configured in the device through the Command Configuration Set Command.

4.5.3 Command Configuration Set Command

The Command Configuration Set Command used to specify which commands SHOULD be sent to nodes within a given Grouping identifier.

Every Command Configuration Set will utilize one Command record from the pool of free Command records.

If the device has no free command records when receiving the Command Configuration Set, the command will be ignored.

The Application on the node may alter the commands when needed.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION_COMMAND_CONFIGURATION							
Command = COMMAND_CONFIGURATION_SET							
Grouping identifier							
Node ID							
Command length							
Command Class identifier							
Command identifier							
Command byte 1							
..							
Command byte n							

Grouping identifier (8 bits)

This grouping identifier used to specify how nodes are grouped together. The grouping identifier values MUST be a sequence starting from 1. This field MAY be ignored in case the node only supports one group.

Node ID (8 bits)

This field contains the node ID within the grouping specified, that should receive the command.

Command length (8 bits)

This field specifies the complete command length (including command class and command identifiers).

Example: Switch Multilevel Set 0x20. The command length field MUST be equal to 0x03.

Command Class Identifier (8 bits)

This field contains the identifier of the command class which should be sent to the Node ID. In case the Configurable Cmd field is equal to 0x0 is this field and the following ignored and can therefore be omitted.

In case the device has reported V/C=0x1, the Command Class and Command Identifiers MUST be equal to Basic Set command.

Command identifier (8 bits)

This field contains the identifier of the Command, which should be sent to the specified Node ID. In case the Configurable Cmd field is equal to 0x0 is this field ignored and can therefore be omitted.

In case the device has reported V/C=0x1, the Command Class and Command Identifiers **MUST** be equal to Basic Set Command.

Command byte (N bytes)

These fields contain the command parameters, which should be sent to the Node ID. In case the Configurable Cmd field is equal to 0x0 is these fields ignored and can therefore be omitted.

4.5.4 Command Configuration Get Command

The Command Configuration Get Command is used to request the commands specified for to a Node ID within a given Grouping identifier.

The Command Configuration Report Command **MUST** be returned in response to this command.

This command **MUST NOT** be issued via multicast addressing.

A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION_COMMAND_CONFIGURATION							
Command = COMMAND_CONFIGURATION_GET							
Grouping identifier							
Node ID							

Grouping identifier (8 bits)

This group identifier used to specify how nodes are grouped together. The group identifier values **MUST** be a sequence starting from 1. This field **MUST** be ignored in case the node only supports one group.

Node ID (8 bits)

This field specifies the node ID within the grouping.

4.5.5 Command Configuration Report Command

The Command Configuration Report Command used to report the commands specified for a Node ID within a given Grouping identifier.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION_COMMAND_CONFIGURATION							
Command = COMMAND_CONFIGURATION_REPORT							
Grouping identifier							
Node ID							
First	Reserved			Reports to follow			
Command length							
Command Class identifier							
Command identifier							
Command byte 1							
...							
Command byte N							

Grouping identifier (8 bits)

This group identifier identifies the group. The group identifier values **MUST** be a sequence starting from 1.

Node ID (8 bits)

This field contains the node ID as requested in the Association Command Get.

Reports to follow (4 bits)

This value indicates how many report frames left before transferring the entire list of commands.

First (1 bit)

This field indicates that this report is the first report relating to a Grouping identifier/Node ID pair

Command length (8 bits)

This field specifies the complete command length (including command class and command identifiers).

Example: Multilevel Switch Set 0x20. The command length field **MUST** be equal to 0x03.

Command Class Identifier (8 bits)

This field contains the identifier of the command class which is sent to the Node ID.

In case the device has reported V/C=0x1, the Command Class and Command Identifiers **MUST** be equal to Basic Set command

Command identifier (8 bits)

This field contains the identifier of the Command which is sent to the specified Node ID.

In case the device has reported V/C=0x1, the Command Class and Command Identifiers **MUST** be equal to Basic Set Command

Command byte (N bytes)

These fields contain the command parameter which is sent to the Node ID.

4.6 Association Group Information (AGI) command class, version 1

The Association Group Information (AGI) Command Class allows a node to advertise the capabilities of each association group supported by a given application resource.

CC:0059.01.00.12.001

Controllers and installer tools SHOULD use AGI information to support controller-assisted button-to-button association and GUI-based drag-and-drop association in a plug-and-play fashion.

Centralized gateway-based deployments may create a single association from the lifeline association group to a central management application.

4.6.1 Compatibility considerations

CC:0059.01.00.21.001

A node supporting the AGI Command Class MUST support the Association Command Class.

4.6.1.1 Multi Channel considerations

The Association Group Information (AGI) Command Class also applies to Multi Channel devices.

CC:0059.01.00.21.002

If a node implements Multi Channel End Points and supports the AGI Command Class, each individual End Point MUST support the AGI Command Class.

CC:0059.01.00.21.003

Functions such as Device Reset Locally, low battery notification or tamper alarm which do not relate to the actual application functionality of the product MUST reside in the Root Device if the product is implemented as a Multi Channel device.

4.6.2 Association Principles

A light control transmitter may have one or more keys. A key is mapped to one or more association groups.



Figure 1, Light control transmitter (example)



Figure 2, Lamp module (example)

A light control transmitter key can be configured to control a lamp by adding the NodeID of the lamp to an association group that represents the key.

4.6.3 The Association Group Information

CC:0059.01.00.12.002

A node may implement one or more association groups. If the device implements association groups, the device SHOULD provide an Association Group Information (AGI) table as described in Table 3.

Table 3, Layout of one line of the Association Group Information table

Group identifier	Profile 2 bytes	Command Class & Command (list) N bytes	Group Name (UTF-8) M bytes
------------------	--------------------	---	----------------------------------

An AGI table entry carries a number of fields. The information is typically static and can be defined at compile time. Thus, the AGI table does not need to occupy any RAM or non-volatile storage.

4.6.3.1 Group identifier

The Group identifier indicates which association group the actual table entry relates to. As association groups are always numbered in a sequence starting from 1, an actual implementation does not have to store the group identifier in its memory. The number of association groups may be requested via the Association Groupings Get or Multi Channel Association Groupings Get commands.

CC:0059.01.00.11.001 Association group 1 is reserved for the Z-Wave Plus Lifeline association group. Group 1 MUST NOT be assigned to any other use than the Lifeline group. The actual Device Type specifies a mandatory list of commands which the device MUST send to all targets associated to the Lifeline group. A manufacturer MAY add additional commands to the Lifeline group. Refer to [10].

CC:0059.01.00.11.002
CC:0059.01.00.12.003 The Lifeline group MUST be advertised for the Root Device of a Multi Channel device. A Multi Channel End Point SHOULD advertise commands for association group 1 which End Points can send via the Root Device Lifeline group if a Multi Channel Association is created from the Root Device Lifeline group.

4.6.3.2 Profile

CC:0059.01.00.13.002 A device MAY implement one or several AGI profiles. The profile defines the scope of events which triggers the transmission of commands to the actual association group. As an example a temperature sensor may issue different Basic Set Command parameters when the temperature exceeds a threshold and when the temperature drops below a threshold. The actual behavior is application dependent and out of scope of the AGI Command Class.

The profile identifiers are referenced in Table 8. Profiles are divided into categories in the following sections.

4.6.3.2.1 General profiles

CC:0059.01.00.11.004 The “General” category comprises the “Not Applicable” and “Lifeline” profiles. The “Not Applicable” profile identifier MUST be advertised if the actual association group does not match any of the defined profiles.

CC:0059.01.00.11.005 The “Lifeline” profile identifier MUST be advertised for association group 1 of the Root Device.


4.6.3.2.2 Control profiles


“Control” profiles are intended for association groups of which commands are triggered by a user control mechanism, such as keys or buttons. As an example, a light control transmitter may comprise two control keys that can each control a group of lamps.

A control key may control up and down dimming as well as on/off state. Thus, a control key may send more than one command with one specific parameter. A control key may be implemented as a physical key, a group of icons, touch screen gestures or other means. The actual implementation is out of scope of this command class.

In case of multiple logical functions in a device, the device MAY implement one Multi Channel End Point for each logical function; e.g. each push button is implemented as a separate End Point.

Table 4 gives an example of a Multi Channel a battery-powered two-button light control device.

Root Device: 			
Group identifier	Profile 2 bytes	Command Class & Command (list) N bytes	Group Name (UTF-8) M bytes
1	General:Lifeline	Central Scene Notification Notification Report Battery Report Device Reset Locally Notification	Lifeline
2	Control:Key1	Basic Set	On/Off control (Button 1)
3	Control:Key1	Multilevel Switch Set	Dimmer control (Button 1)
4	Control:Key2	Basic Set	On/Off control (Button 2)
5	Control:Key2	Multilevel Switch Set	Dimmer control (Button 2)

End Point 1: 			
Group identifier	Profile 2 bytes	Command Class & Command (list) N bytes	Group Name (UTF-8) M bytes
1	General:Lifeline	-	Lifeline
2	Control:Key1	Basic Set	On/Off control (Button 1)
3	Control:Key1	Multilevel Switch Set	Dimmer control (Button 1)


End Point 2: 			
Group identifier	Profile 2 bytes	Command Class & Command (list) N bytes	Group Name (UTF-8) M bytes
1	General:Lifeline	-	Lifeline
2	Control:Key2	Basic Set	On/Off control (Button 2)
3	Control:Key2	Multilevel Switch Set	Dimmer control (Button 2)

Table 4, Example: AGI tables for two-button light control transmitter

The Profile identifiers Control:Key 1 and Control:Key 2 allows an installer tool to determine which Multi Channel End Point and which association group to use in order to configure buttons to control a light dimmer. The Group Name allows a human user to make a qualified decision even if the installer tool is not, e.g. because it is running an older version not updated with newer profile identifiers.

End Points 1 and 2 in the above example do not support any Command Class that must be sent via the Z-Wave Plus Lifeline. Therefore End Points 1 and 2 advertise an empty Command Class list for the Lifeline association group.


The Root Device in the above example advertises the Basic Set Command in association group 2. This is a feature of End Point 1. A controlling node **SHOULD NOT** create associations from other Root Device association groups than the Lifeline group if it supports the Multi Channel Command Class. The purpose of the Root Device Association Groups (with the exception of the Lifeline group) is only to provide backwards compatibility for legacy devices without support for the Multi Channel Command Class.


4.6.3.2.3 Sensor profiles

“Sensor” profiles are intended for association groups of which commands are triggered by sensor readings. As an example, a sensor product could comprise two temperature sensors.

Sensor profile identifiers are constructed by prepending the Multilevel Sensor Command Class identifier (referred to with AGI_PROFILE_SENSOR) to a multilevel sensor type defined in the Multilevel Sensor Command Class. The values in Table 8 only serve as examples to illustrate the construction of AGI Sensor Profile Identifiers. For the full list of available sensor types, refer to the “Multilevel Sensor Report Command”.

In case of multiple logical functions in a device, the device **SHOULD** implement one Multi Channel End Point for each logical function; e.g. each sensor instance is implemented as a separate End Point. Table 5 gives an example of a Multi Channel battery-powered sensor device with two multilevel sensor functions:

 Root Device:			
Group identifier	Profile 2 bytes	Command Class & Command (list) N bytes	Group Name (UTF-8) M bytes
1	General:Lifeline	Multilevel Sensor Report Notification Report Battery Report Device Reset Locally Notification	Lifeline
2	Sensor:Temperature	Basic Set	On/Off control (Indoor temperature)
3	Sensor:Temperature	Basic Set	On/Off control (Outdoor temperature)

 End Point 1:			
Group identifier	Profile 2 bytes	Command Class & Command (list) N bytes	Group Name (UTF-8) M bytes
1	Sensor:Temperature	Multilevel Sensor Report	Indoor temperature via Lifeline
2	Sensor:Temperature	Basic Set	On/Off control (Indoor temperature)


 End Point 2:			
Group identifier	Profile 2 bytes	Command Class & Command (list) N bytes	Group Name (UTF-8) M bytes
1	Sensor:Temperature	Multilevel Sensor Report	Outdoor temperature via Lifeline
2	Sensor:Temperature	Basic Set	On/Off control (Outdoor temperature)

Table 5, Example: AGI tables for two-function sensor

The Profile identifier is Sensor:Temperature for both End Points but the Group Name allows a user to determine which End Point and which association group to use in order to configure the indoor temperature sensor to control a heating element.

The Root Device in the above example advertises the Multilevel Sensor Report command in the Lifeline group. Residing in a Multi Channel device, the Root Device does not actually implement any application functionality. The Multilevel Sensor Report Command is a feature of the End Points which is advertised for backwards compatibility with legacy devices not supporting the Multi Channel Command Class. Likewise, the Root device advertises the Basic Set Command in association groups 2 and 3. This is also a feature of the End Points. Refer to the Multi Channel Command Class for details on backwards compatibility.

End Point 1 in the above example advertises the Multilevel Sensor Report Command in association group 1. By advertising that zero NodeIDs are supported for association group 1, End Point 1 indicates

that this command is sent via the Root Device Lifeline group if a Multi Channel association is created for the Root Device Lifeline group. The same principle applies to End Point 2.

4.6.3.2.4 Notification profiles

“Notification” profiles are intended for association groups of which commands are triggered by detected events or state changes. As an example, a detector product could comprise one smoke sensor and one CO2 detector.

Notification profile identifiers are constructed by prepending the Notification Command Class identifier (referred to with AGI_PROFILE_NOTIFICATION) to a notification type defined in the Notification Command Class. The values below only serve as an example to illustrate the construction of AGI Notification Profile identifiers. For the full list of available notification types, refer to the Notification Command Class [16].

CC:0059.01.00.12.006

In case of multiple logical functions in a device, the device **SHOULD** implement one Multi Channel End Point for each logical function.

The following example outlines how a standalone smoke alarm is represented by an AGI table:


<div>Root Device: </div>			
Group identifier	Profile 2 bytes	Command Class & Command (list) N bytes	Group Name (UTF-8) M bytes
1	General:Lifeline	Notification Report Battery Report Device Reset Locally Notification	Lifeline
2	Notification:SmokeAlarm	Basic Set	On/Off control (Smoke)

Table 6, Example: AGI table for one-function alarm device

The Profile identifier Notification:SmokeAlarm indicates to an installer that the Basic Set Command of association group 2 is issued in response to smoke alarm events.

4.6.3.3 Command Class and Command

CC:0059.01.00.13.004

This field contains the command class and the command that is sent to targets associated with this association group if an event occurs. This field **MAY** advertise a list of commands and **MAY** contain a combination of extended and normal command classes. Refer to 4.6.9.

The command list is used by a controlling application to discover what functionality the node provides via a given association group. It also allows the controlling node to ensure that the associated node understands the received commands.

CC:0059.01.00.13.005

A node **MAY** list only one command if the actual association group sends different commands which actually relate to the same overall functionality and belong to the same Command Class and Commands Class version (e.g. Multilevel Switch Set / Multilevel Switch Start Level Change / Multilevel Stop Level Change)

CC:0059.01.00.12.007

In case Multi Channel End Points are implemented, the Lifeline association group (group 1) of End Points **SHOULD** be used to advertise commands that will be sent via the Root Device Lifeline group if a Multi Channel association is created from the Root Device Lifeline group.

4.6.3.4 Association group name

This field contains a name reflecting the purpose of the group, e.g. “On/Off control (Smoke)”.

CC:0059.01.00.11.006

The Root Device Lifeline group MUST be named “Lifeline”.

CC:0059.01.00.11.007

Group names MUST be encoded using UTF-8. The available capacity for characters depends on the actual characters encoded with UTF-8. Plain ASCII characters only occupy one byte while special characters may need two or more bytes for representation.

4.6.4 Association Group Name Get

This command is used to query the name of an association group.

CC:0059.01.01.11.001

The Association Group Name Report Command MUST be returned in response to this command.

CC:0059.01.01.11.002

This command MUST NOT be issued via multicast addressing.
A receiving node MUST NOT return a response if this command is received via multicast addressing.
The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION_GRP_INFO							
Command = ASSOCIATION_GROUP_NAME_GET							
Grouping Identifier							

Grouping Identifier (1 byte)

This field is used to specify the requested association group identifier.

CC:0059.01.01.12.001

A node receiving this command for an unsupported Grouping Identifier SHOULD return information relating to Grouping Identifier 1.

4.6.5 Association Group Name Report

This command is used to advertise the assigned name of an association group.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION_GRP_INFO							
Command = ASSOCIATION_GROUP_NAME_REPORT							
Grouping Identifier							
Name Length							
Name 1							
...							
Name N							

Grouping Identifier (1 byte)

This field is used to advertise the actual association group identifier.

Name Length (1 byte)

CC:0059.01.02.11.001

This field indicates the length in bytes of the Name field. The value MUST be in the range 0..42.

Name (N bytes)

This field is used to indicate the assigned name for the actual Group Identifier.

CC:0059.01.02.11.002

The length of this field in bytes MUST comply with the advertised value in the Name Length field.

CC:0059.01.02.11.003

The string MUST NOT contain any appended termination characters.

CC:0059.01.02.11.004

The characters MUST be encoded in UTF-8 format.

4.6.6 Association Group Info Get

This command is used to request the properties of one or more association group.

CC:0059.01.03.11.001

The Association Group Info Report MUST be returned in response to this command.

CC:0059.01.03.11.002

This command MUST NOT be issued via multicast addressing.

A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION_GRP_INFO							
Command = ASSOCIATION_GROUP_INFO_GET							
Refresh cache	List Mode	Reserved					
Grouping Identifier							

List Mode (1 bit)

This field is used to request the properties of the supported association groups of a node.

- CC:0059.01.03.11.003
CC:0059.01.03.13.001 If List Mode is set to 1, a receiving node **MUST** ignore the Grouping Identifier field and return the properties of all its supported associations groups. The receiving node **MAY** return the response in several Reports, (e.g. due to memory constraints or a high number of association groups).
- CC:0059.01.03.11.004 If List Mode is set to 0, a receiving node **MUST** advertise the properties of the association group identified by the Grouping Identifier.
- CC:0059.01.03.11.005 The Association Group Info Report returned in response to this command **MUST** advertise the same List Mode value as specified by this field.

Refresh Cache (1 bit)

- CC:0059.01.03.11.006 If AGI information is transferred via a gateway, the gateway **MUST** cache information for all nodes; also listening nodes.
The AGI Get command flag "Refresh Cache" is used by a management application to instruct a gateway to update its cache on the first chance given. In case of a sleeping device, this may require a user operation to wake up the device. In case of a Wake Up device, the gateway may wait for the next Wake Up Notification from the actual device.
- CC:0059.01.03.11.007
CC:0059.01.03.11.008 The "Refresh Cache" value 0 **MUST** specify that the gateway is to return its cached information.
The "Refresh Cache" value 1 **MUST** specify that the gateway is to return its cached information and update its cache on the first chance given.
- CC:0059.01.03.12.001 The "Refresh Cache" **SHOULD** be set to 0 by a sending node.
A receiving node may have re-configurable AGI properties, e.g. if it is a configurable multi-purpose remote control. If the "Dynamic Info" flag of the AGI Report command is set to 1, a sending node **MAY** set the "Refresh Cache" flag in the AGI Get command to force the gateway to update its cache, e.g. after the sending node has re-configured the AGI properties of the receiving node.
- CC:0059.01.03.13.002

Reserved

- CC:0059.01.03.11.009 This field **MUST** be set to 0 by a sending node and **MUST** be ignored by a receiving node.

Grouping Identifier (1 byte)

- CC:0059.01.03.11.00A This field is used to specify the requested association group identifier. This value **MUST** be ignored if the List Mode field is set to 1.
- CC:0059.01.03.12.002 A node receiving this command for an unsupported Grouping Identifier **SHOULD** return information relating to Grouping Identifier 1.

4.6.7 Association Group Info Report

This command is used to advertise the properties of one or more association groups.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION_GRP_INFO							
Command = ASSOCIATION_GROUP_INFO_REPORT							
List mode	Dynamic Info	Group Count					
Grouping Identifier							
Mode = 0							
Profile MSB							
Profile LSB							
Reserved = 0							
Event Code MSB = 0							
Event Code LSB = 0							
...							
Grouping Identifier							
Mode = 0							
Profile MSB							
Profile LSB							
Reserved = 0							
Event Code MSB = 0							
Event Code LSB = 0							

List Mode (1 bit)

- CC:0059.01.04.11.001 The List Mode field **MUST** advertise the same value as specified in the Association Group Info Get command which caused this command to be returned.
- CC:0059.01.04.13.001 If List Mode is 1, a sending node is advertising the properties of all association groups. The sending node **MAY** return several Reports.
- CC:0059.01.04.11.002 If List Mode is 0 a sending node **MUST** advertise the properties of one association group.

Dynamic Info (1 bit)

- CC:0059.01.04.13.002 If the Dynamic Info field is set to 1, the information **MAY** change and a controlling node **SHOULD** perform periodic cache refresh for this node. Nodes **MUST** set this bit if they are able to change the Association Group Information on the fly.
- CC:0059.01.04.11.003
- CC:0059.01.04.11.004 If this field is set to 0, a controlling node **MUST NOT** request this information more than one time.

Group Count (6 bits)

CC:0059.01.04.11.005 This field indicates the number of association groups advertised in the command. The Grouping Identifier, Mode, Profile, Reserved and Event Code fields MUST be repeated for each association group.

CC:0059.01.04.11.006 If List Mode is set to 0, the Group Count field MUST be set to 1.
If List Mode is set to 1, the Group Count field MUST advertise the number of association group property blocks following this field.

A node receiving this report may determine the total number of association groups via the Association Supported Groupings Get or Multi Channel Association Supported Groupings Get commands.

Grouping Identifier (1 byte)

CC:0059.01.04.11.007 This field MUST advertise the association group identifier of the actual association group property block.

Mode (1 byte)

CC:0059.01.04.11.008 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Profile (2 bytes)

CC:0059.01.04.11.009 This field is used to advertise the profile of the actual association group. Refer to section 4.6.3.2 and Table 8. This field MUST be encoded according to Table 8.

Reserved

CC:0059.01.04.11.00A This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Event Code (2 bytes)

CC:0059.01.04.11.00B This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

4.6.8 Association Group Command List Get

This command is used to request the commands that are sent via a given association group.

CC:0059.01.05.11.001 The Association Group Command List Report **MUST** be returned in response to this command.

CC:0059.01.05.11.002 This command **MUST NOT** be issued via multicast addressing.
A receiving node **MUST NOT** return a response if this command is received via multicast addressing.
The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION_GRP_INFO							
Command = ASSOCIATION_GROUP_COMMAND_LIST_GET							
Allow cache	Reserved						
Grouping Identifier							

Allow Cache (1 bit)

This field indicates that a Z-Wave Gateway device is allowed to intercept the request and return a cached response on behalf of the specified target.

CC:0059.01.05.11.003 If this bit is not set to 1, a Z-Wave Gateway device **MUST** forward the request to the specified target.

CC:0059.01.05.12.001 A requesting node **SHOULD** allow caching to save network bandwidth.

CC:0059.01.05.11.004 A Z-Wave Gateway device **MUST** cache information for all nodes; also listening nodes. This will save network bandwidth.

CC:0059.01.05.11.005 A Z-Wave Gateway device **MUST** forward the request if it does not hold cached information for the specified target. The Z-Wave Gateway device **MUST** cache the data returned.

Reserved

CC:0059.01.05.11.006 This field **MUST** be set to 0 by a sending node and **MUST** be ignored by a receiving node.

Grouping Identifier (1 byte)

This field is used to specify the requested association group identifier.

CC:0059.01.05.12.002 A node that receives an unsupported Grouping Identifier **SHOULD** return information relating to Grouping Identifier 1.

4.6.9 Association Group Command List Report

This command is used to advertise the commands that are sent via an actual association group.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION_GRP_INFO							
Command = ASSOCIATION_GROUP_COMMAND_LIST_REPORT							
Grouping Identifier							
List Length							
Command Class 1 (1 or 2 bytes)							
Command 1							
...							
Command Class N (1 or 2 bytes)							
Command N							

Grouping Identifier (1 byte)

This field is used to advertise the actual association group identifier.

List Length (1 byte)

This field advertises the length in bytes of the command list (Command Class and Command fields).

Command Class and Command

CC:0059.01.06.13.001

Command Classes MAY be normal or extended Command Classes. Normal Command Classes are represented as one byte while extended Command Classes are represented as two bytes. Thus, a command list entry (command class + command) is either 2 or 3 bytes long.

The receiving node must parse individual command list entries to determine if the individual Command Class is a normal or an extended Command Class.

The first byte of normal command classes is in the range 0x20 – 0xEE, while the first byte of extended command classes is in the range 0xF1 – 0xFF.

CC:0059.01.06.11.001

A sending node MUST NOT add any command payload in this field.

CC:0059.01.06.11.002

Command Classes already containing destination NodeID such as Wake Up Command Class MUST NOT be listed in the Association Group Command List Report. This also means that the Wake Up Command Class MUST NOT be covered by the Lifeline association group.

4.7 Association Group Information (AGI) command class, version 2

The Association Group Information (AGI) Command Class allows a node to advertise the capabilities of each association group supported by a given application resource.

4.7.1 Compatibility considerations

The Association Group Information (AGI) Command Class, version 2 introduces the Profile category “Meter”.

The Association Group Information (AGI) Command Class, version 2 defines no new command fields or changes to the interpretation of Association Group Information (AGI) Command Class, version 1.

All sections and commands not described in this version remain unchanged from version 1.

4.7.2 The Association Group Information

A device MAY implement one or more association groups. If the device implements association groups, the device SHOULD provide an Association Group Information (AGI) table as described in in Table 3

4.7.2.1 Profile

The profile identifiers are referenced in Table 8. The profiles category “Meter” introduced in version 2 is described in 4.7.2.1.1.

4.7.2.1.1 Meter profiles

“Meter” profiles are intended for association groups of which commands are triggered by meter readings. As an example, a two-phase meter product could comprise three electric meters: one for each phase and one for the total consumption.

Meter profile identifiers are constructed by prepending the Meter Command Class identifier (referred to with AGI_PROFILE_METER) to a meter type defined in the Meter Command Class. The values in Table 8 only serve as examples to illustrate the construction of AGI Meter Profile identifiers. For the full list of available meter types, refer to the Meter Command Class.

In case of multiple logical functions in a device, the device SHOULD implement one Multi Channel End Point for each logical function; e.g. each meter instance. Table 7 gives an example of a Multi Channel Meter device with two normal and one aggregated meter functions.

Root Device:



Group identifier	Profile 2 bytes	Command Class & Command (list) N bytes	Group Name (UTF-8) M bytes
1	General:Lifeline	Meter Report Device Reset Locally Notification	Lifeline
2	Meter:Electric	Basic Set	On/Off control (Phase 1)
3	Meter:Electric	Basic Set	On/Off control (Phase 2)
4	Meter:Electric	Basic Set	On/Off control (Total consumption)

End Point 1:



Group identifier	Profile 2 bytes	Command Class & Command (list) N bytes	Group Name (UTF-8) M bytes
1	Meter:Electric	Meter Report	Phase 1 via Lifeline
2	Meter:Electric	Basic Set	On/Off control (Phase 1)

End Point 2:



Group identifier	Profile 2 bytes	Command Class & Command (list) N bytes	Group Name (UTF-8) M bytes
1	Meter:Electric	Meter Report	Phase 2 via Lifeline
2	Meter:Electric	Basic Set	On/Off control (Phase 2)

End Point 3 (aggregated):



Group identifier	Profile 2 bytes	Command Class & Command (list) N bytes	Group Name (UTF-8) M bytes
1	Meter:Electric	Meter Report	Total consumption via Lifeline
2	Meter:Electric	Basic Set	On/Off control (Total consumption)

Table 7, Example: AGI tables for three-function meter

The Profile identifier is Meter:Electric for all End Points but the Group Name allows a user to determine which End Point and which association group to use in order to configure the phase 2 meter to control an alarm lamp.

The Root Device in the above example advertises the Meter Report Command in the Lifeline group. Residing in a Multi Channel device, the Root Device does not actually implement any application functionality. The Meter Report command is a feature of the End Points which is advertised for backwards compatibility with legacy devices not supporting the Multi Channel Command Class. Likewise, the Root device advertises the Basic Set Command in association groups 2..4. This is also a

feature of the End Points. Refer to the Multi Channel Command Class for details on backwards compatibility.

End Points 1..3 in the above example advertise the Meter Report Command in association group 1. By advertising that zero NodeIDs are supported for association group 1, End Points indicate that this command is sent via the Root Device Lifeline group if a Multi Channel association is created for the Root Device Lifeline group.

4.8 Association Group Information (AGI) command class, version 3

The Association Group Information (AGI) Command Class allows a node to advertise the capabilities of each association group supported by a given application resource.

4.8.1 Compatibility considerations

The Association Group Information (AGI) Command Class, version 3 introduces the Profile category “Irrigation”.

The Association Group Information (AGI) Command Class, version 3 defines no new command fields or changes to the interpretation of Association Group Information (AGI) Command Class, version 1 and version 2.

All sections and commands not described in this version remain unchanged from version 2.

4.8.2 The Association Group Information

A node may implement one or more association groups. If the device implements association groups, the device SHOULD provide an Association Group Information (AGI) table as described in Table 3.

4.8.2.1 Profile

The profile categories are referenced in Table 8. The profiles category “Irrigation” introduced in version 3 is described in 4.8.2.1.1.

4.8.2.1.1 Irrigation Profiles

“Irrigation” profiles are intended for association groups of which commands are triggered by irrigation events. As an example, an irrigation control device may provide 8 channels that can each control an external resource such as a valve or a pump.

Table 8, AGI Profiles

Profile	Explanation	Profile identifier	
		MSB	LSB
General:NA (v1)	“Not Applicable” There is no specific class of events for this association group	AGI_PROFILE_ GENERAL = 0x00	AGI_ GENERAL_NA = 0x00
General:Lifeline (v1)	“Lifeline” This association group is intended for all events relevant for the Lifeline group.	AGI_PROFILE_ GENERAL = 0x00	AGI_ GENERAL_LIFELINE = 0x01
Control:Key01 (v1)	“Control Key 1” Members of this association group are controlled or receive reports in response to user input for key 1	AGI_PROFILE_ CONTROL = 0x20	AGI_ CONTROL_KEY01 = 0x01
Control:Key xx (v1)	...		
Control:Key32 (v1)	“Control Key 32” Members of this association group are controlled or receive reports in response to user input for key 32	AGI_PROFILE_ CONTROL = 0x20	AGI_ CONTROL_KEY32 = 0x20
Sensor: Air temperature (v1)	“Sensor, Air Temperature” Members of this association group are controlled or receive reports when the sensor value changes.	AGI_PROFILE_ SENSOR = 0x31	MULTILEVEL_SENSOR_TYPE_ TEMPERATURE = 0x01
Sensor: Humidity (v1)	“Sensor, Humidity” Members of this association group are controlled or receive reports when the sensor value changes.	AGI_PROFILE_ SENSOR = 0x31	MULTILEVEL_SENSOR_TYPE_ HUMIDITY = 0x05
...	(Only examples above. Sensor profiles are built based on the Sensor Type as described in 4.6.3.2.3)	AGI_PROFILE_ SENSOR = 0x31	...
Notification: Smoke Alarm (v1)	“Notification, Smoke Alarm” Members of this association group are controlled or receive reports when an event or state change is detected for the given Notification Type	AGI_PROFILE_ NOTIFICATION = 0x71	NOTIFICATION_TYPE_ SMOKE = 0x01
Notification: CO ₂ Alarm (v1)	“Notification, CO ₂ Alarm” Members of this association group are controlled or receive reports when an event or state change is detected for the given Notification Type	AGI_PROFILE_ NOTIFICATION = 0x71	NOTIFICATION_TYPE_ CO ₂ = 0x03
...	(Only examples above. Notification profiles are built based on the Notification Type as described in 4.6.3.2.4)	AGI_PROFILE_ NOTIFICATION = 0x71	...

Profile	Explanation	Profile identifier	
		MSB	LSB
Meter: Electric, kWh (v2)	"Meter, Electric" Members of this association group receive meter reports or are controlled when a metering event is detected.	AGI_PROFILE_ METER = 0x32	METER_TYPE_ELECTRIC = 0x01
Meter: Gas (v2)	"Meter, Gas" Members of this association group receive meter reports or are controlled when a metering event is detected.	AGI_PROFILE_ METER = 0x32	METER_TYPE_GAS = 0x02
Meter: Water (v2)	"Meter, Water" Members of this association group receive meter reports or are controlled when a metering event is detected.	AGI_PROFILE_ METER = 0x32	METER_TYPE_WATER = 0x03
...	(Only examples above. Meter Profiles are built based on the Meter Types as described in 4.7.2.1.1)	AGI_PROFILE_ METER = 0x32	...
Irrigation: Channel 01 (v3)	"Irrigation Channel 01" Member(s) of this association group are controlled by channel 1 of an irrigation control device	AGI_PROFILE_ IRRIGATION = 0x6B	AGI_ IRRIGATION_CHANNEL_01 = 0x01
Irrigation: Channel xx (v3)	...		
Irrigation: Channel 32 (v3)	"Irrigation Channel 32" Member(s) of this association group are controlled by channel 32 of an irrigation control device	AGI_PROFILE_ IRRIGATION = 0x6B	AGI_ IRRIGATION_CHANNEL_32 = 0x20

4.9 Battery Command Class, version 1

The Battery Command Class is used to request and report battery levels for a given device.

4.9.1 Battery Level Get Command

The Battery Level Get Command is used to request the level of a battery.

CC:0080.01.02.11.001 The Battery Level Report Command **MUST** be returned in response to this command.

CC:0080.01.02.11.002 This command **MUST NOT** be issued via multicast addressing.
A receiving node **MUST NOT** return a response if this command is received via multicast addressing.
The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_BATTERY							
Command = BATTERY_GET							

4.9.2 Battery Level Report Command

This command is used to report the battery level of a battery operated device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_BATTERY							
Command = BATTERY_REPORT							
Battery Level							

Battery Level (8 bits)

This field is used to report the percentage indicating the battery level.

CC:0080.01.03.11.001 This field **MUST** be in the range 0x00..0x64 or set to 0xFF.

CC:0080.01.03.11.002 Values in the range 0x00..0x64 **MUST** indicate the battery percentage level from 0 to 100%
CC:0080.01.03.11.003 The value 0xFF **MUST** indicate a low-battery warning.

4.10 Device Reset Locally Command Class, version 1

The Device Reset Locally Command Class is used to notify central controllers that a Z-Wave device is resetting its network specific parameters.

CC:005A.01.00.12.001 Any device SHOULD implement this Command Class if it can be reset to factory default.

CC:005A.01.00.11.001 Z-Wave Plus compliant devices MUST implement this command class if they can be reset to factory default.

4.10.1 Device Reset Locally Notification Command

CC:005A.01.01.11.001 The Device Reset Locally Notification Command is used to advertise that the device will be reset to default. The reset operation MUST reset protocol data (HomeID, NodeID, etc.) as well as all application specific data to factory default values. In case a Lifeline destination is configured in Association Group #1, the device MUST send a Device Reset Locally Notification Command to the Lifeline destination.

CC:005A.01.01.11.002 The Device Reset Locally Notification Command MUST be issued by the Root Device. If Multi Channel encapsulation is used, the source End Point MUST be set to 0.

CC:005A.01.01.12.001 A controller device receiving the Device Reset Locally Notification Command SHOULD consider the sending node to be a failing node and accordingly perform relevant maintenance operations like removing failing nodes, removing associations to failing nodes, etc.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_DEVICE_RESET_LOCALLY							
Command = DEVICE_RESET_LOCALLY_NOTIFICATION							

4.11 Firmware Update Meta Data Command Class, version 1 [DEPRECATED]

THIS VERSION HAS BEEN DEPRECATED

CC:007A.01.00.13.001
CC:007A.01.00.12.001

A device MAY implement support for this version of the command class, but it is RECOMMENDED that new implementations implement the Firmware Update Meta Data Command Class, version 3.

If implementing support for this version of the command class, it is RECOMMENDED that support for Firmware Update Meta Data Command Class, version 3 is also implemented.

The Firmware Update Meta Data Command Class may be used to transfer a firmware image to a Z-Wave device.

CC:007A.01.00.11.001

A device which supports the Firmware Update CC MUST support the Version CC and the Manufacturer Specific CC to enable other devices to select the correct firmware for a specific target. The Version Command Class may be used to verify that the intended firmware version is installed.

CC:007A.01.00.11.002

Devices implementing the Firmware Update CC MUST support a data rate of 40kbit/s or more.

CC:007A.01.00.12.002
CC:007A.01.00.12.003

A checksum SHOULD be appended to the firmware image to ensure the integrity of the image. A receiving device SHOULD verify the integrity of the received firmware image by matching the received firmware image checksum and the firmware image checksum that is indicated by the Firmware Meta Data Report when the firmware update is initiated.

CC:007A.01.00.12.004

It is RECOMMENDED that firmware update is enabled by out-of-band authentication (e.g. physical activation of a pushbutton). If out-of-band authentication is implemented, the device to receive a firmware update must be armed for firmware update before the Firmware Update Meta Data Request Get Command is issued.

4.11.1 Firmware Meta Data Get Command

The Firmware Meta Data Get Command is used to request information on the current firmware in the device.

CC:007A.01.01.11.001

The Firmware Meta Data Report Command MUST be returned in response to this command.

CC:007A.01.01.11.002
CC:007A.01.01.11.003

This command MUST NOT be issued via multicast addressing.

A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_MD_GET							

4.11.2 Firmware Meta Data Report Command

The Firmware Meta Data Report Command is used to advertise the status of the current firmware in the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_MD_REPORT							
Manufacturer ID 1							
Manufacturer ID 2							
Firmware ID 1							
Firmware ID 2							
Checksum 1							
Checksum 2							

Manufacturer ID (16 bits)

The Manufacturer ID is a unique ID identifying the manufacturer of the device.

Manufacturer identifiers can be found in [12].

CC:007A.01.02.11.001 The first byte **MUST** be the most significant byte.

Firmware ID (16 bits)

CC:007A.01.02.12.001 A manufacturer **SHOULD** assign a unique Firmware ID to each existing product variant. A product variant may be a particular hardware version for a particular world region. When combined with the Manufacturer ID, the Firmware ID is a unique identification of a firmware image that is guaranteed to work with a particular product among all Z-Wave enabled products in the world. If the product has no Firmware ID, the value 0x0000 **MUST** be returned.

CC:007A.01.02.11.002 The first byte **MUST** be the most significant byte.

A user may request Manufacturer Specific information to get more detailed information on the product.

CC:007A.01.02.12.002 A controlling device **SHOULD** match the advertised firmware ID for the existing firmware image to the firmware ID provided for a new image. If the firmware IDs do not match, the controlling device **SHOULD** abort the firmware update operation.

Checksum (16 bits)

CC:007A.01.02.13.001 The checksum field is used to report a checksum value of the firmware image currently running in the Z-Wave chip. As an alternative, a value of zero MAY be returned.

CC:007A.01.02.11.003 The first byte MUST be the most significant byte.

CC:007A.01.02.12.003 It is RECOMMENDED to use a checksum algorithm that implements a CRC-CCITT polynomial using initialization value equal to 0x1D0F and 0x1021 (normal representation).

4.11.3 Firmware Update Meta Data Request Get Command

The Firmware Update Meta Data Request Get Command is used to request that a firmware update is initiated.

CC:007A.01.03.11.001 The Firmware Update Meta Data Request Report Command MUST be returned in response to this command.

CC:007A.01.03.11.002 The firmware update MUST NOT be initiated if the Manufacturer ID and the Firmware ID do not match the actual firmware image values.

CC:007A.01.03.11.003 The firmware update MUST be aborted if the checksum does not match the calculated checksum after the firmware image has been transferred.

CC:007A.01.03.12.001 It is RECOMMENDED that firmware update is initiated by out-of-band authentication (e.g. physical activation of a pushbutton).

CC:007A.01.03.11.004 This command MUST NOT be issued via multicast addressing.

CC:007A.01.03.11.005 A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_REQUEST_GET							
Manufacturer ID 1							
Manufacturer ID 2							
Firmware ID 1							
Firmware ID 2							
Checksum 1							
Checksum 2							

Manufacturer ID (16 bits)

The Manufacturer ID is a unique ID identifying the manufacturer of the device.

CC:007A.01.03.11.006 Manufacturer identifiers can be found in [12]. The first byte **MUST** be the most significant byte.

Firmware ID (16 bits)

CC:007A.01.03.12.002 A manufacturer **SHOULD** assign a unique Firmware ID to each existing product variant. A product variant may be a particular hardware version for a particular world region. When combined with the Manufacturer ID, the Firmware ID is a unique identification of a firmware image that is guaranteed to work with a particular product among all Z-Wave enabled products in the world.

CC:007A.01.03.11.007 A receiving application **MUST** check that the Manufacturer ID and Firmware ID fields match the Manufacturer ID and Firmware ID values of the current firmware. While it is **NOT RECOMMENDED**, one MAY design a the product that has no Firmware ID.

CC:007A.01.03.11.008 A sending node **MUST** specify the Firmware ID value 0x0000 to indicate an absent Firmware ID.

CC:007A.01.03.11.009 A receiving node **MUST** interpret the value 0x0000 as “No Firmware ID available”.

CC:007A.01.03.11.00A The first byte **MUST** be the most significant byte.

Checksum (16 bits)

CC:007A.01.03.11.00B The checksum field **MUST** carry the checksum of the firmware image about to be transferred.

CC:007A.01.03.11.00C A receiving node **MUST** match this value and the checksum calculated from the received firmware image.

CC:007A.01.03.12.004 It is **RECOMMENDED** to use a checksum algorithm that implements a CRC-CCITT polynomial using initialization value equal to 0x1D0F and 0x1021 (normal representation). Refer to appendices in [2].

4.11.4 Firmware Update Meta Data Request Report Command

This command is used to advertise if the firmware update will be initiated.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_REQUEST_REPORT							
Status							

Status (8 bits)

CC:007A.01.04.11.001 This field **MUST** comply with Table 9.

4.11.5 Firmware Update Meta Data Get Command

The Firmware Update Meta Data Get Command is used to request one or more Firmware Update Meta Data Report Commands.

CC:007A.01.05.11.001 The Firmware Update Meta Data Report Command **MUST** be returned in response to this command.

CC:007A.01.05.13.001 The transmission of the next Firmware Update Meta Data Get Command **MAY** be delayed if time is required to store the most recent firmware fragment in temporary non-volatile memory. A node **MAY** request multiple Firmware Update Meta Data Report Commands to improve throughput.

CC:007A.01.05.11.002 This command **MUST NOT** be issued via multicast addressing.
 CC:007A.01.05.11.003 A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

Two exceptions may apply: A noise burst may interfere the transmission or the data source may stop returning Firmware Update Meta Data Report Commands.

CC:007A.01.05.12.001 To accommodate for bursts of RF noise, a device receiving data **SHOULD** repeatedly retransmit the same Firmware Update Meta Data Get Command every 10 seconds in case no Firmware Update Meta Data Report Commands are received.

CC:007A.01.05.12.002 A device receiving data **SHOULD** stop retransmitting Firmware Update Meta Data Get commands 2 minutes after the last successful reception of a Firmware Update Meta Data Report Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_GET							
Number of Reports							
Res	Report number 1						
Report number 2							

Res

CC:007A.01.05.11.004 This field **MUST** be set to 0 by a sending node and **MUST** be ignored by a receiving node.

Number of Reports (8 bits)

Number of Firmware Update Meta Data Report Commands to be received in response to this Firmware Update Meta Data Get Command.

Report number 1 .. 2 (15 bits)

CC:007A.01.05.11.005 The Report number field indicates the Firmware Update Meta Data Report Command to be requested. The report number values **MUST** be a sequence starting from 1. The first byte (Report number 1) is the most significant byte.

4.11.6 Firmware Update Meta Data Report Command

The Firmware Update Meta Data Report Command is used to transfer a firmware image fragment.

If sending more than a single Firmware Update Meta Data Report Command at a time, a node **MUST** apply a delay between each transmitted command. The minimum required time delay and number of frames before a delay must be inserted depends on the actual bit rate.

- 40 kbit/s: At least 35 ms if sending more than 1 frame back-to-back
- 100 kbit/s: At least 15 ms if sending more than 2 frames back-to-back

If needed, a controlling node **SHOULD** abort an ongoing transfer by responding to a Firmware Update Meta Data Get Command with a Firmware Update Meta Data Report Command with the Last bit enabled and the Data fields intentionally corrupted.

This will invalidate the calculated firmware checksum, which should eventually cause the receiving device to return a Firmware Update Meta Data Status Report Command with the status code <checksum error>.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_REPORT							
Last	Report number 1						
Report number 2							
Data 1							
...							
Data N							

Last (1 bit)

The Last flag indicates if the requested Firmware Update Meta Data Report Command carries the last firmware image fragment.

The flag **MUST** be set to '1' if this is the last fragment. Otherwise the flag **MUST** be '0'.

Report number (15 bits)

The Report number field indicates the sequence number of the contained firmware fragment. The first firmware fragment **MUST** be identified by the Report number value 1. The sequence number of each following firmware fragment **MUST** be incremented.

Report number 1 is the most significant byte.

The report number may be used to calculate the offset of the data by the formula:

Offset = (Report number – 1) x Number of Data fields (N)

Except for the last frame, each Report **MUST** carry the same number of Data bytes as the first fragment. The last frame **MAY** carry a shorter Data field.

Data (N bytes)

The Data field is used to carry one firmware image fragment. Except for the last frame, each Firmware Update Meta Data Report Command MUST carry the same number of Data bytes as the first fragment. The last frame MAY carry a shorter Data field.

CC:007A.01.06.11.005

CC:007A.01.06.13.002

CC:007A.01.06.11.006

A sending device MUST use a fragment size which matches the actual number of available Data bytes. The number of available Data bytes depends on the actual bit rate, the use of security encapsulation as well as the presence of Checksum bytes in the Firmware Update Meta Data Report.

CC:007A.01.06.11.007

A receiving device MUST determine the number of Data bytes from the length of the first received frame.

4.11.7 Firmware Update Meta Data Status Report Command

This command is used to advertise the firmware update status.

CC:007A.01.07.11.001

The command MUST be issued when the firmware update is completed or aborted by the device receiving the firmware.

CC:007A.01.07.11.002

A node MUST NOT issue Firmware Update Meta Data Get Command after receiving a Firmware Update Meta Data Status Report.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_STATUS_REPORT							
Status							

Status (8 bits)

CC:007A.01.07.11.003

This field MUST comply with Table 10.

4.11.8 Examples

Figure 3 shows a controlling node requesting the Manufacturer ID and Firmware ID of another node by issuing the Firmware Meta Data Get Command.

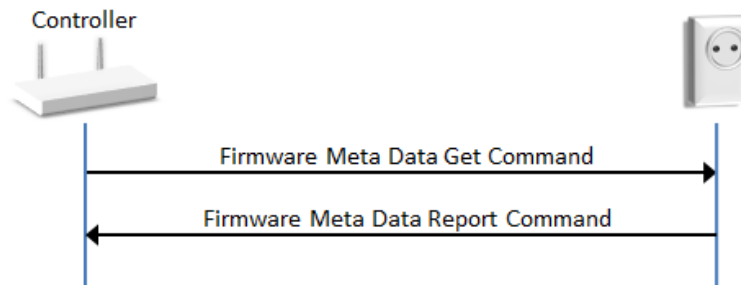


Figure 3, Requesting Manufacturer ID and Firmware ID of a node

Figure 4 outlines the actual firmware update message flow. Prior to the firmware update, the supporting node may receive an out-of-band authentication (e.g. physical activation of a pushbutton). The controller sends a Firmware Meta Data Request Get Command to initiate the downloading a new firmware image. The supporting node returns a Firmware Meta Data Request Report Command report to confirm the update request and the supporting node begins pulling firmware image fragments from the controller. Finally a Firmware Update Meta Data Status Report is returned to the controller to indicate the success (or failure) of the update process.

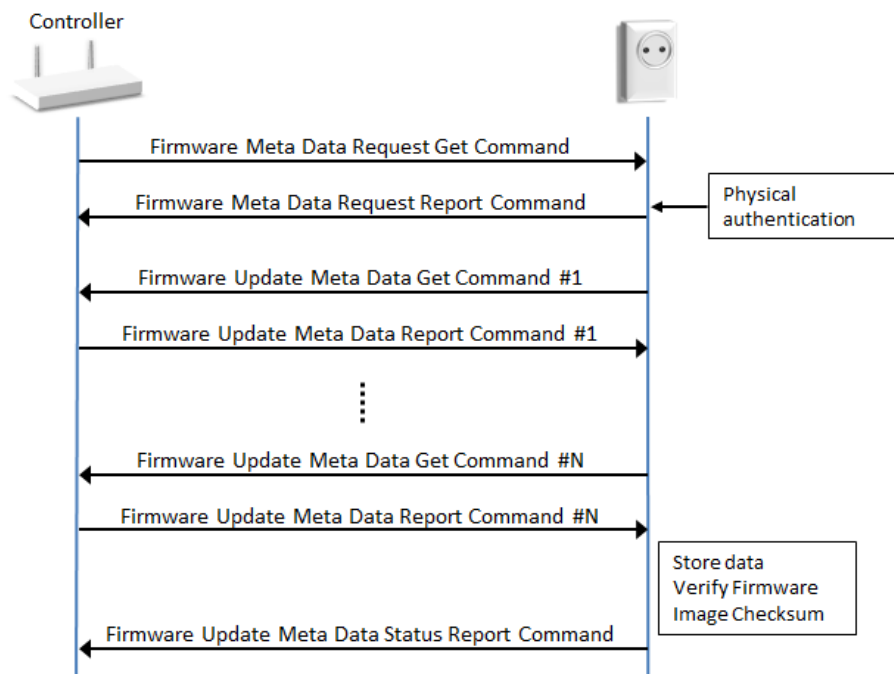


Figure 4, Transferring a firmware image to a device

4.12 Firmware Update Meta Data Command Class, version 2

Firmware Update Meta Data Command Class, version 2 updates the Firmware Update Meta Data Report Command.

While support for version 1 has been deprecated, a controlling device implementing Firmware Update Meta Data Command Class, version 2 **MUST** also be able to control Firmware Update Meta Data Command Class, version 1 based devices.

4.12.1 Compatibility considerations

A node supporting Firmware Update Meta Data Command Class, version 2 **MUST** also support Firmware Update Meta Data Command Class, version 1.

All commands and fields not described in this version remain unchanged from version 1.

4.12.2 Interoperability considerations

4.12.2.1 Interoperability with v1 devices

Version 2 of the Firmware Update Meta Data Report command introduces a 16-bit Checksum field to follow the variable-length Data field. No fields follow the variable-length Data field in Version 1 of the Firmware Update Meta Data Report command.

At the same time, the specified method for determining the length of the variable-length Data field is that the length must be calculated from the length of the received frame.

The unintended consequence is that a version 1 implementation receiving a Firmware Update Meta Data Report v2 command will consider the 16-bit Checksum field to be the last two bytes of the Data field. Similarly, a version 2 implementation receiving a Firmware Update Meta Data Report v1 command will consider the last two bytes of the Data field to be the 16-bit Checksum field. In either case, the transfer will fail.

Therefore, a controlling device implementing the Firmware Update Meta Data Command Class, version 2 **MUST** do the following when initiating the transfer of a firmware image:

1. Request the version of the Firmware Update Meta Data Command Class from the target device
2. Use the version of the Firmware Update Meta Data Report implemented by the target device
3. Determine the number of Data bytes that can fit into the Firmware Update Meta Data Report command so that the complete command can still fit into the payload field of the transport frame. The number of available payload bytes in the frame depends on the actual bit rate, the use of security encapsulation as well as the presence of Checksum bytes in the Firmware Update Meta Data Report.

The controller **SHOULD** accept any binary image format and transfer that without any modifications. The image may be encrypted or carry checksums.

4.12.2.2 Checksum calculation

Version 1 of the Firmware Update Meta Data Command Class introduced a 16-bit Checksum field used to verify firmware image integrity. It has been a recommendation to use the CRC-CCITT polynomial for calculating checksums. Nodes having implemented another method for calculating the checksum will find a non-matching checksum.

From Version 5 onwards, the checksum calculation method is mandatory. For more details about the checksum calculation, refer to Appendix B

4.12.3 Firmware Update Meta Data Report Command

The Firmware Update Meta Data Report command is used to transfer a firmware image fragment.

CC:007A.02.06.11.001

If sending more than a single Firmware Update Meta Data Report Command at a time, a node **MUST** apply a delay between each transmitted command. The minimum required time delay and number of frames before a delay must be inserted depends on the actual bit rate.

- 40 kbit/s: At least 35 ms if sending more than 1 frame back-to-back
- 100 kbit/s: At least 15 ms if sending more than 2 frames back-to-back

CC:007A.02.06.12.001

If needed, a controlling node **SHOULD** abort an ongoing transfer by responding to a Firmware Update Meta Data Get Command with a Firmware Update Meta Data Report Command with the Last bit enabled and the Data fields intentionally corrupted while the Checksum field of the Firmware Update Meta Data Report Command is valid, i.e. calculated over the corrupted Data fields. This will invalidate the calculated firmware checksum, which should eventually cause the receiving device to return a Firmware Update Meta Data Status Report Command with the status code <checksum error>.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_REPORT							
Last	Report number 1						
Report number 2							
Data 1							
...							
Data N							
Checksum 1							
Checksum 2							

All fields not described below remain unchanged from version 1.

Checksum (16 bits)

- CC:007A.02.06.11.002 The checksum field **MUST** be used to ensure the consistency of the entire command; including the command class and command identifiers.
- CC:007A.02.06.12.002 It is **RECOMMENDED** to use a checksum algorithm that implements the CRC-CCITT polynomial using initialization value equal to 0x1D0F and 0x1021 (normal representation).

The Checksum field is known to cause compatibility issues with version 1 devices. Refer to 4.12.1.

4.13 Firmware Update Meta Data Command Class, version 3

The Firmware Update Meta Data Command Class, version 3 allows a Z-Wave enabled device to receive one out of several firmware images.

As an example, one may construct a product comprising a Z-Wave chip and a secondary host processor that maintains a security certificate. With the capability to handle multiple firmware images enables the Z-Wave chip, the host processor and the security certificate may all be updated via individual firmware IDs.

4.13.1 Compatibility considerations

CC:007A.03.00.21.001 A device implementing Firmware Update Meta Data Command Class, version 3 MUST also implement Firmware Update Meta Data Command Class, version 2.

CC:007A.03.00.21.002 While support for version 1 has been deprecated, a controlling device implementing Firmware Update Meta Data Command Class, version 2 MUST also be able to control Firmware Update Meta Data Command Class, version 1 based devices.

All commands and fields not described in this version remain unchanged from version 2.

4.13.1.1 New fields and values

The Firmware Update Command Class, Version 3, introduces WaitTime parameter and a new Status code 0xFE for the Firmware Update Meta Data Status Report Command.

CC:007A.03.00.21.003 The status code 0xFE is intended for confirming the successful transfer of images which do not necessitate a restart, e.g. security certificates. If this code was returned to a controller implementing a previous version, the controller would report that an error had occurred. Therefore, a supporting node MUST NOT return this code after having updated the firmware image for the Z-Wave chip image, i.e. the Firmware ID 0 target.

4.13.2 Interoperability considerations

CC:007A.03.00.32.001 Unintended modification of a firmware image SHOULD be prevented. It is RECOMMENDED that firmware update is enabled by out-of-band authentication (e.g. physical activation of a pushbutton).

CC:007A.03.00.31.001 A checksum MUST be appended to the Firmware 0 image to ensure the integrity of the image. A receiving device SHOULD verify the integrity of the received Firmware 0 image by matching the received
CC:007A.03.00.32.002 Firmware 0 image checksum and the Firmware 0 image checksum that is indicated by the Firmware Meta Data Report when the firmware update is initiated.

CC:007A.03.00.32.003 Any image transferred via the Firmware Update Meta Data Command Class SHOULD include a fingerprint value for validation of the integrity of the entire image after the transfer. The image and the
CC:007A.03.00.31.002 fingerprint value MUST be packed in one entity which can be stored in one file and transferred as one entity.

CC:007A.03.00.31.003 A manufacturer MUST assign a unique Firmware ID to each existing product variant. A product variant may be a particular hardware version for a particular world region. The combination of Manufacturer ID and Firmware ID provides unique identification of a firmware image that is guaranteed to work with a particular component of a particular product.

4.13.2.1 Interoperability with v1 devices

Version 2 of the Firmware Update Meta Data Report command introduces a 16-bit Checksum field to follow the variable-length Data field. No fields follow the variable-length Data field in Version 1 of the Firmware Update Meta Data Report command. Version 3 of the Firmware Update Meta Data Report command uses the v2 format.

At the same time, the specified method for determining the length of the variable-length Data field is that the length must be calculated from the length of the received frame.

The unintended consequence is that a version 1 implementation receiving a Firmware Update Meta Data Report v2 command will consider the 16-bit Checksum field to be the last two bytes of the Data field. Similarly, a version 2 implementation receiving a Firmware Update Meta Data Report v1 command will consider the last two bytes of the Data field to be the 16-bit Checksum field. In either case, the transfer will fail.

CC:007A.03.00.31.004

Therefore, a node controlling the Firmware Update Meta Data Command Class, version 3 **MUST** do the following when initiating the transfer of a firmware image:

1. Request the version of the Firmware Update Meta Data Command Class from the target device
2. Use the version of the Firmware Update Meta Data Report implemented by the target device
3. Determine the number of Data bytes that can fit into the Firmware Update Meta Data Report command so that the complete command can still fit into the payload field of the transport frame. The number of available payload bytes in the frame depends on the actual bit rate, the use of security encapsulation as well as the presence of Checksum bytes in the Firmware Update Meta Data Report

4.13.2.2 Checksum calculation

Version 1 of the Firmware Update Meta Data Command Class introduced a 16-bit Checksum field used to verify firmware image integrity. It has been a recommendation to use the CRC-CCITT polynomial for calculating checksums. Nodes having implemented another method for calculating the checksum will find a non-matching checksum.

From Version 5 onwards, the checksum calculation method is mandatory.

For more details about the checksum calculation, refer to Appendix B

4.13.3 Firmware Meta Data Report Command

This command is used to advertise the status of the current firmware in the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_MD_REPORT							
Manufacturer ID 1							
Manufacturer ID 2							
Firmware 0 ID 1							
Firmware 0 ID 2							
Firmware 0 Checksum 1							
Firmware 0 Checksum 2							
Firmware Upgradable							
Number of Firmware Targets							
Max Fragment Size 1							
Max Fragment Size 2							
Firmware 1 ID 1							
Firmware 1 ID 2							
...							
Firmware N ID 1							
Firmware N ID 2							

Fields not described below remain unchanged from version 2

Firmware 0 ID (16 bits)

The Firmware 0 ID field is dedicated to target 0, i.e. the Z-Wave chip.

CC:007A.03.02.11.001 A manufacturer **MUST** assign a unique Firmware ID to each existing product variant. A product variant may be a particular hardware version for a particular world region. The combination of Manufacturer ID and Firmware ID provides unique identification of a firmware image that is guaranteed to work with a particular component of a particular product.

CC:007A.03.02.11.002 The first byte **MUST** be the most significant byte.

A user may request Manufacturer Specific information to get more detailed information on the product.

CC:007A.03.02.12.001 A controlling node **SHOULD** match the advertised firmware ID for the existing firmware image to the firmware ID provided for a new image. If the firmware IDs do not match, the controlling device **SHOULD** abort the firmware update operation.

Firmware 0 Checksum (16 bits)

CC:007A.03.02.11.003 The checksum field **MUST** be used to ensure consistency of the Firmware 0 image currently in the device.

CC:007A.03.02.11.004 The first byte **MUST** be the most significant byte.

CC:007A.03.02.12.002 It is **RECOMMENDED** to use a checksum algorithm that implements a CRC-CCITT polynomial using initialization value equal to 0x1D0F and 0x1021 (normal representation).

Firmware Upgradable (8 bits)

This field defines whether the Z-Wave chip is firmware upgradable.

CC:007A.03.02.11.005 The value 0x00 **MUST** indicate that the firmware 0 image is not upgradable.
The value 0xFF **MUST** indicate that the firmware 0 image is upgradable.

Number of Firmware Targets (8 bits)

CC:007A.03.02.11.006 The Number of Firmware Targets field **MUST** report the number of firmware IDs following this field.

CC:007A.03.02.11.007 The Firmware 0 ID field is not included. The field **MUST** be zero if the device only implements a Firmware 0 target, i.e. the Z-Wave chip.

Max Fragment Size (16 bits)

CC:007A.03.02.11.008 The Max Fragment Size field **MUST** report the maximum number of Data bytes that a device is able to receive at a time. A sending node **MAY** send shorter fragments. The fragment size actually used is indicated in the Firmware Update Meta Data Request Get Command and confirmed in the Firmware Update Meta Data Request Report Command.

CC:007A.03.02.11.009 The Max Fragment Size may be longer than the supported frame length of Z-Wave if the image is to be transferred over e.g. IP. If the image is to be transferred over Z-Wave, the sending node **MUST** use a fragment size which matches the actual number of available Data bytes. The number of available Data bytes depends on the actual bit rate, the use of security encapsulation as well as the presence of Checksum bytes in the Firmware Update Meta Data Report.

Firmware n ID (16 bits)

CC:007A.03.02.11.00A The Firmware 1 ID field indicates the Firmware ID that **MUST** be used for target 1. The Firmware 2 ID field represents target 2. And so on.

CC:007A.03.02.11.00B The first byte **MUST** be the most significant byte.

CC:007A.03.02.12.003 A controlling device **SHOULD** match the advertised firmware ID for the existing firmware image to the firmware ID provided for a new image. If the firmware IDs do not match, the controlling device **SHOULD** abort the firmware update operation.

4.13.4 Firmware Update Meta Data Request Get Command

The Firmware Update Meta Data Request Get Command is used to request that a firmware update is initiated by the node receiving this command.

CC:007A.03.03.11.001 The Firmware Update Meta Data Request Report Command **MUST** be returned in response to this command.

- CC:007A.03.03.11.002 This command **MUST NOT** be issued via multicast addressing.
- CC:007A.03.03.11.003 A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.
- CC:007A.03.03.11.004 The firmware update **MUST NOT** be initiated if the Manufacturer ID and the Firmware ID do not match the actual firmware image values for the specified Firmware Target.
- CC:007A.03.03.11.005 The firmware update **MUST** be aborted if the checksum does not match the calculated checksum after the firmware image has been transferred.
- CC:007A.03.03.12.001 It is **RECOMMENDED** that firmware update is enabled by out-of-band authentication (e.g. physical activation of a pushbutton) prior to the transmission of this command.
- CC:007A.03.03.12.002 Any image transferred via the Firmware Update Meta Data Command Class **SHOULD** include a fingerprint value to enable the validation of the integrity of the entire image after the transfer. The image and the fingerprint value **MUST** be packed in one entity which can be stored in one file and transferred as one entity.
- CC:007A.03.03.11.006

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_REQUEST_GET							
Manufacturer ID 1							
Manufacturer ID 2							
Firmware ID 1							
Firmware ID 2							
Checksum 1							
Checksum 2							
Firmware Target							
Fragment Size 1							
Fragment Size 2							

All fields not described below remain unchanged from version 2.

Firmware ID (16 bits)

- CC:007A.03.03.11.007 The Firmware ID **MUST** match the specified target of the actual device.
- CC:007A.03.03.11.008 The firmware update **MUST NOT** be initiated if the Firmware ID does not match the specified target of the actual device.
- CC:007A.03.03.11.009 The first byte **MUST** be the most significant byte.

Firmware Target (8 bits)

CC:007A.03.03.11.00A

This field **MUST** be used to identify the firmware image to be updated. The firmware images **MUST** be identified as follows:

Firmware Target	Description
0x00	Firmware image targeted for the Z-Wave chip.
0x01	Firmware image intended for target 1 defined by the manufacturer.
...	
0xFF	Firmware image intended for target 255 defined by the manufacturer.

Fragment Size (16 bits)

CC:007A.03.03.11.00B

The Fragment Size field **MUST** report the fragment size that is to be used for firmware fragments. A receiving device **MUST** use this fragment size for the firmware update. The fragment size is not exchanged during the actual firmware update.

CC:007A.03.03.11.00C

The Fragment Size **MUST NOT** exceed the Max Fragment Size value of the Firmware Meta Data Report.

CC:007A.03.03.11.00D

A version 1 Firmware Update Meta Data Request Get Command does not carry a Fragment Size field. A receiving node **MUST** determine the number of Data bytes from the length of the first received frame, the use of security encapsulation as well as the presence of Checksum bytes in the Firmware Update Meta Data Report.

4.13.5 Firmware Update Meta Data Request Report Command

This command is used to advertise if the firmware update will be initiated.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_REQUEST_REPORT							
Status							

Status (8 bits)

CC:007A.03.04.11.001

This field **MUST** comply with Table 9.

4.13.6 Firmware Update Meta Data Get Command

The Firmware Update Meta Data Get Command is used to request one or more Firmware Update Meta Data Report Commands.

- CC:007A.03.05.11.001 One or more Firmware Update Meta Data Report Commands **MUST** be returned in response to this command.
- CC:007A.03.05.11.002 This command **MUST NOT** be issued via multicast addressing.
- CC:007A.03.05.11.003 A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.
- CC:007A.03.05.11.004 A node **MUST** verify the Manufacturer ID and Firmware ID fields received in a Firmware Update Meta Data Request Get Command before sending any Firmware Update Meta Data Get Commands.
- CC:007A.03.05.11.005 A controlling node receiving a Firmware Update Meta Data Get Command **MUST** return fragments of the firmware image that was previously identified by the Manufacturer ID and Firmware ID fields received in a Firmware Update Meta Data Request Get Command.
- CC:007A.03.05.13.001 The transmission of the next Firmware Update Meta Data Get Command **MAY** be delayed if time is required to store the most recent firmware fragment in temporary non-volatile memory. A node **MAY** request multiple Firmware Update Meta Data Report Commands to improve throughput.
- CC:007A.03.05.13.002
- CC:007A.03.05.13.003 RAM may be a limited resource. A node **MAY** adjust the size of incoming Firmware Update Meta Data Report Commands to the available first level RAM receive buffer by reporting the desired Max Fragment Size when returning a Firmware Meta Data Report Command in response to a Firmware Update Meta Data Get Command.
- Two exceptions may apply: A noise burst may interfere the transmission or the data source may stop returning Firmware Update Meta Data Report Commands.
- CC:007A.03.05.12.001 To accommodate for bursts of RF noise, a device receiving data **SHOULD** repeatedly retransmit the same Firmware Update Meta Data Get Command every 10 seconds in case no Firmware Update Meta Data Report Commands are received.
- CC:007A.03.05.12.002 A device receiving data **SHOULD** stop retransmitting Firmware Update Meta Data Get commands 2 minutes after the last successful reception of a Firmware Update Meta Data Report Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_GET							
Number of Reports							
Res	Report Number 1						
Report Number 2							

Res

- CC:007A.03.05.11.006 This field **MUST** be set to 0 by a sending node and **MUST** be ignored by a receiving node.

Number of Reports (8 bits)

Number of Firmware Update Meta Data Report Commands to be received in response to a single Firmware Update Meta Data Get Command.

CC:007A.03.05.11.007 The requesting node **MUST** keep track of missing Firmware Update Meta Data Report Commands.

Report Number (15 bits)

CC:007A.03.05.11.008 The Report number field indicates the sequence number of the requested firmware fragment. The first firmware fragment **MUST** be identified by the Report Number value 1.

CC:007A.03.05.11.009 Report number 1 **MUST** be the most significant byte.

CC:007A.03.05.11.00A If the Number of Reports field is larger than 1, the Report Number **MUST** identify the first firmware fragment in the requested sequence of firmware fragments.

4.13.7 Firmware Update Meta Data Report Command

The Firmware Update Meta Data Report command is used to transfer a firmware image fragment.

CC:007A.03.06.11.001 If sending more than a single Firmware Update Meta Data Report Command at a time, a node **MUST** apply a delay between each transmitted command. The minimum required time delay and number of frames before a delay must be inserted depends on the actual bit rate.

- 40 kbit/s: At least 35 ms if sending more than 1 frame back-to-back
- 100 kbit/s: At least 15 ms if sending more than 2 frames back-to-back

After receiving all Firmware Update Meta Data Report Commands required for a complete firmware image, a receiving node **MUST** store the firmware image in the Firmware Target specified in a previously received Firmware Update Meta Data Request Get Command.

CC:007A.03.06.12.001 If needed, a controlling node **SHOULD** abort an ongoing transfer by responding to a Firmware Update Meta Data Get Command with a Firmware Update Meta Data Report Command with the Last bit enabled and the Data fields intentionally corrupted while the Checksum field of the Firmware Update Meta Data Report Command is valid, i.e. calculated over the corrupted Data fields. This will invalidate the calculated firmware checksum, which should eventually cause the receiving device to return a Firmware Update Meta Data Status Report Command with the status code <checksum error>.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_REPORT							
Last	Report number 1						
Report number 2							
Data 1							
...							
Data N							
Checksum 1							
Checksum 2							

All fields not described below remain unchanged from version 2.

Last (1 bit)

This field indicates if the Firmware Update Meta Data Report Command is the last one.

CC:007A.03.06.11.002 The value 1 **MUST** indicate that this is the last report. Otherwise the field **MUST** be set to 0.

CC:007A.03.06.11.003 On the reception of the last image fragment, the receiving node **MUST** verify that the transferred image does match the indicated Manufacturer ID, Firmware ID and Firmware Target values.

Checksum (16 bits)

CC:007A.03.06.11.004 The checksum field **MUST** be used to ensure the consistency of the entire command; including the command class and command identifiers.

CC:007A.03.06.12.002 It is **RECOMMENDED** to use a checksum algorithm that implements the CRC-CCITT polynomial using initialization value equal to 0x1D0F and 0x1021 (normal representation).

CC:007A.03.06.11.005 The checksum algorithm used for the Firmware Update Meta Data Command Class, version 3 **MUST** be the same as the algorithm used for the Firmware Update Meta Data Command Class, version 2 in the actual product.

The Checksum field is known to cause compatibility issues with version 1 devices. Refer to 4.12.1.

4.13.8 Firmware Update Meta Data Status Report Command

This command is used to advertise the firmware update status.

CC:007A.03.07.11.001 The command **MUST** be issued when the firmware update is completed or aborted by the device receiving the firmware.

CC:007A.03.07.11.002 A node **MUST NOT** issue the Firmware Update Meta Data Get Command after receiving a Firmware Update Meta Data Status Report.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_STATUS_REPORT							
Status							
WaitTime MSB							
WaitTime LSB							

Status (8 bits)

CC:007A.03.07.11.003 This field **MUST** comply with Table 10.

CC:007A.03.07.13.001 The status code 0xFE **MAY** be used for confirming the successful transfer of images which do not necessitate a restart, e.g. security certificates.

CC:007A.03.07.11.004 The status code 0xFE **MUST NOT** be advertised after the transfer of an image for the Firmware ID 0 target (the "Z-Wave chip" image).

CC:007A.03.07.11.005 Controlling nodes implementing earlier versions of the Firmware Update Meta Data CC do not support status codes defined for newer versions. Therefore, a node returning the Firmware Update Meta Data Status Report Command **MUST** comply with the version implemented on the controlling device (device sending the image). The device returning the Firmware Update Meta Data Status Report can identify the version of the controlling device based on the Firmware Update Meta Data Request Get. This may be done as follows:

- If the Firmware Update Meta Data Request Get does not include Firmware Target and Fragment Size (8 bytes), the controller is version 1 or 2
- If the Firmware Update Meta Data Request Get includes Firmware Target and Fragment Size but not Activation (11 bytes), the controller is version 3

WaitTime (16 bits)

- CC:007A.03.07.11.006 The WaitTime field MUST report the time that is needed before the receiving node again becomes available for communication after the transfer of an image. The unit is the second.
- CC:007A.03.07.11.007 The value 0 (zero) MUST indicate that the node is ready.
The value 0xFFFF is reserved and MUST NOT be returned.
- CC:007A.03.07.12.001 A controlling node receiving this command SHOULD wait for the number of seconds specified in this field before trying to resume communication. When resuming communication, the controlling application SHOULD issue a NOP command and wait for acknowledgement.
- CC:007A.03.07.12.002
- CC:007A.03.07.13.002 The controlling application MAY attempt resuming communication repeatedly. In that case, the NOP interval SHOULD be five seconds and MUST be at least one second.
- CC:007A.03.07.11.008

4.14 Firmware Update Meta Data Command Class, version 4

The Firmware Update Meta Data Command Class may be used to transfer a firmware image to a Z-Wave device.

4.14.1 Compatibility considerations

All commands and fields not described in this version remain unchanged from version 3.

CC:007A.04.00.21.001

A device implementing Firmware Update Meta Data Command Class, version 4 MUST also implement Firmware Update Meta Data Command Class, versions 3.

CC:007A.04.00.21.002

While support for version 1 has been deprecated, a controlling device implementing Firmware Update Meta Data Command Class, version 2 MUST also be able to control Firmware Update Meta Data Command Class, version 1 based devices.

4.14.1.1 New commands, fields and values

The Firmware Update Meta Data Command Class version 4 supports delaying and scheduling a firmware update after downloading the image.

Version 4 updates the Firmware Update Meta Data Request Get Command and introduces the following commands:

- New Firmware Update Activation Set Command
- New Firmware Update Activation Status Report

After downloading an image, a device may use the Firmware Update Meta Data Status Report Command to indicate to the controlling device that the device is waiting for an activation command.

4.14.2 Interoperability considerations

4.14.2.1 Interoperability with v1 devices

Version 2 of the Firmware Update Meta Data Report command introduces a 16-bit Checksum field to follow the variable-length Data field. No fields follow the variable-length Data field in Version 1 of the Firmware Update Meta Data Report command. Version 4 of the Firmware Update Meta Data Report command uses the v2 format.

At the same time, the specified method for determining the length of the variable-length Data field is that the length must be calculated from the length of the received frame.

The unintended consequence is that a version 1 implementation receiving a Firmware Update Meta Data Report v2 command will consider the 16-bit Checksum field to be the last two bytes of the Data field. Similarly, a version 2 implementation receiving a Firmware Update Meta Data Report v1 command will consider the last two bytes of the Data field to be the 16-bit Checksum field. In either case, the transfer will fail.

CC:007A.04.00.31.001

Therefore, a controlling device implementing the Firmware Update Meta Data Command Class, version 2 **MUST** do the following when initiating the transfer of a firmware image:

1. Request the version of the Firmware Update Meta Data Command Class from the target device
2. Use the version of the Firmware Update Meta Data Report implemented by the target device
3. Determine the number of Data bytes that can fit into the Firmware Update Meta Data Report command so that the complete command can still fit into the payload field of the transport frame. The number of available payload bytes in the frame depends on the actual bit rate, the use of security encapsulation as well as the presence of Checksum bytes in the Firmware Update Meta Data Report

4.14.2.2 Checksum calculation

Version 1 of the Firmware Update Meta Data Command Class introduced a 16-bit Checksum field used to verify firmware image integrity. It has been a recommendation to use the CRC-CCITT polynomial for calculating checksums. Nodes having implemented another method for calculating the checksum will find a non-matching checksum.

From Version 5 onwards, the checksum calculation method is mandatory.

For more details about the checksum calculation, refer to Appendix B

4.14.3 Firmware Update Meta Data Request Get Command

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_REQUEST_GET							
Manufacturer ID 1							
Manufacturer ID 2							
Firmware ID 1							
Firmware ID 2							
Checksum 1							
Checksum 2							
Firmware Target							
Fragment Size 1							
Fragment Size 2							
Reserved						Activation	

All fields not described below remain unchanged from version 3.

Activation (1 bit)

The Activation field is used to advertise if the receiving node may delay the actual firmware update.

The field MUST be interpreted as:

- '1': The receiving device MAY delay the actual firmware update.
If the receiving node delays the firmware update, the delay MUST be advertised via the Status code 0xFD in the Firmware Update Meta data Status Report Command.
- '0': The receiving device MUST NOT delay the firmware update.

CC:007A.04.03.11.001

CC:007A.04.03.13.001

CC:007A.04.03.11.002

CC:007A.04.03.11.003

4.14.4 Firmware Update Meta Data Status Report Command

This command is used to advertise the firmware update status.

CC:007A.04.07.11.001 The command **MUST** be issued when the firmware update is completed or aborted by the device receiving the firmware.

CC:007A.04.07.11.002 A node **MUST NOT** issue the Firmware Update Meta Data Get Command after receiving a Firmware Update Meta Data Status Report.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_STATUS_REPORT							
Status							
WaitTime MSB							
WaitTime LSB							

All fields not described below remain unchanged from version 3.

Status (8 bits)

CC:007A.04.07.11.003 This field **MUST** comply with Table 10.

CC:007A.04.07.13.001 The status code 0xFD **MAY** be used by a version 4 device for confirming that an image has been successfully transferred but that the actual firmware update will not be performed until a Firmware Update Activation Set Command is received.

CC:007A.04.07.13.002 The Firmware Update Activation Set Command **MAY** be delayed for any period of time. The delay may be controlled via the Schedule Command Class.

CC:007A.04.07.13.003 The status code 0xFE **MAY** be used for confirming the successful transfer of an image which does not necessitate a restart, e.g. security a certificate.

CC:007A.04.07.11.004 The status code 0xFE **MUST NOT** be advertised after the transfer of an image for the Firmware ID 0 target (the "Z-Wave chip" image).

CC:007A.04.07.11.005 Controlling nodes implementing earlier versions of the Firmware Update Meta Data CC do not support status codes defined for newer versions. Therefore, a node returning the Firmware Update Meta Data Status Report Command **MUST** comply with the version implemented on the controlling device (device sending the image). The device returning the Firmware Update Meta Data Status Report can identify the version of the controlling device based on the Firmware Update Meta Data Request Get. This may be done as follows:

- If the Firmware Update Meta Data Request Get does not include Firmware Target and Fragment Size (8 bytes), the controller is version 1 or 2
- If the Firmware Update Meta Data Request Get includes Firmware Target and Fragment Size but not Activation (11 bytes), the controller is version 3
- If the Firmware Update Meta Data Request Get includes Activation (12 bytes), the controller is version 4

4.14.5 Firmware Update Activation Set Command

This command is used to initiate the programming of a previously transferred firmware image. Refer to the Firmware Update Meta Data Status Report Command Status code 0xFD.

CC:007A.04.08.13.001 This command MAY be issued directly by a controlling node or MAY be scheduled for later execution via the Schedule Command Class.

CC:007A.04.08.11.001 The Firmware Update Activation Status Report Command MUST be returned in response to this command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_ACTIVATION_SET							
Manufacturer ID 1							
Manufacturer ID 2							
Firmware ID 1							
Firmware ID 2							
Checksum 1							
Checksum 2							
Firmware Target							

For fields' description, refer to the Firmware Update Meta Data Request Get Command.

4.14.6 Firmware Update Activation Status Report

This command is used to advertise the result of a firmware update operation initiated by the Firmware Update Activation Set Command.

CC:007A.04.09.11.001

The Firmware Update Activation Status Report fields **MUST** reflect the values specified in the Firmware Update Activation Set Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_ACTIVATION_STATUS_REPORT							
Manufacturer ID 1							
Manufacturer ID 2							
Firmware ID 1							
Firmware ID 2							
Checksum 1							
Checksum 2							
Firmware Target							
Firmware Update Status							

For fields not described below, refer to the Firmware Update Meta Data Request Get Command.

Firmware Update Status (8 bits)

CC:007A.04.09.11.002

This field **MUST** comply with Table 11.

4.15 Firmware Update Meta Data Command Class, version 5

The Firmware Update Meta Data Command Class may be used to transfer a firmware image to or from a Z-Wave node.

4.15.1 Compatibility considerations

A device implementing Firmware Update Meta Data Command Class, version 5 MUST also implement Firmware Update Meta Data Command Class, version 4.

Firmware Update Meta Data Command Class, version 5 is backwards compatible with Firmware Update Meta Data Command Class, version 4.

All commands and fields not mentioned in this version remain unchanged from version 4.

4.15.1.1 New commands, fields and values

The Firmware Update Meta Data Command Class, version 5 introduces the support of a hardware identifier to uniquely identify firmware images that can be loaded on devices. This allows devices running identical software version on different hardware revisions to receive the correct firmware image.

The following commands are extended from version 4:

- Firmware Meta Data Report Command
- Firmware Update Meta Data Request Get Command
- Firmware Update Meta Data Request Report Command
- Firmware Update Meta Data Status Report Command
- Firmware Update Activation Set Command
- Firmware Update Activation Status Report Command

The Firmware Update Meta Data Command Class, version 5 also introduces the support firmware download from the Z-Wave node to the controller.

The following commands are introduced:

- Firmware Update Meta Data Prepare Get Command
- Firmware Update Meta Data Prepare Report Command

4.15.2 Interoperability Considerations

4.15.2.1 Interoperability with v1 devices

Version 2 of the Firmware Update Meta Data Report command introduces a 16-bit Checksum field to follow the variable-length Data field. No fields follow the variable-length Data field in Version 1 of the Firmware Update Meta Data Report command. Version 5 of the Firmware Update Meta Data Report command uses the version 2 format.

At the same time, the specified method for determining the length of the variable-length Data field is that the length must be calculated from the length of the received frame.

The unintended consequence is that a version 1 implementation receiving a Firmware Update Meta Data Report version 2 Command will consider the 16-bit Checksum field to be the last two bytes of the Data field.

Similarly, a version 2 implementation receiving a Firmware Update Meta Data Report version 1 Command will consider the last two bytes of the Data field to be the 16-bit Checksum field. In either case, the transfer will fail.

Therefore, a controlling device implementing the Firmware Update Meta Data Command Class, version 2 **MUST** do the following when initiating the transfer of a firmware image:

1. Request the version of the Firmware Update Meta Data Command Class from the target device
2. Use the version of the Firmware Update Meta Data Report implemented by the target device
3. Determine the number of Data bytes that can fit into the Firmware Update Meta Data Report command so that the complete command can still fit into the payload field of the transport frame. The number of available payload bytes in the frame depends on the actual bit rate, the use of security encapsulation as well as the presence of Checksum bytes in the Firmware Update Meta Data Report

4.15.2.2 Checksum calculation

Version 1 of the Firmware Update Meta Data Command Class introduced a 16-bit Checksum field used to verify firmware image integrity. It has been a recommendation to use the CRC-CCITT polynomial for calculating checksums. There is no way to identify what method has been used for calculating the checksum for nodes implementing version 1 to version 4. Nodes having implemented another method for calculating the checksum will find a non-matching checksum.

From Version 5 onwards, the checksum calculation method is mandatory.

For more details about the checksum calculation, refer to Appendix B

CC:007A.05.00.31.001

4.15.3 Firmware Meta Data Report Command

This command is used to advertise the status of the current firmware in the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_MD_REPORT							
Manufacturer ID 1							
Manufacturer ID 2							
Firmware 0 ID 1							
Firmware 0 ID 2							
Firmware 0 Checksum 1							
Firmware 0 Checksum 2							
Firmware Upgradable							
Number of Firmware Targets							
Max Fragment Size 1							
Max Fragment Size 2							
Firmware 1 ID 1							
Firmware 1 ID 2							
...							
Firmware N ID 1							
Firmware N ID 2							
Hardware Version							

All fields not described below remain unchanged from version 4.

Firmware 0 Checksum (16 bits)

The checksum field MUST carry the checksum of the Firmware 0 Image.

CC:007A.05.02.11.001

The checksum MUST be calculated using the CRC-CCITT polynomial using initialization value equal to 0x1D0F and 0x1021 (normal representation).

For more details, refer to Appendix B

Hardware Version (8 bits)

- CC:007A.05.02.11.002 This field MUST report a value which is unique to this particular version of the product.
- CC:007A.05.02.11.003 It MUST be possible to uniquely identify applicable firmware images via the Manufacturer ID, Firmware ID and the Hardware Version fields. This information allows selecting a firmware image that is guaranteed to work with this particular version of the product.
- CC:007A.05.02.11.004 The Hardware Version field MUST apply to the entire product and not only to the version of the Z-Wave radio chip.
- CC:007A.05.02.11.005 This field MUST report the same value as the Version CC Version Report Command: Hardware Version field.

4.15.4 Firmware Update Meta Data Request Get Command

This command is used to request that a firmware update is initiated by the node receiving this command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_REQUEST_GET							
Manufacturer ID 1							
Manufacturer ID 2							
Firmware ID 1							
Firmware ID 2							
Firmware Checksum 1							
Firmware Checksum 2							
Firmware Target							
Fragment Size 1							
Fragment Size 2							
Reserved							Activation
Hardware Version							

All fields not described below remain unchanged from version 4.

Firmware Checksum (16 bits)

- CC:007A.05.03.11.001 The checksum field MUST carry the checksum of the firmware image about to be transferred.
- CC:007A.05.03.11.002 The checksum MUST be calculated using the CRC-CCITT polynomial using initialization value equal to 0x1D0F and 0x1021 (normal representation).

For more details, refer to Appendix B

Hardware Version (8 bits)

Refer to 4.15.3 Firmware Meta Data Report Command.

4.15.5 Firmware Update Meta Data Request Report Command

This command is used to advertise if the firmware update will be initiated.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_REQUEST_REPORT							
Status							

Status (8 bits)

CC:007A.05.04.11.001

The Status field **MUST** comply with Table 9.

Table 9, Firmware Update Meta Data Request Report::Status encoding

Status	Description	Version
0x00	ERROR. Invalid combination of Manufacturer ID and Firmware ID. The device will not initiate the firmware update.	1
0x01	ERROR. Device expected an authentication event to enable firmware update. The device will not initiate the firmware update.	1
0x02	ERROR. The requested Fragment Size exceeds the Max Fragment Size. The device will not initiate the firmware update.	3
0x03	ERROR. This firmware target is not upgradable. The device will not initiate the firmware update.	3
0x04	ERROR. Invalid Hardware Version. The device will not initiate the firmware update.	5
0xFF	OK. The device will initiate the firmware update of the target specified in the Firmware Update Meta Data Request Get Command.	1

CC:007A.05.04.11.002

All other values are reserved and **MUST NOT** be used by a sending node. Reserved values **MUST** be ignored by a receiving node.

4.15.6 Firmware Update Meta Data Status Report Command

This command is used to advertise the firmware update status.

CC:007A.05.07.11.001 The command **MUST** be issued when the firmware update is completed or aborted by the device receiving the firmware.

CC:007A.05.07.11.002 A device **MUST NOT** issue the Firmware Update Meta Data Get Command after receiving a Firmware Update Meta Data Status Report.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_STATUS_REPORT							
Status							
WaitTime MSB							
WaitTime LSB							

All fields not described below are the same as in version 4.

Status (8 bits)

CC:007A.05.07.11.003 This field **MUST** comply with Table 10.

CC:007A.05.07.13.001 The status code 0xFD **MAY** be used by a version 4 device for confirming that an image has been successfully transferred but that the actual firmware update will not be performed until a Firmware Update Activation Set Command is received.

CC:007A.05.07.13.002 The Firmware Update Activation Set Command **MAY** be delayed for any period of time. The delay may be controlled via the Schedule Command Class.

CC:007A.05.07.13.003 The status code 0xFE **MAY** be used for confirming the successful transfer of an image which does not necessitate a restart, e.g. security a certificate.

CC:007A.05.07.11.004 The status code 0xFE **MUST NOT** be advertised after the transfer of an image for the Firmware ID 0 target (the "Z-Wave chip" image).

Controlling nodes implementing earlier versions of the Firmware Update Meta Data CC do not support status codes defined for newer versions. Therefore, a device returning the Firmware Update Meta Data Status Report Command **MUST** comply with the version implemented on the controlling device (device sending the image). The device returning the Firmware Update Meta Data Status Report can identify the version of the controlling device based on the Firmware Update Meta Data Request Get. This may be done as follows:

- If the Firmware Update Meta Data Request Get does not include Firmware Target and Fragment Size (8 bytes), the controller is version 1 or 2
- If the Firmware Update Meta Data Request Get includes Firmware Target and Fragment Size but not Activation (11 bytes), the controller is version 3
- If the Firmware Update Meta Data Request Get includes Activation but not Hardware Version (12 bytes), the controller is version 4
- If the Firmware Update Meta Data Request Get includes Hardware Version (13 bytes), the controller is version 5.

Table 10, Firmware Update Meta Data Status Report::Status encoding

Status	Description	Version
0x00	The device was unable to receive the requested firmware data without checksum error. Number of retries and request sequence of missing frames are implementation specific. The image is not stored.	1
0x01	The device was unable to receive the requested firmware data. Number of retries and request sequence of missing frames are implementation specific. The image is not stored.	1
0x02	The transferred image does not match the Manufacturer ID. The image is not stored.	4
0x03	The transferred image does not match the Firmware ID. The image is not stored.	4
0x04	The transferred image does not match the Firmware Target. The image is not stored.	4
0x05	Invalid file header information. The image is not stored.	4
0x06	Invalid file header format. The image is not stored.	4
0x07	Insufficient memory. The image is not stored.	4
0x08	The transferred image does not match the Hardware version The image is not stored	5
...	<i>Reserved</i>	-
0xFD	Firmware image downloaded successfully, waiting for activation command.	4
0xFE	New image was successfully stored in temporary non-volatile memory. The device does not restart itself. This Status code MUST NOT be used when updating the Z-Wave chip image	3
0xFF	New image was successfully stored in temporary non-volatile memory. The device will now start storing the new image in primary non-volatile memory dedicated to executable code. Then the device will restart itself.	1

CC:007A.05.07.11.006

All other values are reserved and MUST NOT be used by a sending node. Reserved values MUST be ignored by a receiving node.

4.15.7 Firmware Update Activation Set Command

This command is used to initiate the programming of a previously transferred firmware image. Refer to the Firmware Update Meta Data Status Report Command Status code 0xFD.

CC:007A.05.08.13.001 This command MAY be issued directly by a controlling node or MAY be scheduled for later execution via the Schedule Command Class.

CC:007A.05.08.11.001 The Firmware Update Activation Status Report Command MUST be returned in response to this command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_ACTIVATION_SET							
Manufacturer ID 1							
Manufacturer ID 2							
Firmware ID 1							
Firmware ID 2							
Firmware Checksum 1							
Firmware Checksum 2							
Firmware Target							
Hardware Version							

All fields not described below remain unchanged from version 4.

Firmware Checksum (16 bits)

CC:007A.05.08.11.002 The checksum field MUST carry the checksum of the firmware image about to be activated.

CC:007A.05.08.11.003 The checksum MUST be calculated using the CRC-CCITT polynomial using initialization value equal to 0x1D0F and 0x1021 (normal representation).

For more details, refer to Appendix B

Hardware Version (8 bits)

Refer to 4.15.3 Firmware Meta Data Report Command.

4.15.8 Firmware Update Activation Status Report Command

This command is used to advertise the result of a firmware update operation initiated by the Firmware Update Activation Set Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_ACTIVATION_STATUS_REPORT							
Manufacturer ID 1							
Manufacturer ID 2							
Firmware ID 1							
Firmware ID 2							
Firmware Checksum 1							
Firmware Checksum 2							
Firmware Target							
Firmware Update Status							
Hardware Version							

All fields not described below remain unchanged from version 4.

Firmware Checksum (16 bits)

CC:007A.05.09.11.001 The checksum field **MUST** carry the checksum of the firmware image activated by the Firmware Update Activation Set Command.

CC:007A.05.09.11.002 The checksum **MUST** be calculated using the CRC-CCITT polynomial using initialization value equal to 0x1D0F and 0x1021 (normal representation).

For more details, refer to Appendix B

Hardware Version (8 bits)

Refer to 4.15.3 Firmware Meta Data Report Command.

Firmware Update Status (8 bits)

CC:007A.05.09.11.003 The Firmware Update Status field **MUST** comply with Table 11.

Table 11, Firmware Update Activation Status Report::Firmware Update Status encoding

Status	Description	Version
0x00	Invalid combination of manufacturer ID, firmware ID and Hardware Version or Firmware Target. The received image will not be stored.	4
0x01	Error activating the firmware. Last known firmware image has been restored.	4
...	<i>Reserved</i>	-
0xFF	Firmware update completed successfully.	4

All other values are reserved and MUST NOT be used by a sending node. Reserved values MUST be ignored by a receiving node.

4.15.9 Firmware Update Meta Data Prepare Get Command

This command is used to request that a firmware download is initiated by the node sending this command.

CC:007A.05.0A.11.001 The Firmware Update Meta Data Prepare Report Command **MUST** be returned in response to this command when the receiving node is ready to send the requested firmware image.

CC:007A.05.0A.11.002 This command **MUST NOT** be issued via multicast addressing.
 CC:007A.05.0A.11.003 A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

CC:007A.05.0A.12.001 Any image transferred via the Firmware Update Meta Data Command Class **SHOULD** include a fingerprint value to enable the validation of the integrity of the entire image after the transfer. The image and the fingerprint value **MUST** be packed in one entity which can be stored in one file and transferred as one entity.
 CC:007A.05.0A.11.004

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_PREPARE_GET							
Manufacturer ID 1							
Manufacturer ID 2							
Firmware ID 1							
Firmware ID 2							
Firmware Target							
Fragment Size 1							
Fragment Size 2							
Hardware Version							

Refer to 4.15.4 Firmware Update Meta Data Request Get Command for field description.

4.15.10 Firmware Update Meta Data Prepare Report Command

This command is used to advertise if the firmware image has been prepared and is ready to be transferred.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_FIRMWARE_UPDATE_MD							
Command = FIRMWARE_UPDATE_MD_PREPARE_REPORT							
Status							
Firmware Checksum 1							
Firmware Checksum 2							

Status (8 bits)

CC:007A.05.0B.11.001 The Status field MUST comply with Table 12.

Table 12, Firmware Update Meta Data Prepare Report::Status encoding

Status	Description	Version
0x00	ERROR. Invalid combination of Manufacturer ID and Firmware ID. The receiving node MUST NOT initiate the firmware download.	5
0x01	ERROR. Device expected an authentication event to enable firmware update. The receiving node MUST NOT initiate the firmware download.	5
0x02	ERROR. The requested Fragment Size exceeds the Max Fragment Size. The receiving node MUST NOT initiate the firmware download.	5
0x03	ERROR. This firmware target is not downloadable. The receiving node MUST NOT initiate the firmware download.	5
0x04	ERROR. Invalid Hardware Version. The receiving node MUST NOT initiate the firmware download.	5
0xFF	OK. The receiving node can initiate the firmware download of the target specified in the Firmware Update Meta Data Prepare Get Command.	5

CC:007A.05.0B.11.002 All other values are reserved and MUST NOT be used by a sending node. Reserved values MUST be ignored by a receiving node.

Firmware Checksum (16 bits)

CC:007A.05.0B.11.003 The checksum field MUST carry the checksum of the firmware image about to be transferred.

CC:007A.05.0B.11.004 The checksum MUST be calculated using the CRC-CCITT polynomial using initialization value equal to 0x1D0F and 0x1021 (normal representation).

For more details, refer to Appendix B

4.15.11 Examples

4.15.11.1 Identifying firmware revisions

The different fields Manufacturer ID, Firmware ID and Hardware version are used to identify compatible firmware image with the product.

The Version Command Class is also used for identifying the Firmware version/subversion

An example of the different fields' usage for a wall outlet is shown in Table 13

Table 13, Labelling different Firmware revisions

Product	Firmware identification			
	Manufacturer ID	Hardware Version	Firmware ID	Firmware version /sub-version
Initial Wall Outlet	0x0001	0x01	0x0001	0x01 / 0x01
New revision				
Fixing previous bugs	0x0001	0x01	0x0001	0x01 / 0x02
New revision				
Introducing new features (e.g. multiple press)	0x0001	0x01	0x0001	0x02 / 0x01
New revision				
US version (initial version)	0x0001	0x01	0x0001	0x02 / 0x01
China version (matching local requirements)	0x0001	0x01	0x0002	0x02 / 0x01
New revision				
Proximity Sensor added (US)	0x0001	0x02	0x0001	0x02 / 0x01
Proximity Sensor added (China)	0x0001	0x02	0x0002	0x02 / 0x01
New revision				
Adding S2 Capability (without proximity sensor, US)	0x0001	0x01	0x0001	0x03 / 0x01
Adding S2 Capability (without proximity sensor, China)	0x0001	0x01	0x0002	0x03 / 0x01
Adding S2 Capability (with proximity sensor, US)	0x0001	0x02	0x0001	0x03 / 0x01
Adding S2 Capability (with proximity sensor, US)	0x0001	0x02	0x0002	0x03 / 0x01

An illustration of a controller retrieving the firmware information is given in Figure 5

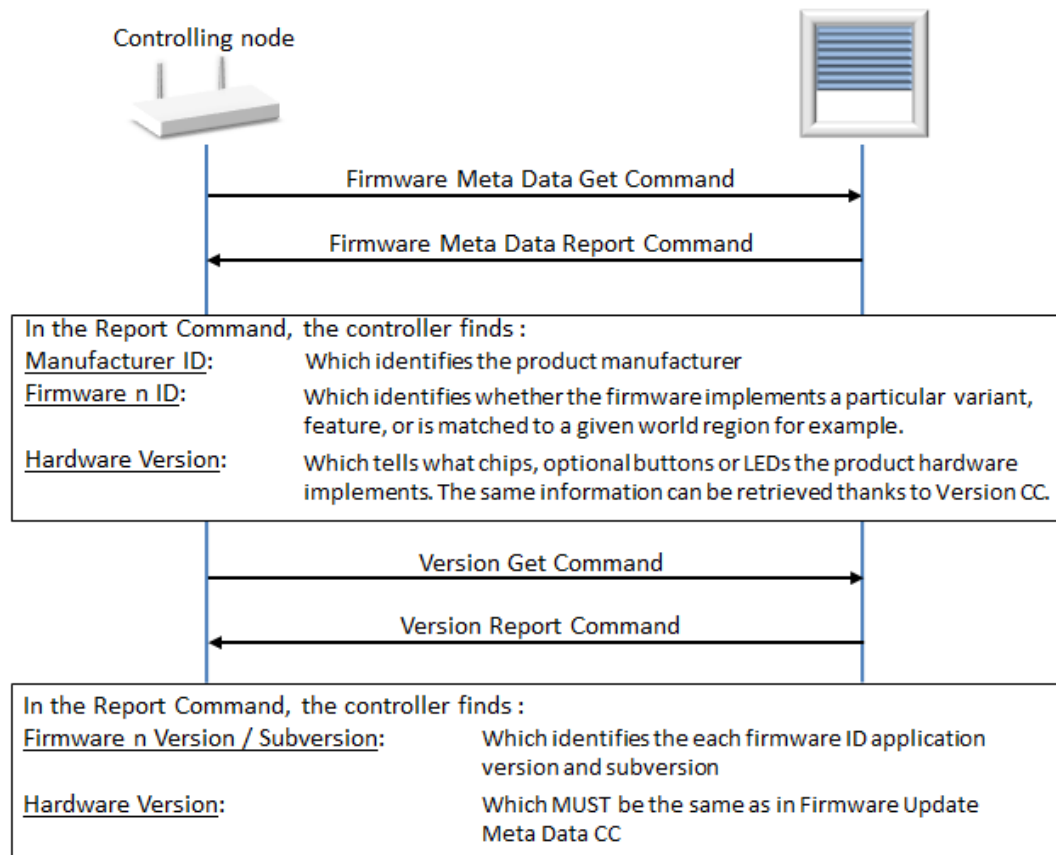


Figure 5, Identifying compatible firmware images

4.15.11.2 Firmware download

The use of the Firmware Update Meta Data Prepare Get and Firmware Update Meta Data Prepare Report commands is illustrated in Figure 6.

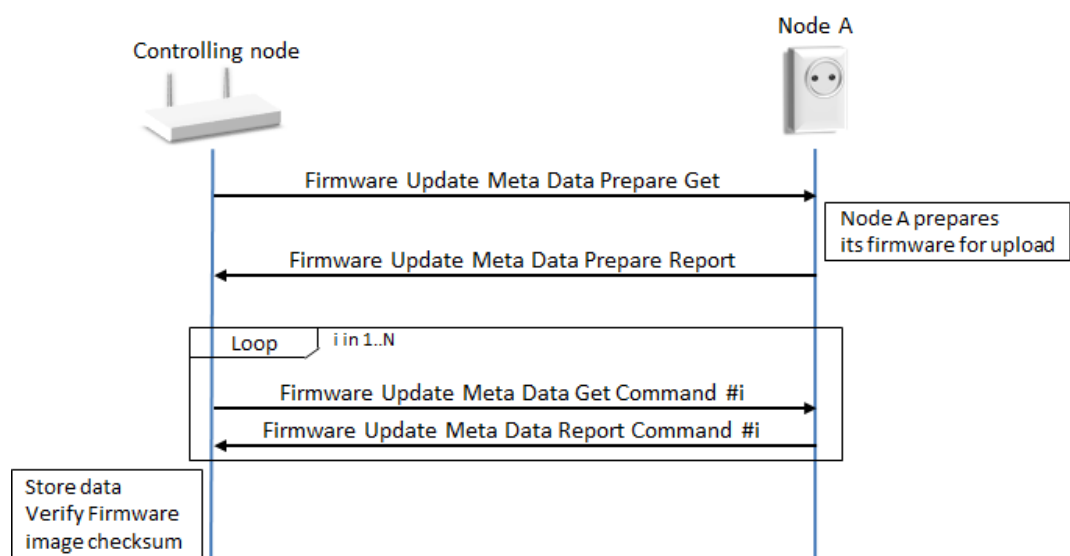


Figure 6, Firmware download flow diagram

4.16 Grouping Name Command Class, version 1 [DEPRECATED]

THIS COMMAND CLASS HAS BEEN DEPRECATED

A device MAY implement this command class, but it is RECOMMENDED that new implementations use the Association Group Information (AGI) Command Class for naming association groups. If implementing this command class, it is RECOMMENDED that the Association Group Information (AGI) Command Class is also implemented.

The Grouping Name Command Class is used to transfer name of groupings (as defined by the grouping identifier in the Association Command Class).

4.16.1 Grouping Name Set Command

The Grouping Name Set Command used to set a grouping Identifier name.

7	6	5	4	3	2	1	0			
Command Class = COMMAND_CLASS_GROUPING_NAME										
Command = GROUPING_NAME_SET										
Grouping identifier										
Reserved					Char. Presentation					
Grouping Name 1										
...										
Grouping Name N										

Grouping Identifier (8 bits)

The field specifies the Grouping ID.

Reserved

This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Char. Presentation (3 bits)

The char presentation identifier MAY be set to the following values:

Char. Presentation	Description
0	Using standard ASCII codes, see appendices in [2] (values 128-255 are ignored)
1	Using standard and OEM Extended ASCII codes, see appendices in [2]
2	Unicode UTF-16

Note: Devices supporting Unicode UTF-16 characters can have strings of a maximum of 8 characters because each character is described by a 2 byte long decimal representation. The first byte is the most significant byte. I.e. if there is one Unicode character in the set frame the char 1 will be MSB and char 2 will be LSB of the Unicode character.

All other values are reserved and MUST NOT be used by a sending node. Reserved values MUST be ignored by a receiving node.

Grouping Name (N bytes)

Grouping name using specified character representation.

The Grouping name MAY have a maximum of 16 characters and a minimum of 0 characters. The number of character fields transmitted MUST be determined from the frame length. If a frame with more than 16 characters is received, the receiving node MUST ignore any characters following the 16th character.

4.16.2 Grouping Name Get Command

The Grouping Name Get Command is used to request a grouping Identifier name.

The Grouping Name Report Command MUST be returned in response to this command.

This command MUST NOT be issued via multicast addressing.

A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_GROUPING_NAME							
Command = GROUPING_NAME_GET							
Grouping identifier							

Grouping Identifier (8 bits)

The field specifies the grouping identifier.

4.16.3 Group Name Report Command

The Grouping Name Report Command is used to report the grouping Identifier name.

7	6	5	4	3	2	1	0			
Command Class = COMMAND_CLASS_GROUPING_NAME										
Command = GROUPING_NAME_REPORT										
Grouping identifier										
Reserved					Char. Presentation					
Grouping Name 1										
...										
Grouping Name N										

Grouping Identifier (8 bits)

The field specifies the grouping identifier.

Reserved

This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Char. Presentation (3 bits)

Refer to description under the Grouping Name Set Command

Grouping Name (N bytes)

The Grouping Name fields contain the assigned group name. The Group Name can have a maximum of 16 characters and a minimum of 0 characters.

4.17 Hail Command Class, version 1 [OBSOLETE]

THIS COMMAND CLASS HAS BEEN OBSOLETE

New implementations MUST report local state updates via the Lifeline Association Group or use dedicated Command Classes for application specific purposes. Refer to [10]

The Hail Command Class used by applications to hail other devices in the Z-Wave network. The usage of the Hail Command Class is application specific.

4.17.1 Hail Command

Application can send unsolicited Hail Command to other devices in a Z-Wave network.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HAIL							
Command = HAIL							

4.18 Indicator Command Class, version 1

The Indicator Command Class is used to help end users to monitor the operation or condition of the application provided by a supporting node.

4.18.1 Interoperability considerations

Version 1 supporting nodes implement a single indicator resource. The indicator resource can only be turned on or off.

Nodes supporting the Wake-Up Command Class SHOULD keep the indicator in the same state (On/Off) as last instructed by an Indicator Set Command.

4.18.2 Indicator Set Command

This command is used to enable or disable the indicator resource.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_INDICATOR							
Command = INDICATOR_SET							
Value							

Value (8 bits)

This field is used to enable or disable the indicator resource.

This field MUST be in the range 0x00..0x63 or 0xFF.

The value 0x00 MUST indicate that the indicator MUST be turned off/disabled.
Values in the range 0x01..0x63 MUST indicate that the indicator MUST be turned on/enabled.
The value 0xFF MUST indicate that the indicator MUST be turned on/enabled.

4.18.3 Indicator Get Command

This command is used to request the state of the indicator resource.

The Indicator Report Command MUST be returned in response to this command.

This command MUST NOT be issued via multicast addressing.
A receiving node MUST NOT return a response if this command is received via multicast addressing.
The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_INDICATOR							
Command = INDICATOR_GET							

4.18.4 Indicator Report Command

This command is used to advertise the current state of the indicator resource.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_INDICATOR							
Command = INDICATOR_REPORT							
Value							

Value (8 bits)

This field is used to advertise the current state of the indicator resource. This field **MUST** be in the range 0x00..0x63 or set to 0xFF.

The value 0x00 **MUST** indicate that the indicator is turned off/disabled.

Values in the range 0x01..0x63 **MUST** indicate that the indicator is turned on/enabled.

The value 0xFF **MUST** indicate that the indicator is turned on/enabled.

4.19 Indicator Command Class, version 2

The Indicator Command Class, version 2 is used to manipulate indicator resources in a supporting node. An indicator may be an LED, an LCD display or a buzzer.

4.19.1 Compatibility Considerations

The Indicator Command Class, version 1 provides a single unspecified indicator resource. Version 2 introduces support for multiple indicator resources. Each indicator resource may implement a number of properties (or capabilities) such as turning on/off, set on a specific level or toggling state.

A controlling node may discover the supported indicators ID and their property IDs via the Indicator Supported Report Command.

The Indicator Command Class, version 2 renames the *Value* field of version 1 to “Indicator 0 Value”. The Indicator 0 Value field properties are unspecified as in version 1.

A version 2 supporting node MUST map the Indicator 0 to one of its supported Indicator ID and Property ID. The Indicator 0 SHOULD be mapped to an indicator 0x01 (Multilevel) or 0x02 (Binary) Property ID.

A supporting node advertising no support for the “Low power” indication Property ID and supporting the Wake Up Command Class MUST keep awake as long as one of the property ID is not 0x00 (off) for the actual Indicator ID. Such a node MUST return to sleep after both the Wake Up No More Information Command is received and all Property IDs are back to 0x00 for the actual Indicator ID.

A supporting node advertising support for the “Low power” indication Property ID and supporting the Wake Up Command Class MAY return to sleep even if an indication is ongoing for the actual Indicator ID.

4.19.2 Interoperability considerations

A supporting device MAY use its indicator resources for local status reporting until the first Indicator Set command is received.

After receiving an Indicator Set command, the supporting device MUST NOT use its indicator resources for local status reporting until it has been reset to default or re-included in the network.

An exception to this MAY be interface feedback functionality like a short beep on each button press. Likewise, transmission failure to a controlling node MAY be advertised via the use of local indications.

As an example, an entry control keypad device may feature an LCD backlight. If the backlight is advertised as an indicator resource, a central control application also has to control the backlight of the keypad, e.g. in response to local user activity and to temporarily draw attention to all keypads when an alarm is enabled.

4.19.3 Indicator Set Command

This command is used to manipulate one or more indicator resources at a supporting node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_INDICATOR							
Command = INDICATOR_SET							
Indicator 0 Value (Indicator ID 0 = 0x00, Property ID 0 = 0x01)							
Reserved			Indicator Object Count				
Indicator ID 1							
Property ID 1							
Value 1							
...							
Indicator ID N							
Property ID N							
Value N							

Indicator 0 Value (8 bits)

This field provides backwards compatibility for version 1 supporting nodes.

A receiving node MUST ignore this field if the *Indicator Object Count* field is not set to 0.
A receiving node MUST map this value to a supported indicator resource if the *Indicator Object Count* field is set to 0 or not included in the command.

Refer to 4.19.1 Compatibility Considerations.

Reserved

This field MUST be set to 0 by a sending device and MUST be ignored by a receiving device.

Indicator Object Count (5 bits)

This field is used to advertise the number of indicator objects carried in the actual command. An indicator object MUST comprise an *Indicator ID*, a *Property ID* and a *Value* field.

Indicator ID (8 bits, N times)

This field is used to identify the actual indicator resource. Indicator IDs values are defined in [21].

A receiving node MUST ignore the entire indicator object if an unsupported Indicator ID is specified.
A receiving node MUST NOT ignore other indicator objects included in the command if an unsupported Indicator ID is specified.

Property ID (8 bits, N times)

This field is used to identify the specific property of the indicator resource identified by the corresponding *Indicator ID* field.

A receiving node **MUST** ignore the entire indicator object if an unsupported Property ID for the corresponding Indicator ID is specified. The receiving node **MUST NOT** ignore other indicator objects included in the command.

If an *indicator ID* is specified in an object but not all *Property IDs* are included in the command, a supporting node **MUST** assume non-specified Property IDs values to be set to 0x00.

If several Property IDs not belonging to the same Property group defined in [21] are specified in the command, the Property IDs from one Property group **MUST** be applied and all other Property IDs **MUST** be ignored. For example, if Multilevel and Binary property IDs are defined, only one Property ID value **MUST** be applied and the other **MUST** be ignored.

Properties marked as “ADVERTISE ONLY:” **MUST** be ignored if received in a controlling command.

Value (8 bits, N times)

This field is used to specify the value to assign to the specific indicator property identified by the *Indicator ID* and the *Property ID* fields.

This field **MUST** be set according to the defined values and requirements for Property IDs in [21].

4.19.4 Indicator Get Command

This command is used to request the state of an indicator.

The Indicator Report Command **MUST** be returned in response to this command.

This command **MUST NOT** be issued via multicast addressing.

A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_INDICATOR							
Command = INDICATOR_GET							
Indicator ID							

Indicator ID (8 bits)

This field is used to specify the actual indicator resource.

If an unsupported Indicator ID is specified in this command, a receiving node **MUST** return an indicator object with the specified *Indicator ID* and the *Property ID* and *Value* fields set to 0x00.

4.19.5 Indicator Report Command

This command is used to advertise the state of an indicator resource.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_INDICATOR							
Command = INDICATOR_REPORT							
Indicator 0 Value (Indicator ID 0 = 0x00, Property ID 0 = 0x01)							
Reserved			Indicator Object Count				
Indicator ID 1							
Property ID 1							
Value 1							
...							
Indicator ID N							
Property ID N							
Value N							

Indicator 0 Value (8 bits)

This field provides backwards compatibility for version 1 controlling nodes.

A sending node **MUST** advertise the current value of the mapped Indicator ID and Property ID.

A version 2 controlling node **MUST** ignore the Indicator 0 Value field if other values are advertised. Refer to 4.19.1 Compatibility Considerations..

Reserved

This field **MUST** be set to 0 by a sending device and **MUST** be ignored by a receiving device.

Indicator Object Count (5 bits)

This field is used to advertise the number of indicator objects carried in this command. An indicator object **MUST** comprise an *Indicator ID*, a *Property ID* and a *Value* field.

Indicator ID (N bytes)

This field is used to identify the actual indicator resource.
All indicator objects **MUST** carry the same Indicator ID.

Property ID (N bytes)

This field is used to identify the specific property of the indicator resource identified by the corresponding *Indicator ID* field.

If not all Property IDs are included in the command for the actual Indicator ID, a controlling node **MUST** assume non-specified Property IDs values to be 0x00.

Value (N bytes)

This field is used to specify the current value of the specific indicator property identified by the *Indicator ID* and the *Property ID* fields.

This field **MUST** be set according to the defined values for Property IDs in [21].

4.19.6 Indicator Supported Get Command

This command is used to request the supported properties of an indicator.

The Indicator Supported Report Command **MUST** be returned in response to this command.

This command **MUST NOT** be issued via multicast addressing.

A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_INDICATOR							
Command = INDICATOR_SUPPORTED_GET							
Indicator ID							

Indicator ID (1 byte)

This field is used to specify the actual indicator resource.

A controlling node **SHOULD** set this field to zero to discover the supported Indicator IDs. A supporting node receiving this field set to 0x00 **MUST** advertise the first supported Indicator ID in response.

A supporting node receiving a non-zero Indicator ID that is not supported **MUST** set all fields (*Indicator ID*, *Next Indicator ID*, *Property Supported Bit Mask Length*) to 0x00 in the returned response.

4.19.7 Indicator Supported Report Command

This command is used to advertise the supported properties for a given indicator.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_INDICATOR							
Command = INDICATOR_SUPPORTED_REPORT							
Indicator ID							
Next Indicator ID							
Reserved			Property Supported Bit Mask Length				
Property Supported Bit Mask 1							
...							
Property Supported Bit Mask N							

Indicator ID (1 byte)

This field is used to specify the actual indicator ID resource for the supported properties are being advertised.

The supported Indicator IDs MUST be according to the Indicator IDs values defined in [21].

Reserved

This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Next Indicator ID (8 bits)

This field is used to advertise if more Indicator IDs are supported after the actual Indicator ID advertised by the *Indicator ID* field.

If a sending node supports additional Indicator IDs, this field MUST advertise the next supported Indicator ID.

If the sending node does not support additional Indicator IDs this field MUST be set to 0x00.

The supported Indicator IDs MUST be according to the Indicator IDs values defined in [21].

Property Supported Bit Mask Length (5 bits)

This field is used to advertise the length of the *Property Supported Bit Mask* field in bytes. The value MUST be in the range 0..32.

Property Supported Bit Mask (N bytes)

This field is used to advertise the properties supported by the actual Indicator ID.

The length of this field in bytes MUST be according to the *Property Supported Bit Mask Length* field value. This field MUST be encoded as a bitmask as follow:

- Bit 0 in Bit Mask 1 is not allocated to any property and MUST be set to zero.
- Bit 1 in Bit Mask 1 MUST indicate if Property ID = 1 (Multilevel) is supported.
- Bit 2 in Bit Mask 1 MUST indicate if Property ID = 2 (On/Off) is supported.
- ...

If the Property ID is supported, the corresponding bit MUST be set to 1.

If the Property ID is not supported the corresponding bit MUST be set to 0.

The supported Property IDs MUST be according to the Property IDs values defined in [21].

4.20 Indicator Command Class, version 3

The Indicator Command Class, version 3 is used to manipulate indicator resources in a supporting node. An indicator may be an LED, an LCD display or a buzzer.

4.20.1 Compatibility Considerations

The Indicator Command Class, version 3 is backwards compatible with the Indicator Command Class, version 2. All commands and fields not mentioned in this version **MUST** remain unchanged from the Indicator Command Class, version 2.

This version introduces new Indicator IDs and Property IDs. The list of supported Indicator IDs and Property IDs is moved to [21]. Values not defined in [21] are reserved and **MUST NOT** be used by a supporting node.

New values **MAY** be added to [21] and will be labelled under version 3.

4.21 IP Association Command Class, version 1

The IP Association Command Class is used to create and maintain Z-Wave application bindings between IPv6 hosts. An association group sends a predefined command to the configured destinations when triggered by an event.

IP Association identifies nodes via IPv6 addresses.

IP Association is natively Z-Wave Multi Channel End Point aware.

IP Association groups are specific to a particular End Point or Root Device.

4.21.1 Compatibility considerations

The IP Association Command Class extends the functionality of the Z-Wave Multi Channel Association Command Class. Z-Wave nodes may be represented as Z/IP nodes in an IP subnet. IP Associations may be mapped to classic Z-Wave Multi Channel associations.

The IP Association Remove Command extends the Multi Channel Association Remove Command by adding the ability to clear all groups in all End Points of a node. Only one IP association can be created or maintained with a single command.

In order to respond quickly and to conserve battery, it is **RECOMMENDED** that a Z/IP Gateway caches IP associations and corresponding Z-Wave associations and only looks up all associations when explicitly instructed to do so.

4.21.2 Terminology

The IP Association Command Class operates on IP primitives but works for all generations of Z-Wave technology. In the following, the term “Classic Z-Wave” refers to operations that are possible with Z-Wave-only releases.

The term “Z-Wave node” is used to identify a Z-Wave node as used in Classic Z-Wave while the term “Z/IP node” denotes an IP enabled Z-Wave node that implements the ICMP echo service and the Z-Wave UDP service. A Z/IP node may implement other IP services. Such IP services are out of scope of this document.

An IP association is created by sending an IP Association Set command from a configuration tool to the association source, instructing the association source to add an association destination to an association group maintained in the association source. The association destination has no knowledge that an association is created.

A Z/IP Gateway represents classic Z-Wave nodes as Z/IP nodes in an IP environment. The Z/IP Gateway therefore maps IP association commands from Z/IP clients to relevant Association and Multi Channel Association commands for Z-Wave nodes.

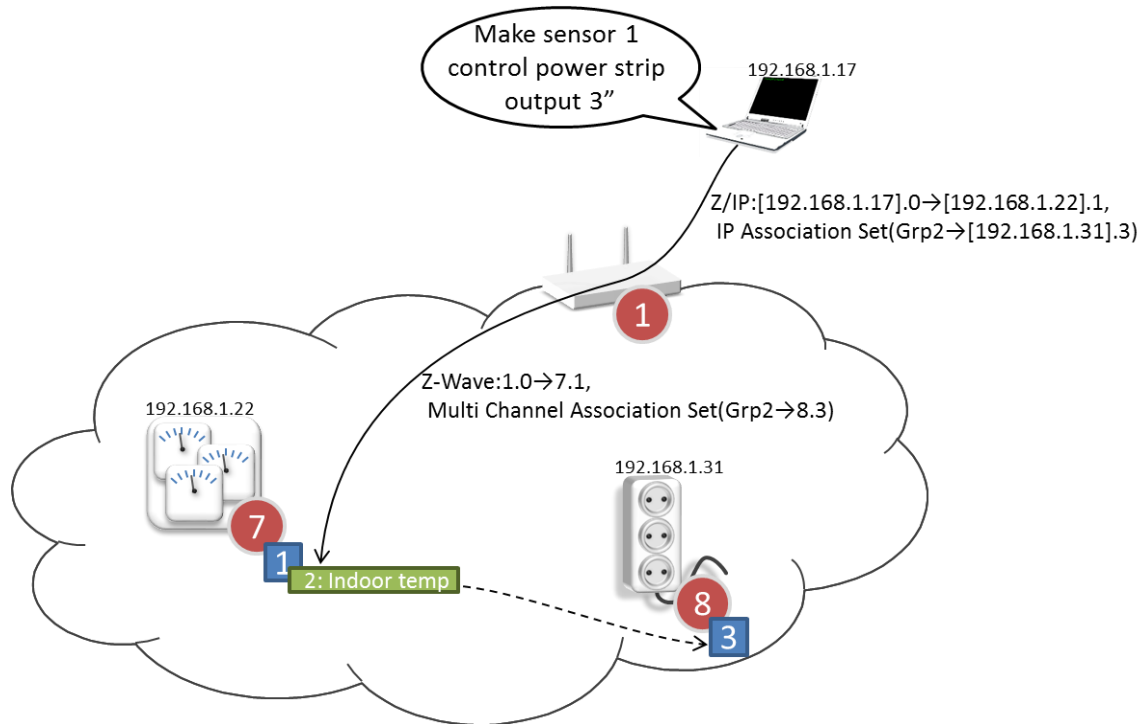


Figure 7, IP Association example

4.21.3 Z-Wave Multi Channel compatibility considerations

IP Association is natively Multi Channel aware. The same message format is used to create an association to a logical End Point as to the Root Device.

In the Z/IP framework, End Point 0 represents the physical entity.

IP Associations are conceptually created between two Z/IP resources identified by an IPv6 address and an End Point. In case one or both parties are Z-Wave nodes represented as Z/IP nodes, the Z/IP Gateway MUST create Z-Wave associations in the following way, based on the Resource Directory information pertaining to the actual nodes.

A Z/IP gateway MUST map IP associations to Z-Wave associations according to Table 14.

Table 14, IP association mapping to Z Wave association

Association source Multi Channel Capability	Association destination Multi Channel Capability	Association type	Creating the association
-	-	Root Device → Root Device	An Association Set Command MUST be sent un-encapsulated to the association source node
✓	✓	End Point m → End Point n m, n MAY be 0	A Multi Channel Association Set Command MUST be sent Multi Channel encapsulated to the association source End Point. A Multi Channel Association from the Lifeline association group of a sensor Root Device enables the transmission of sensor reports with different Source End Points to the Lifeline destination.
✓	-	End Point m → Root Device	An Association Set Command MUST be sent Multi Channel encapsulated to the association source End Point.
-	✓	Root Device → End Point n	An Association Set Command MUST be sent un-encapsulated to the association source node. This first association MUST target a NodeID owned by the Z/IP Gateway. A second association MUST be created from the abovementioned Z/IP Gateway to the Multi Channel End Point specified in the IP Association Set Command.

4.21.4 Advertising the IP Association Command Class

The Node Information Frame emitted by Classic Z-Wave nodes does not include IP Association. To allow IP clients to issue IP Association commands, the Z/IP Gateway MUST rewrite the list of supported command classes from the node, replacing the Association and Multi Channel Association Command Classes with the IP Association Command Class in the following situations:

- When a Node Info Cached Report is sent to an IP client.
- When a Node Add Status is being sent to an IP client.

In both cases, rewriting MUST NOT be performed if the report is being sent to a native Z-Wave node.

Furthermore, mDNS responses to queries for nodes supporting the IP Association Command Class MUST also advertise nodes supporting the Association and Multi Channel Association Command Classes (but the command classes must be replaced with `COMMAND_CLASS_IP_ASSOCIATION` before advertised via mDNS).

4.21.5 IP Association Set Command

This command is used to add one resource to a given association group.

The receiving IPv6 host **SHOULD** add the specified destination resource to the specified association group.

This command **MAY** be ignored if the association group is already full.

Unless the association destination is a gateway, a controlling node **SHOULD NOT** create an association if the association destination node does not support the controlling commands (Set/Get type) that the actual association group will be sending. The AGI Command Class **SHOULD** be used to probe the commands that a given association group will be sending.

A controlling node **SHOULD NOT** create an association if the source and destination nodes are bootstrapped with different security levels.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_IP_ASSOCIATION							
Command = IP_ASSOCIATION_SET							
Grouping identifier							
IPv6 Address 1							
..							
IPv6 Address 16							
End Point							

Grouping identifier (8 bits)

This field is used to specify the actual association group.

Grouping Identifiers **MUST** be assigned in a consecutive range starting from 1.

IPv6 Address (16 bytes)

This field specifies the full IPv6 address for the association destination.

The IPv6 address **MUST NOT** be compressed.

The IPv6 address **SHOULD** be in the ULA IPv6 prefix or in a globally routable IPv6 prefix.

The IPv6 address **MAY** be an IPv4-mapped IPv6 address.

The field **MUST NOT** carry a link-local IPv6 address.

End Point (8 bits)

This field is used in combination with the IPv6 Address to identify the actual resource.

The field represents the Z/IP Bit Address flag (bit 7) as well as the Z/IP 7-bit End Point identifier (bits 6..0).

The receiving node **MUST** treat this field as one scalar value when adding or removing associations; i.e. any 8-bit End Point value causes one association to be added.

4.21.6 IP Association Get Command

This command is used to request active associations for a given association group. The IP Association Report Command MUST be returned in response to this command.

This command MUST NOT be issued via multicast addressing.

A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_IP_ASSOCIATION							
Command = IP_ASSOCIATION_GET							
Grouping identifier							
Index							

Grouping identifier (8 bits)

This field is used to specify the actual association group.

Grouping Identifiers MUST be assigned in a consecutive range starting from 1.

If an unsupported Grouping Identifier is specified, the IP Association Report Command returned in response to this command MUST carry IPv6 address and End Point fields which are set to all-zeros.

Index (8 bits)

This field is used to specify an entry in the association table for the group identified by the Grouping Identifier.

A requesting host SHOULD start specifying the index value 1.

The actual number of nodes in the association group may be determined from the Actual Nodes field of the IP Association Report.

4.21.7 IP Association Report Command

This command is used to advertise one association group entry.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_IP_ASSOCIATION							
Command = IP_ASSOCIATION_REPORT							
Grouping identifier							
Index							
Actual Nodes							
IPv6 Address 1							
..							
IPv6 Address 16							
End Point							

Grouping identifier (8 bits)

This field is used to identify the actual group of nodes.

Index (8 bits)

This field is used to advertise the actual entry in the association group identified by the Grouping Identifier.

Actual Nodes (8 bits)

This value indicates the number of nodes in the association group advertised by the Grouping identifier field.

IPv6 Address (16 bytes)

This field carries a full IPv6 address with no compression. The address SHOULD be in the ULA IPv6 prefix or in a globally routable IPv6 prefix. The address MAY be an IPv4-mapped IPv6 address. The field MUST NOT carry a link-local IPv6 address.

If the Actual Nodes value is zero, the IPv6 Address field MUST be all zeros.

End Point (8 bits)

This field is used in combination with the IPv6 Address to identify the actual resource

If the Actual Nodes value is zero, the End Point field MUST be all zeros.

The field represents the Z/IP Bit Address flag (bit 7) as well as the Z/IP 7-bit End Point identifier (bits 6..0).

4.21.8 IP Association Remove Command

This command is used to remove one resource from a given association group.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_IP_ASSOCIATION							
Command = IP_ASSOCIATION_REMOVE							
Grouping identifier							
IPv6 Address 1							
..							
IPv6 Address 16							
End Point							

Grouping identifier (8 bits)

This field is used to specify the actual association group. Grouping Identifiers **MUST** be assigned in a consecutive range starting from 1.

A receiving node **MUST** ignore an unsupported Grouping Identifier.

This field **MUST** be interpreted in combination with the IPv6 Address and End Point fields.

IPv6 Address (16 bytes)

End Point (1 byte)

These fields specify the resources that are to be removed.

The Grouping identifier and these fields **MUST** be interpreted as follows.

Table 15, IP Association Remove, V1 :: Parameter interpretation

Command parameters			Removal operation
Grouping identifier	IPv6 Address	End Point ID	
> 0	<> 0	= 0 (Root Device)	Remove destination from association group in the association source Root Device
> 0	<> 0	> 0 (End Point)	Remove destination from association group in the specified association source End Point
> 0	= 0	= 0 (Root Device)	Remove all destinations from association group in the association source Root Device
> 0	= 0	> 0 (End Point)	Remove all destinations from association group in the specified association source End Point
= 0	= 0	= 0 (Root Device)	Remove all destinations from all association groups in the association source Root Device <u>and</u> in all association source End Points
= 0	= 0	> 0 (End Point)	Remove all destinations from all groups in the specified association source End Point
= 0	<> 0	>= 0	<i>Reserved</i>

*) The End Point field represents the Z/IP Bit Address flag (bit 7) as well as the Z/IP 7-bit End Point identifier (bits 6..0).

The receiving node **MUST** treat the End Point field as one scalar value when removing associations; i.e. any 8-bit End Point value causes one association to be removed.

4.22 Manufacturer Specific Command Class, version 1

The Manufacturer Specific Command Class is used to advertise manufacturer specific information. Version 2 of this command class further allows device specific information to be advertised.

4.22.1 Security Considerations

This Command Class provides information about the device implementation. This information can potentially be used by an attacker to find vulnerabilities.

CC:0072.01.00.41.003 A node supporting the Security 2 Command Class MUST support this Command Class according to the Device Type specification [10] requirements.

CC:0072.01.00.41.004 In order to increase interoperability with legacy controlling nodes, a node supporting the Security 0 Command Class MUST reply to Manufacturer Specific Get Commands received non-securely if it was granted the S0 network key as its highest Security Class.

CC:0072.01.00.43.001 In this case, it is OPTIONAL for the node to advertise the Manufacturer Specific Command Class in its NIF.

4.22.2 Manufacturer Specific Get Command

This command is used to request manufacturer specific information from another node.

CC:0072.01.04.11.001 The Manufacturer Specific Report Command MUST be returned in response to this command.

CC:0072.01.04.11.002 This command MUST NOT be issued via multicast addressing.

CC:0072.01.04.11.003 A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MANUFACTURER_SPECIFIC							
Command = MANUFACTURER_SPECIFIC_GET							

4.22.3 Manufacturer Specific Report Command

This command is used to advertise manufacturer specific device information.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MANUFACTURER_SPECIFIC							
Command = MANUFACTURER_SPECIFIC_REPORT							
Manufacturer ID 1							
Manufacturer ID 2							
Product Type ID 1							
Product Type ID 2							
Product ID 1							
Product ID 2							

Manufacturer ID (16 bits)

CC:0072.01.05.11.001 The Manufacturer ID field MUST carry the unique ID identifying the manufacturer of the device.

Manufacturer identifiers can be found in [12]. The first byte is the most significant byte.

Product Type ID (16 bits)

CC:0072.01.05.11.002 The Product Type ID field MUST carry a unique ID identifying the actual product type.

The first byte is the most significant byte.

CC:0072.01.05.11.003 A specific Product Type ID MUST be defined by the manufacturer for each type of product.

Product ID (16 bits)

CC:0072.01.05.11.004 The Product ID field MUST carry a unique ID identifying the actual product.

The first byte is the most significant byte.

CC:0072.01.05.11.005 A specific Product ID MUST be defined by the manufacturer for each product of a given product type. Thus, the same Product ID value may appear for different Product Type ID values.

4.23 Manufacturer Specific Command Class, version 2

Manufacturer Specific Command Class, version 2 adds a set of commands to communicate unique identification, e.g. the serial number, of the product.

Commands not mentioned here remains unchanged as specified for Manufacturer Specific Command Class, Version 1.

4.23.1 Security Considerations

A supporting node **MUST** comply with 4.22.1 Security Considerations

4.23.2 Device Specific Get Command

This command is used to request device specific information.

The Device Specific Report Command **MUST** be returned in response to this command.

This command **MUST NOT** be issued via multicast addressing.

A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MANUFACTURER_SPECIFIC							
Command = DEVICE_SPECIFIC_GET							
Reserved				Device ID Type			

Device ID Type (3 bits)

This field contains values for the Device ID Type.

Device ID Type	Value
Return OEM <u>factory default</u> Device ID Type	0
Serial Number	1
Pseudo Random	2
Reserved	3-7

A sending node **SHOULD** specify a value of zero when issuing the Device Specific Get command since the responding node may only be able to return one Device ID Type.

4.23.3 Device Specific Report Command

This command is used to advertise device specific information.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MANUFACTURER_SPECIFIC							
Command = DEVICE_SPECIFIC_REPORT							
Reserved					Device ID Type		
Device ID Data Format			Device ID Data Length				
Device ID Data 1							
...							
Device ID Data N							

Device ID Data Format (3 bits)

This command field defines the format used for Device ID Data.

Device ID Data Format	Value	Description
UTF-8	0x00	The Device ID Data MUST be in UTF-8 format.
Binary	0x01	The Device ID Data is in plain binary format and MUST be displayed as hexadecimal values e.g. 0x30, 0x31, 0x32, 0x33 MUST be displayed as h'30313233.

CC:0072.02.07.11.001

CC:0072.02.07.11.002

CC:0072.02.07.11.003

All other values are reserved and MUST NOT be used by a sending node. Reserved values MUST be ignored by a receiving node.

Device ID Type (3 bits)

This field contains values for the Device ID Type. See Device Specific Get Command for details.

CC:0072.02.07.13.001

In case the Device ID Type specified in a Device Specific Get command is not supported by the responding node, the responding node MAY return the factory default Device ID Type (as if receiving the value 0 in the Device Specific Get command).

Device ID Data Length(5 bits)

CC:0072.02.07.11.004

This field contains the length of the Device ID Data field. The field MUST NOT carry the value zero.

Device ID Data (N bytes)

Data fields for Device ID. "Device ID Data Format" defines data format and "Device ID Data Length Indicator" defines length.

CC:0072.02.07.11.005

The Manufacturer ID and Device ID combination MUST be globally unique.

4.24 Multi Channel Association Command Class, version 1 [OBSOLETE]

THIS VERSION HAS BEEN OBSOLETE

New implementations MUST use the Multi Channel Association Command Class, version 2 or newer.

CC:008E.01.00.11.001

4.25 Multi Channel Association Command Class, version 2

The Multi Channel Association Command Class is used to manage associations to Multi Channel End Point destinations as well as to NodeID destinations.

An association group sends an unsolicited command to the configured destinations when triggered by an event. The parameters of the command may be dynamic, e.g. the temperature of a sensor reading or the light level for a dimmer.

A NodeID association identifies its destination by a NodeID.

An End Point association identifies its destination by a combination of a NodeID and an End Point.

Refer to [15] for an introduction to the Multi Channel concept.

4.25.1 Compatibility considerations

The Multi Channel Association Command Class extends the functionality of the Association command class. A device supporting this Command Class version **MUST** also support the (non-Multi Channel) Association Command Class, version 2.

A controlling device **SHOULD NOT** create End Point associations to dynamic Multi Channel End Points.

The Association Group Information (AGI) Command Class **SHOULD** be supported to enable automated discovery of association group properties.

The Association and Multi Channel Association Command Classes **MUST** access the same collection of association groups. Further, the following applies:

The two command classes **MUST** advertise the total number of association groups.

The two command classes **MUST** advertise the total number of supported nodes for a given association group.

Any advertised association group **MUST** support NodeID destinations as well as End Point destinations.

NodeID associations maintained via the Multi Channel Association Command Class **MUST** be identical to NodeID associations maintained via the Association Command Class.

The Multi Channel Command Class specifies that, for backwards compatibility with non-Multi Channel devices, the Root Device of a Multi Channel device **MUST** mirror the functionality of End Point 1 and it **MAY** mirror the functionality of more End Points. This principle also applies to association groups advertised by the Root Device.

Each association group advertised by the Root Device, except for association group 1, **SHOULD** mirror an association group of an End Point.

If a Root Device association group mirrors an End Point association group, the Root Device **SHOULD** map all Association commands for that group to the mirrored End Point association group.

Except for association group 1 (the Z-Wave Plus Lifeline), a Multi Channel aware controlling device **SHOULD** ignore all Root Device association groups since they are just mirrored End Point association groups.

The use of encapsulation **MUST** comply with Table 16.

Table 16, V2 Associations and the transmissions they may trigger

Source → Destination	Association type	Allowed transmissions
NodeID → NodeID (V2)	NodeID	Non-encapsulated
End Point → NodeID (V2)	NodeID	Non-encapsulated
NodeID → End Point (V2)	End Point	Encapsulated *) (src≥0, dst>0)
End Point → End Point (V2)	End Point	Encapsulated (src>0, dst>0)
NodeID → Root Device	End Point	None **)

*) The source End Point MAY be different than 0 if the associated group is an End Point group mapped to the Root Device

**) Version 2 of this command class does not allow the destination End Point to be zero.

4.25.2 Z-Wave Plus considerations

The Z-Wave Plus certification program mandates that Association group 1 is reserved for the Lifeline association group. Group 1 MUST NOT be assigned to any other use than the Lifeline group. The actual Device Type specifies a mandatory list of commands which the device must be able to send to all lifeline group destinations. A manufacturer MAY add additional commands to the lifeline group.

End Points SHOULD NOT implement the Lifeline Association Group. End Points SHOULD report that zero NodeIDs are supported for association group 1.

The Z-Wave Plus certification program mandates support for the Association Group Information (AGI) Command Class if a device supports the Multi Channel Association Command Class.

4.25.3 Security considerations

A device which is included securely MUST NOT accept Association commands unless the commands are received via the highest security key assigned to the device.

4.25.4 Multi Channel Association Set Command

This command is used to request that one or more destinations are added to a given association group.

CC:008E.02.01.13.001 The destinations MAY be a mix of NodeID destinations and End Point destinations.

CC:008E.02.01.12.001 The receiving node SHOULD add the specified destinations to the specified association group.
CC:008E.02.01.13.002 This command MAY be ignored if the association group is already full.

CC:008E.02.01.11.001 Routing slaves MUST have return routes assigned to all association destinations.

CC:008E.02.01.11.008 Unless the association destination is a gateway, a controlling node MUST NOT create an association if the association destination node does not support the controlling commands (Set/Get types) that the actual association group will be sending. The AGI Command Class MUST be used to probe the commands that a given association group will be sending.

CC:008E.02.01.12.004 A controlling node SHOULD NOT create an association if the source and destination nodes are bootstrapped with different security levels.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL_ASSOCIATION							
Command = MULTI_CHANNEL_ASSOCIATION_SET							
Grouping Identifier							
NodeID 1							
...							
NodeID M							
Marker = MULTI_CHANNEL_ASSOCIATION_SET_MARKER							
Multi Channel NodeID 1							
Bit Address 1	End Point 1						
...							
Multi Channel NodeID N							
Bit Address N	End Point N						

Grouping Identifier (8 bits)

CC:008E.02.01.11.002 This field is used to specify the actual association group. Grouping Identifiers MUST be assigned in a consecutive range starting from 1.

CC:008E.02.01.11.003 A node that receives an unsupported Grouping Identifier MUST ignore this command

NodeID (M bytes)

This field specifies a list of NodeID destinations that are to be added to the specified association group as a NodeID association.

CC:008E.02.01.11.004 A NodeID association created via this field MUST be identical to a NodeID association created with the (non-Multi Channel) Association Set command.

Marker (8 bits)

This field is used to indicate the end of NodeID destinations and the start of End Point destinations. The Marker field **MUST** be set to the value MULTI_CHANNEL_ASSOCIATION_SET_MARKER. The field **MAY** be omitted if no End Point destinations are specified.

CC:008E.02.01.11.005
CC:008E.02.01.13.003

Multi Channel NodeID (N bytes)

The Multi Channel NodeID, Bit Address and End Point fields specify a list of End Points which are to be added to the specified association group as an End Point association.

The complete identification of an End Point destination requires a NodeID as well as an End Point identifier. This command **MAY** carry multiple copies of the same Multi Channel NodeID in combination with different End Point identifiers.

CC:008E.02.01.13.004

Bit Address + End Point (N bytes)

These fields **MUST** be processed in combination with the Multi Channel NodeID field.

CC:008E.02.01.11.006

The receiving node **SHOULD** treat the Bit Address flag and the End Point identifier as one scalar value; thus creating only one association group entry. Using a single scalar value enables:

CC:008E.02.01.12.005

- Better utilization of association group capacity
- Simple transmission of command for a multi-End Point destination
- Well-defined removal of bit addressed End Points

Refer to the Multi Channel Command Class for the actual encoding of bit addressed End Points.

CC:008E.02.01.11.007

The 7-bit End Point value **MUST** be in the range 1..127.

4.25.5 Multi Channel Association Get Command

This command is used to request the current destinations of a given association group.

CC:008E.02.02.11.001 The Multi Channel Association Report Command **MUST** be returned in response to this command.

CC:008E.02.02.11.002 This command **MUST NOT** be issued via multicast addressing.

CC:008E.02.02.11.003 A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL_ASSOCIATION							
Command = MULTI_CHANNEL_ASSOCIATION_GET							
Grouping Identifier							

Grouping Identifier (8 bits)

CC:008E.02.02.11.004 This field is used to specify the actual association group. Grouping Identifiers **MUST** be assigned in a consecutive range starting from 1.

CC:008E.02.02.12.001 A node that receives an unsupported Grouping Identifier **SHOULD** return information relating to Grouping Identifier 1.

4.25.6 Multi Channel Association Report Command

This command is used to advertise the current destinations for a given association group.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL_ASSOCIATION							
Command = MULTI_CHANNEL_ASSOCIATION_REPORT							
Grouping Identifier							
Max Nodes Supported							
Reports to Follow							
NodeID 1							
...							
NodeID M							
Marker = MULTI_CHANNEL_ASSOCIATION_REPORT_MARKER							
Multi Channel NodeID 1							
Bit address 1	End Point 1						
...							
Multi Channel NodeID N							
Bit address N	End Point N						

Grouping Identifier (8 bits)

CC:008E.02.03.11.001 This field is used to advertise the actual association group. Grouping Identifiers MUST be assigned in a consecutive range starting from 1.

Max Nodes Supported (8 bits)

CC:008E.02.03.13.001 The maximum number of destinations supported by the advertised association group. Each destination MAY be a NodeID destination or an End Point destination.

Reports to Follow (8 bits)

CC:008E.02.03.11.002 The entire list destinations of the advertised association group may be too long for one command. This field MUST advertise how many report frames will follow this report.

NodeID (M bytes)

CC:008E.02.03.11.003 This field advertises a list of NodeID destinations of the advertised association group. The list of NodeIDs MUST be empty if there are no NodeID destinations configured for the advertised association group.

Marker (8 bits)

Refer to description under the Multi Channel Association Set command.

Multi Channel NodeID (N bytes)

The Multi Channel NodeID, Bit Address and End Point fields specify a list of End Points which are currently in the advertised association group.

The complete identification of an End Point requires a NodeID as well as an End Point identifier. The Multi Channel Association Report command MAY carry multiple copies of the same Multi Channel NodeID in combination with different End Point identifiers.

The list of Multi Channel NodeID and End Point identifiers MUST be empty if there are no End Point destinations configured for the advertised association group.

Bit address + End Point (N bytes)

These fields MUST be processed in combination with the Multi Channel NodeID field.

Refer to the Multi Channel Command Class for the actual encoding of bit addressed End Points.

4.25.7 Multi Channel Association Remove Command

This command is used to remove NodeID and End Point destinations from a given association group.

This command MUST manipulate the same list of NodeID destinations as the Association Remove Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL_ASSOCIATION							
Command = MULTI_CHANNEL_ASSOCIATION_REMOVE							
Grouping Identifier							
NodeID 1							
...							
NodeID M							
Marker = MULTI_CHANNEL_ASSOCIATION_REMOVE_MARKER							
Multi Channel NodeID 1							
Bit address 1	End Point 1						
...							
Multi Channel NodeID N							
Bit address N	End Point N						

Grouping Identifier (8 bits)

This field is used to specify from which association group the specified destinations should be removed.

Grouping Identifiers MUST be assigned in a consecutive range starting from 1.

A receiving node MUST ignore an unsupported Grouping Identifier; except for the value 0.

CC:008E.02.04.11.004 This field **MUST** be interpreted in combination with the NodeID and End Point fields.

(NodeID Destination)	NodeID	(M bytes)
(End Point Destination)	Multi Channel NodeID	(N bytes)
(End Point Destination)	End Point	(N bytes)

These fields specify the destinations that are to be removed.

CC:008E.02.04.11.005 The Grouping Identifier and these fields **MUST** be interpreted as follows.

“NodeID Destinations” denote the NodeID fields.

“End Point destinations” denote the combined Multi Channel NodeID and End Point fields found after the marker.

CC:008E.02.04.13.001 A receiving node **MAY** interpret the empty command (only Command Class and Command fields) as an instruction to Remove all NodeID destinations and End Point destinations from all association groups.

CC:008E.02.04.11.006 This field **MUST** be interpreted according to Table 17

Table 17, Multi Channel Association Remove, V2 :: Parameter interpretation

Grouping Identifier	Number of NodeID Destinations	Number of End Point Destinations	Interpretation
> 0	> 0	(don't care)	Remove NodeID destinations from association group. Then assess End Points (next line) – if any specified
> 0	(don't care)	> 0 *)	Remove End Point destinations from association group *).
> 0	= 0	= 0	Remove all NodeID destinations and End Point destinations from association group.
= 0	> 0	(don't care)	Remove NodeID destinations from all association groups. Then assess End Point destinations (next line) – if any specified
= 0	(don't care)	> 0 *)	Remove End Point destinations from all association groups *).
= 0	= 0	= 0	Remove all NodeID destinations and End Point destinations from all association groups.

CC:008E.02.04.11.007 *) The receiving node **MUST** treat the Bit Address flag and the End Point identifier as one scalar value.
 CC:008E.02.04.11.008 The receiving node **MUST** remove End Points if an exact match can be found, while it **MAY** ignore the removal request if an exact match cannot be found; even though there may exist associations for individual End Point destinations which are covered by the End Point bitmap address.

Marker (8 bits)

Refer to the description in 4.25.4 in the Multi Channel Association Set Command.

4.25.7.1 Examples

Remove specified NodeID from Group 3

Get association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_GET							
Grouping Identifier = 3							

Current association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REPORT							
Grouping Identifier = 3							
Max Nodes Supported = 5							
Reports to Follow = 0							
NodeID = 11							
NodeID = 12							
MULTI_CHANNEL_							
ASSOCIATION_REPORT_MARKER							
Multi Channel NodeID = 16							
End Point = 2							
Multi Channel NodeID = 16							
End Point = 3							
Multi Channel NodeID = 17							
End Point = 2							

Remove NodeID = 12 from Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REMOVE							
Grouping Identifier = 3							
NodeID = 12							

New association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REPORT							
Grouping Identifier = 3							
Max Nodes Supported = 5							
Reports to Follow = 0							
NodeID = 11							
MULTI_CHANNEL_							
ASSOCIATION_REPORT_MARKER							
Multi Channel NodeID = 16							
End Point = 2							
Multi Channel NodeID = 16							
End Point = 3							
Multi Channel NodeID = 17							
End Point = 2							

Remove specified End Point from Group 3Get association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_GET							
Grouping Identifier = 3							

Current association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REPORT							
Grouping Identifier = 3							
Max Nodes Supported = 5							
Reports to Follow = 0							
NodeID = 11							
NodeID = 12							
MULTI_CHANNEL_							
ASSOCIATION_REPORT_MARKER							
Multi Channel NodeID = 16							
End Point = 2							
Multi Channel NodeID = 16							
End Point = 3							
Multi Channel NodeID = 17							
End Point = 2							

Remove Multi Channel NodeID = 16,3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REMOVE							
Grouping Identifier = 3							
MULTI_CHANNEL_							
ASSOCIATION_REPORT_MARKER							
Multi Channel NodeID = 16							
End Point = 3							

New association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REPORT							
Grouping Identifier = 3							
Max Nodes Supported = 5							
Reports to Follow = 0							
NodeID = 11							
NodeID = 12							
MULTI_CHANNEL_							
ASSOCIATION_REPORT_MARKER							
Multi Channel NodeID = 16							
End Point = 2							
Multi Channel NodeID = 17							
End Point = 2							

Remove bit addressable End Points from Group 3

Get association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_GET							
Grouping Identifier = 3							

Current association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REPORT							
Grouping Identifier = 3							
Max Nodes Supported = 5							
Reports to Follow = 0							
NodeID = 11							
NodeID = 12							
MULTI_CHANNEL_							
ASSOCIATION_REPORT_MARKER							
Multi Channel NodeID = 16							
End Point = 134 (b1000 0110)							
Multi Channel NodeID = 17							
End Point = 2							

Remove Multi Channel NodeID = 16,2 and 16,3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REMOVE							
Grouping Identifier = 3							
MULTI_CHANNEL_							
ASSOCIATION_REPORT_MARKER							
Multi Channel NodeID = 16							
End Point = 134 (b1000 0110)							

Note: The Multi Channel Node MUST report end points as bit addressable in order to be removed in the same manners.

New association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REPORT							
Grouping Identifier = 3							
Max Nodes Supported = 5							
Reports to Follow = 0							
NodeID = 11							
NodeID = 12							
MULTI_CHANNEL_							
ASSOCIATION_REPORT_MARKER							
Multi Channel NodeID = 17							
End Point = 2							

Remove specified NodeID and End Point from Group 3Get association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_GET							
Grouping Identifier = 3							

Current association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REPORT							
Grouping Identifier = 3							
Max Nodes Supported = 5							
Reports to Follow = 0							
NodeID = 11							
NodeID = 12							
MULTI_CHANNEL_							
ASSOCIATION_REPORT_MARKER							
Multi Channel NodeID = 16							
End Point = 2							
Multi Channel NodeID = 16							
End Point = 3							
Multi Channel NodeID = 17							
End Point = 2							

Remove NodeID = 12 & Multi Channel NodeID = 16,3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REMOVE							
Grouping Identifier = 3							
NodeID = 12							
MULTI_CHANNEL_							
ASSOCIATION_REPORT_MARKER							
Multi Channel NodeID = 16							
End Point = 3							

New association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REPORT							
Grouping Identifier = 3							
Max Nodes Supported = 5							
Reports to Follow = 0							
NodeID = 11							
MULTI_CHANNEL_							
ASSOCIATION_REPORT_MARKER							
Multi Channel NodeID = 16							
End Point = 2							
Multi Channel NodeID = 17							
End Point = 2							

Remove all associations from Group 3Get association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_GET							
Grouping Identifier = 3							

Current association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REPORT							
Grouping Identifier = 3							
Max Nodes Supported = 5							
Reports to Follow = 0							
NodeID = 11							
NodeID = 12							
MULTI_CHANNEL_							
ASSOCIATION_REPORT_MARKER							
Multi Channel NodeID = 16							
End Point = 2							
Multi Channel NodeID = 16							
End Point = 3							
Multi Channel NodeID = 17							
End Point = 2							

Remove all NodeIDs from Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REMOVE							
Grouping Identifier = 3							

New association settings for Group 3

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REPORT							
Grouping Identifier = 3							
Max Nodes Supported = 5							
Reports to Follow = 0							

Remove specific NodeID from all Groups

7	6	5	4	3	2	1	0
COMMAND_CLASS_							
MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REMOVE							
Grouping Identifier = 0							
NodeID = 12							

Remove specific Multi Channel NodeID from all Groups

7	6	5	4	3	2	1	0
COMMAND_CLASS_ MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REMOVE							
Grouping Identifier = 0							
MULTI_CHANNEL_ ASSOCIATION_REPORT_MARKER							
Multi Channel NodeID = 16							
End Point = 3							

Remove all associations from all Groups

7	6	5	4	3	2	1	0
COMMAND_CLASS_ MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REMOVE							

or

7	6	5	4	3	2	1	0
COMMAND_CLASS_ MULTI_CHANNEL_ASSOCIATION							
MULTI_CHANNEL_ASSOCIATION_REMOVE							
Grouping Identifier = 0							

4.25.8 Multi Channel Association Supported Groupings Get Command

This command is used to request the number of association groups that this node supports.

CC:008E.02.05.11.001

The Multi Channel Association Supported Groupings Report Command MUST be returned in response to this command.

CC:008E.02.05.11.002

This command MUST NOT be issued via multicast addressing.

CC:008E.02.05.11.003

A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL_ASSOCIATION							
Command = MULTI_CHANNEL_ASSOCIATION_GROUPINGS_GET							

4.25.9 Multi Channel Association Supported Groupings Report Command

This command is used to advertise the maximum number of association groups implemented by this node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL_ASSOCIATION							
Command = MULTI_CHANNEL_ASSOCIATION_GROUPINGS_REPORT							
Supported Groupings							

Supported Groupings (8 bits)

This field is used to advertise the number of association groups that this node supports.

- CC:008E.02.06.11.001 Grouping Identifiers **MUST** be assigned in a consecutive range starting from 1.
- CC:008E.02.06.11.002 The value advertised by this field **MUST** be the total number of supported association groups; whether associations are managed via the Association Command Class or the Multi Channel Association Command Class.
- CC:008E.02.06.11.003 Each of these association groups **MUST** support NodeID destinations as well as End Point destinations.

4.26 Multi Channel Association Command Class, version 3

The Multi Channel Association Command Class is used to manage associations to Multi Channel End Point destinations as well as to NodeID destinations.

The following sections specify commands which were extended or added in version 3.

4.26.1 Compatibility considerations

The considerations of section 4.25.1 also apply to this version the following additions:

CC:008E.03.00.21.001

A device supporting this Command Class version **MUST** also

- support the Multi Channel Association Command Class, version 2
- support the (non-Multi Channel) Association Command Class, version 2

This version introduces support for the creation of an End Point Association to the Root Device of another device, e.g. a gateway, by allowing the destination End Point 0 as a valid value. This allows a gateway to create a single Lifeline association from a Multi Channel device and subsequently receive status messages or sensor reports from the Root Device as well as the End Points of that device. Refer to section 4.26.2.

CC:008E.03.00.21.002

The use of Multi Channel encapsulation **MUST** comply with Table 18.

Table 18, V3 Associations and the transmissions they may trigger

Source → Destination	Association type	Allowed transmissions
NodeID → NodeID (V2)	NodeID	Non-encapsulated
End Point → NodeID (V2)	NodeID	Non-encapsulated
NodeID → End Point (V2)	End Point	Encapsulated *) (src≥0, dst>0)
End Point → End Point (V2)	End Point	Encapsulated (src>0, dst>0)
End Point → Root Device (V3)	End Point	Encapsulated (src>0, dst=0)
Root Device → Root Device (V3)	End Point	Non-encapsulated **) or Encapsulated (src>0, dst=0)

*) The source End Point MAY be different than 0 if the associated group is an End Point Group mapped to the Root Device

**) Command must be sent non-encapsulated if both End Points are zero. However, the End Point association from a Root Device Lifeline group to a Root Device also allows a source End Point to send encapsulated commands to the Root Device Lifeline destination. Refer to 4.26.2.

4.26.2 Z-Wave Plus considerations

The considerations of version 2 also apply to this version.

Version 3 adds support for creation of an End Point Association to the Root Device of another device, e.g. a gateway.

CC:008E.03.00.11.001

A controlling device **MUST** verify that a device supports Multi Channel Association Command Class, Version 3 before creating an End Point Association to the Root Device of another device.

Multi Channel End Points may implement functionality relevant to a Lifeline destination, e.g. individual meter reports from a power strip. Multi Channel encapsulation allows the Lifeline destination to distinguish between apparently identical commands from different End Points.

As specified by the Multi Channel Command Class, a Root Device must not use Multi Channel encapsulation when communicating to another Root Device, i.e. if both End Points are zero. It does however still make sense to create an End Point Association from one Root Device to another Root Device, e.g. a gateway.

Consider, as an example, a two-channel temperature sensor. By creating an End Point association from the Lifeline Association Group to the Root Device of a gateway, the gateway may receive unsolicited Multi Channel encapsulated sensor reports from each of the sensor End Points as well as non-encapsulated battery status messages from the sensor Root Device.

- CC:008E.03.00.12.001 The Root Device of a Multi Channel device SHOULD implement support for the Multi Channel Association Command Class if any of the End Points provide association capabilities. If the Root Device implements such support, the Root Device MAY forward Multi Channel encapsulated commands to the association destination on behalf of End Points.
- CC:008E.03.00.13.001
- CC:008E.03.00.11.002 A NodeID association MUST NOT be created in response to a request for an End Point association to the Root Device (destination End Point 0).
- CC:008E.03.00.11.003 If a NodeID Association is created from the Root Device Lifeline Association Group, End Point commands MUST NOT be transmitted via the Lifeline Association Group of the root device. All commands originating from the Root Device (Tamper Alarm, Device Reset Locally, etc.) MUST be sent non-encapsulated to the Lifeline destination.
- CC:008E.03.00.11.004
- CC:008E.03.00.11.005 If a NodeID association is created, two End Points MUST NOT forward identical commands (e.g. Multilevel Sensor Report::Temperature) via the Lifeline group, as the lack of source End Point information will prevent a receiving application from distinguishing the individual commands from each other.
- CC:008E.03.00.11.006 If an End Point Association is created from the Lifeline group, End Points MUST send relevant Multi Channel encapsulated commands to the Lifeline group destination.
- CC:008E.03.00.13.002 A Multi Channel device MAY forward commands from multiple Multi Channel End Points to the Lifeline group destination.

4.26.3 Security considerations

The considerations of section 4.25.3 also apply to this version.

4.26.4 Multi Channel Association Set Command

This command is used to request that one or more destinations are added to a given association group.

CC:008E.03.01.13.001 The destinations MAY be a mix of NodeID destinations and End Point destinations.

CC:008E.03.01.12.001 The receiving node SHOULD add the specified destinations to the specified association group.
CC:008E.03.01.13.002 This command MAY be ignored if the association group is already full.

CC:008E.03.01.11.001 Routing slaves MUST have return routes assigned to all association destinations.

CC:008E.03.01.51.001 Unless the association destination is a gateway, a controlling node MUST NOT create an association if the association destination node does not support the controlling commands (Set/Get types) that the actual association group will be sending. The AGI Command Class MUST be used to probe the commands that a given association group will be sending.

CC:008E.03.01.52.001 A controlling node SHOULD NOT create an association if the source and destination nodes are bootstrapped with different security levels.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL_ASSOCIATION							
Command = MULTI_CHANNEL_ASSOCIATION_SET							
Grouping Identifier							
NodeID 1							
...							
NodeID M							
Marker = MULTI_CHANNEL_ASSOCIATION_SET_MARKER							
Multi Channel NodeID 1							
Bit address 1	End Point 1						
...							
Multi Channel NodeID N							
Bit address N	End Point N						

Grouping Identifier (8 bits)

This field is used to specify the actual association group.

CC:008E.03.01.11.002 A node receiving an unsupported Grouping Identifier MUST ignore this command.

NodeID (M bytes)

The NodeID fields specify a list of NodeID destinations that are to be added to the specified association group as a NodeID association.

CC:008E.03.01.11.003 A NodeID association created via this field MUST be identical to a NodeID association created with the (non-Multi Channel) Association Set command.

Marker (8 bits)

CC:008E.03.01.11.004 This field is used to indicate the end of NodeID destinations and the start of End Point destinations. The Marker field **MUST** be set to the value MULTI_CHANNEL_ASSOCIATION_SET_MARKER. The field **MAY** be omitted if no End Point destinations are specified.

Multi Channel NodeID (N bytes)

The Multi Channel NodeID, Bit Address and End Point fields specify a list of End Point destinations which are to be added to the specified association group as an End Point association.

CC:008E.03.01.13.003 The complete identification of an End Point destination requires a NodeID as well as an End Point identifier. This command **MAY** carry multiple copies of the same Multi Channel NodeID in combination with different End Point identifiers.

Bit address + End Point (N bytes)

CC:008E.03.01.11.005 These fields **MUST** be processed in combination with the Multi Channel NodeID field.

CC:008E.03.01.12.005 The receiving node **SHOULD** treat the Bit Address flag and the End Point identifier as one scalar value; thus creating only one association group entry. Using a single scalar value enables:

- Better utilization of association group capacity
- Simple transmission of command for a multi-End Point destination
- Well-defined removal of bit addressed End Points

Refer to the Multi Channel Command Class for the actual encoding of bit addressed End Points.

CC:008E.03.01.11.006 The 7-bit End Point value **MUST** be in the range 0..127.

4.27 Node Naming and Location Command Class, version 1

The Node Naming and Location Command Class is used to assign a name and a location text string to a supporting node.

4.27.1 Node Name Set Command

This command is used to set the name of the receiving node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_NODE_NAMING							
Command = NODE_NAMING_NODE_NAME_SET							
Reserved					Char. Presentation		
Node name char 1							
...							
Node name char N							

Node name char (N bytes)

This field is used to indicate the actual assigned name for the node.

This field MUST be encoded using the specified character representation in the Char.Presentation field.

The Node name string MUST NOT contain any appended termination characters.

The length of this field MUST be in the range 1..16 bytes. The length of this field MUST be determined from the frame length.

A node receiving this command with a Node name char longer than 16 bytes MUST ignore any byte/character following the 16th byte.

Devices using the Unicode UTF-16 characters representation can set Name strings using a maximum of 8 characters because each character is encoded with 2 bytes. In this case, the first byte MUST be the most significant byte.

Reserved

This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Char. Presentation (3 bits)

This field is used to indicate the character encoding for the Node name char field.
This field **MUST** comply with Table 19:

Table 19, Node Name Set::Char. Presentation encoding

Value	Description
0x00	Using standard ASCII codes, see Appendix A (values 128-255 are ignored)
0x01	Using standard and OEM Extended ASCII codes, see Appendix A
0x02	Unicode UTF-16

All other values are reserved and **MUST NOT** be used by a sending node. Reserved values **MUST** be ignored by a receiving node.

4.27.2 Node Name Get Command

This command is used to request the stored name from a node.

The Node Name Report Command **MUST** be returned in response to this command.

This command **MUST NOT** be issued via multicast addressing.
A receiving node **MUST NOT** return a response if this command is received via multicast addressing.
The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_NODE_NAMING							
Command = NODE_NAMING_NODE_NAME_GET							

4.27.3 Node Name Report Command

This command is used to advertise the name assigned to the sending node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_NODE_NAMING							
Command = NODE_NAMING_NODE_NAME_REPORT							
Reserved					Char. Presentation		
Node name char 1							
...							
Node name char N							

For fields' description, refer to 4.27.1 Node Name Set Command
A sending node **MUST** comply with fields' description from 4.27.1 Node Name Set Command

4.27.4 Node Location Set Command

The Node Location Set Command is used to set a location name in a node in a Z-Wave network.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_NODE_NAMING							
Command = NODE_NAMING_NODE_LOCATION_SET							
Reserved					Char. Presentation		
Node location char 1							
...							
Node location char N							

Node location char (N bytes)

This field is used to indicate the actual assigned location for the node.

This field **MUST** be encoded using the specified character representation in the Char. Presentation field.

The Node location string **MUST NOT** contain any appended termination characters.

The length of this field **MUST** be in the range 1..16 bytes. The length of this field **MUST** be determined from the frame length.

A node receiving this command with a Node location char longer than 16 bytes **MUST** ignore any byte/character following the 16th byte.

Devices using the Unicode UTF-16 characters representation can set Location strings using a maximum of 8 characters because each character is encoded with 2 bytes. In this case, the first byte **MUST** be the most significant byte.

Reserved

This field **MUST** be set to 0 by a sending node and **MUST** be ignored by a receiving node.

Char. Presentation (3 bits)

This field is used to indicate the character encoding for the Node location char field.
This field **MUST** comply with Table 19.

4.27.5 Node Location Get Command

This command is used to request the stored node location from a node.

The Node Location Report Command **MUST** be returned in response to this command.

This command **MUST NOT** be issued via multicast addressing.

A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_NODE_NAMING							
Command = NODE_NAMING_NODE_LOCATION_GET							

4.27.6 Node Location Report Command

This command is used to advertise the node location.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_NODE_NAMING							
Command = NODE_NAMING_NODE_LOCATION_REPORT							
Reserved					Char. Presentation		
Node location char 1							
...							
Node location char N							

For fields' description, refer to 4.27.4 Node Location Set Command

A sending node **MUST** comply with fields' description from 4.27.4 Node Location Set Command.

4.28 Remote Association Activation Command Class, version 1 [OBSOLETE]

THIS COMMAND CLASS VERSION HAS BEEN OBSOLETE

New implementations MUST NOT use this command class.

The Remote Association Activation Command Class is used to remote activations of Association grouping identifiers in other nodes.

Mandatory requirement: Both 'local' and 'remote' node MUST implement the Association Command Class as illustrated below. In addition the 'local' node MUST implement the Remote Association Configuration Command Class as 'Supported'.

The Remote Association Activation Command Class and the Remote Association Configuration Command Class are additions to the functionality to the existing Association Command Class.

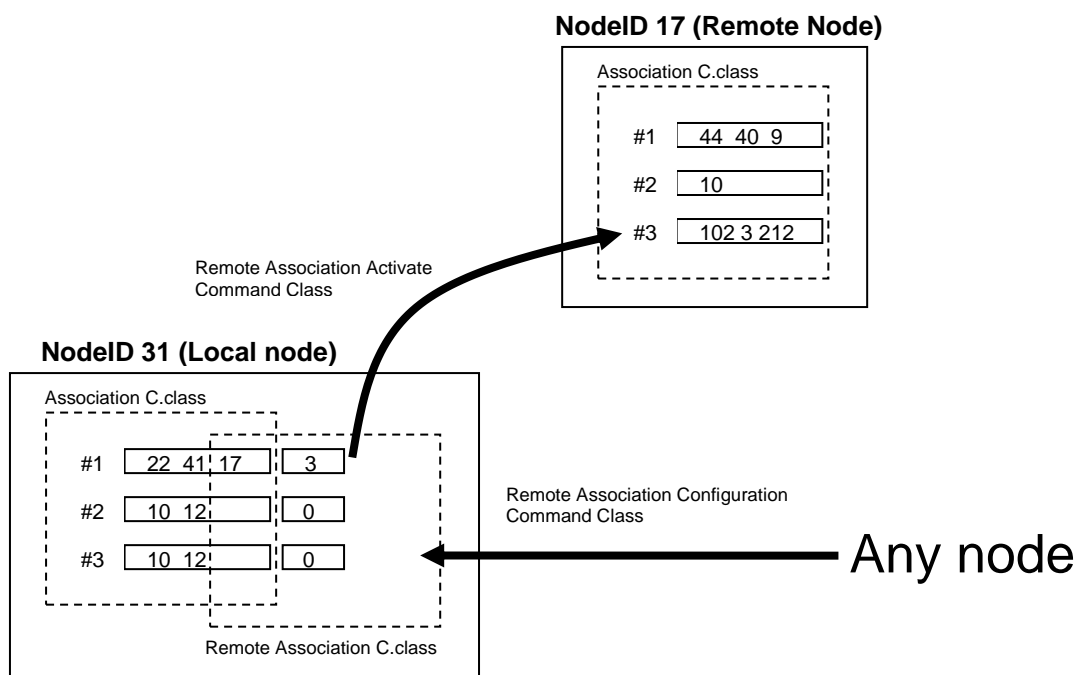


Figure 8, Remote Association Activation Command Class

The Remote Association Configuration Command Class enables a 1st node (any node) to configure a 2nd node (local node) to issue a Remote Association Activate Command to a 3rd node (remote node), which instructs the 3rd node to activate one of its locally stored association group identifiers as defined by the Association Command Class.

4.28.1 Remote Association Activate Command

The Remote Association Activate Command is used to instruct a 'remote' node to activate one of its locally stored association group identifiers as defined by the Association Command Class. This will subsequently generate a number of Commands being issued from the 'remote node to the NodeIDs associated to the grouping identifier

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_REMOTE_ASSOCIATION_ACTIVATE							
Command = REMOTE_ASSOCIATION_ACTIVATE							
Grouping identifier							

Grouping identifier (8 bits)

This group identifier used to specify the grouping identifier on the remote node

4.29 Remote Association Configuration Command Class, version 1 [OBSOLETE]

THIS COMMAND CLASS VERSION HAS BEEN OBSOLETE

New implementations MUST NOT use this command class.

The Remote Association Configuration Command Class is used to configuration of the Remote Association Activation Command Class.

Mandatory requirement: Both 'local' and 'remote' node MUST implement the Association Command Class as 'supported'. In addition the 'local' node MUST implement the Remote Association Configuration Command Class as 'supported' and the node used to make the configuration ('any node' in the description below) MUST implement it as 'controlled'.

The Remote Association Configuration Command Class is an addition to the functionality of the existing Association Command Class.

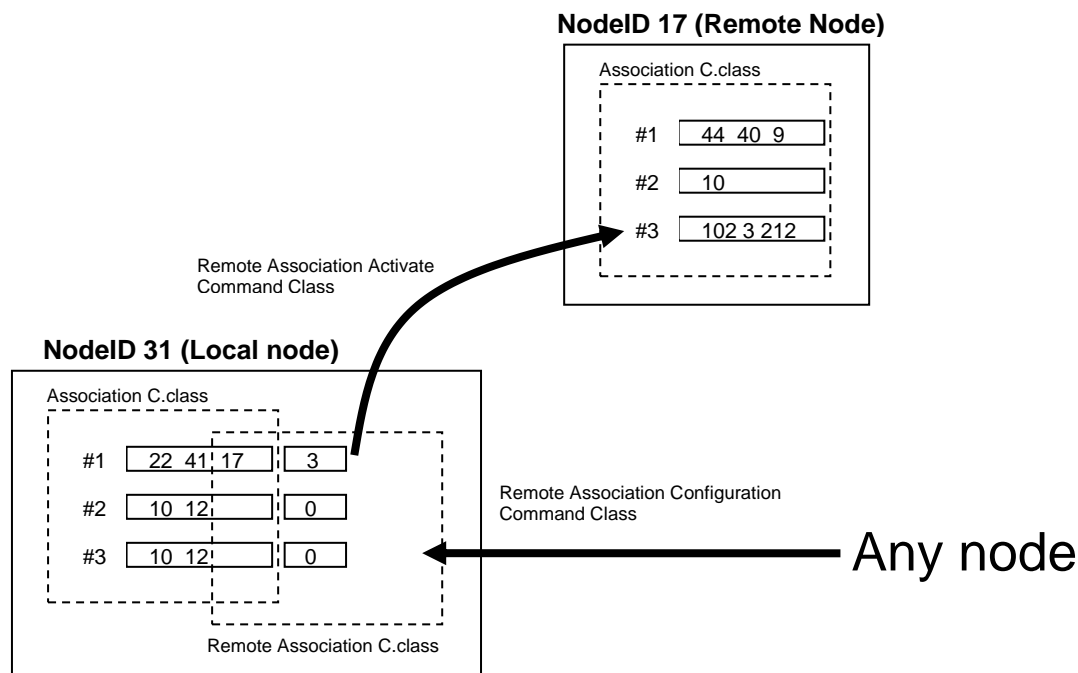


Figure 9, Remote Association Configuration Command Class

The Remote Association Configuration Command Class enables a 1st node (Any node) to configure a 2nd node (Local node) to issue a Remote Association Activate Command to a 3rd node (Remote node), which instructs the 3rd node to activate one of its locally stored association group identifiers as defined by its Association Command Class.

4.29.1 Remote Association Configuration Set Command

The Remote Association Configuration Set Command links two nodes' 'Association Command Class' defined grouping identifiers together. It allows one node (local node) to use its grouping identifiers to control a second node's (remote node) grouping identifiers, using the Remote Association Activation Command Class.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_REMOTE_ASSOCIATION							
Command = REMOTE_ASSOCIATION_CONFIGURATION_SET							
Local Grouping identifier							
Remote NodeID							
Remote Grouping identifier							

Local Grouping identifier (8 bits)

Like in the Association Command Class, the Local Grouping identifier is used to specify the grouping identifier on the local node.

A Local grouping identifier = 0x0 will erase all links between local and remote grouping identifiers.

Remote NodeID (8 bits)

This NodeID used to specify the Node, which should receive the Remote Association Activate Command.

A NodeID = 0x0 will remove a link between the specified local grouping identifier and a remote grouping identifier.

Remote Grouping identifier (8 bits)

This group identifier used to specify the grouping identifier on the remote node.

4.29.2 Remote Association Configuration Get Command

The Remote Association Configuration Get Command is used to request the link between a Local Grouping identifier and a Remote Grouping identifier on a node.

The Remote Association Configuration Report Command **MUST** be returned in response to this command.

This command **MUST NOT** be issued via multicast addressing.

A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_REMOTE_ASSOCIATION							
Command = REMOTE_ASSOCIATION_CONFIGURATION_GET							
Local Grouping identifier							

Local Grouping identifier (8 bits)

Like in the Association Command Class, the Local Grouping identifier is used to specify the grouping identifier on the local node.

4.29.3 Remote Association Configuration Report Command

The Remote Association Configuration Report Command returns the remote NodeID and the grouping identifier.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_REMOTE_ASSOCIATION							
Command = REMOTE_ASSOCIATION_CONFIGURATION_REPORT							
Local Grouping identifier							
Remote NodeID							
Remote Grouping identifier							

Local Grouping identifier (8 bits)

This group identifier used to specify the grouping identifier on the local node

Remote NodeID (8 bits)

This NodeID used to specify the remote node that the Remote Association Activate Command is sent to. If no link is established between the Local Grouping Identifier and a Remote Grouping Identifier, the Remote NodeID will return zero (0x0)

Remote Grouping identifier (8 bits)

This Remote grouping identifier used to specify the grouping identifier on the remote node that should be activated.

4.30 Version Command Class, version 1

The Version Command Class may be used to obtain the Z-Wave library type, the Z-Wave protocol version used by the application, the individual command class versions used by the application and the vendor specific application version from a Z-Wave enabled device.

4.30.1 Compatibility Considerations

- CC:0086.01.00.21.001 In a Multi Channel device, the Version Command Class **MUST** be supported by the Root Device, while the Version Command Class **SHOULD NOT** be supported by individual End Points.
- CC:0086.01.00.22.001
- CC:0086.01.00.21.002 There may be cases where a given Command Class is not implemented by the Root Device of a Multi Channel device. However, the Root Device **MUST** respond to Version requests for any Command Class implemented by the Multi Channel device; also in cases where the actual Command Class is only provided by an End Point.
- CC:0086.01.00.21.003 A node supporting a Command Class having a version higher than 1 **MUST** support the Version Command Class to be able to identify the supported version. If a node does not support the Version Command Class at its highest security level, it may be assumed that all command classes implement version 1.

4.30.2 Security Considerations

- The Version Command Class provides information about the device implementation. This information can potentially be used by an attacker to find vulnerabilities.
- CC:0086.01.00.41.002 If a node supports this Command Class and does not support the Security 2 Command Class, it **MUST** comply with Table 20.

Table 20, Version Command Class support

	After Non-Secure inclusion	After Secure inclusion
Non-Secure or Less-Secure communication	The node MUST support the Version Command Class and advertise it in the NIF..	The node MUST NOT support the Version Command Class and MUST NOT advertise it in the NIF or Security Commands Supported Report commands.
Secure communication at the highest Security Class	N/A	The node MUST support the Version Command Class and advertise it in the Security Commands Supported Report commands.

- CC:0086.01.00.41.003 A node supporting the Security 2 Command Class **MUST** support this Command Class according to the Device Type specification [10] requirements.

4.30.3 Version Get Command

The Version Get Command is used to request the library type, protocol version and application version from a device that supports the Version Command Class. Refer to [1].

CC:0086.01.11.11.001

The Version Report Command **MUST** be returned in response to this command.

CC:0086.01.11.11.002

This command **MUST NOT** be issued via multicast addressing.

CC:0086.01.11.11.003

A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_VERSION							
Command = VERSION_GET							

4.30.4 Version Report Command

The Version Report Command is be used to advertise the library type, protocol version and application version from a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_VERSION							
Command = VERSION_REPORT							
Z-Wave Library Type							
Z-Wave Protocol Version							
Z-Wave Protocol Sub Version							
Application Version							
Application Sub Version							

Z-Wave Library Type (8 bits)

CC:0086.01.12.11.001

This field **MUST** carry the Z-Wave Protocol Library Type.

CC:0086.01.12.11.002

The value **MUST** comply with Table 21.

Table 21, Z-Wave Library Type

Library Type	Description	Version
0x00	N/A	-
0x01	Static Controller	1
0x02	Controller	1
0x03	Enhanced Slave	1
0x04	Slave	1
0x05	Installer	1
0x06	Routing Slave	1
0x07	Bridge Controller	1
0x08	Device Under Test (DUT)	1
0x09	N/A	1
0x0A	AV Remote	1
0x0B	AV Device	1

CC:0086.01.12.11.003 Values marked as not applicable (N/A) MUST be ignored by a receiving node.

CC:0086.01.12.11.004 All other values are reserved and MUST NOT be used by a sending node. Reserved values MUST be ignored by a receiving node.

Z-Wave Protocol Version & Z-Wave Protocol Sub Version

These fields advertise information specific to Software Development Kits (SDK) provided by Silicon Labs . Refer to Silicon Labs SDK documentation.

Application Version (8 bits)

Returns the Application Version and can have values in the range 0 to 255. The manufacturer assigns the Application Version.

Application Sub Version (8 bits)

Returns the Application Sub Version and can have values in the range 0 to 255. The manufacturer assigns the Application Sub Version.

4.30.5 Version Command Class Get Command

The Version Command Class Get Command is used to request the individual command class versions from a device.

CC:0086.01.13.11.001

The Version Command Class Report Command **MUST** be returned in response to this command.

Only versions from the command classes shown in the NIF, in the Security Command Supported Report or in the Multi Channel Capability Report can be requested.

It is not possible to get a version number for the Generic and Specific Device Classes.

CC:0086.01.13.11.002

This command **MUST NOT** be issued via multicast addressing.

CC:0086.01.13.11.003

A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_VERSION							
Command = VERSION_COMMAND_CLASS_GET							
Requested Command Class							

Requested Command Class (8 bits)

This field is used to indicate the Command Class identifier that is being requested.

4.30.6 Version Command Class Report Command

The Version Command Class Report Command is used to report the individual command class versions from a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_VERSION							
Command = VERSION_COMMAND_CLASS_REPORT							
Requested Command Class							
Command Class Version							

Requested Command Class (8 bits)

The Requested Command Class field specifies what Command Class the returned version belongs to.

Command Class Version (8 bits)

CC:0086.01.14.11.001

If the requested Command Class is supported, this field is used to return the Command Class Version. This field **MUST** be in the range 1..255. It starts with 1 and is incremented every time a new version of the Command Class is released.

CC:0086.01.14.11.002

If the requested Command Class is not supported or controlled, a responding node **MUST** set the Command Class Version field to 0x00.

4.31 Version Command Class, version 2

The Version Command Class, version 2 is extended to report the version of various firmware images such as a host processor firmware, etc. in addition to the firmware image running in the Z-Wave chip.

As an example, one may construct a product comprising a Z-Wave chip and a secondary host processor that maintains a security certificate. With Firmware Update Meta Data Command Class, version 3 the Z-Wave chip, the host processor and the security certificate may all be updated via individual firmware IDs. Version 2 of the Version Command Class (this Command Class) allows a controlling node to request the corresponding version information for each firmware ID.

Commands not mentioned here remain the same as specified for Version Command Class, version 1.

4.31.1 Version Report Command

This command is used to report the library type, protocol version and application version from a node.

Version 2 of this command renames the fields Application Version and Application Sub Version to Firmware 0 Version and Firmware 0 Sub Version. The use remains the same.

- CC:0086.02.12.11.001 A node **MUST** advertise the version of all firmware images which can be updated via the Firmware Update Command Class.
- CC:0086.02.12.11.002 A one-chip system **MUST** comply with the following:
 - The Firmware 0 Version **MUST** reflect the complete firmware implementing the Z-Wave protocol stack as well as the Z-Wave application.
- CC:0086.02.12.11.003 A multi-processor system **MUST** comply with the following:
 - The Firmware 0 Version **MUST** reflect the firmware implementing the Z-Wave protocol stack and the inter-chip interface module that enables the Z-Wave application to run in the host processor. Another firmware number (e.g. Firmware 1) version **MUST** reflect the Z-Wave application that runs in the host processor. Any firmware number larger than 0 **MAY** be used for this purpose.
- CC:0086.02.12.11.004
- CC:0086.02.12.11.005
- CC:0086.02.12.11.006
- CC:0086.02.12.13.001
- CC:0086.02.12.13.002 A node **MAY** advertise the version of additional images hosted by the node; even if such images cannot be updated via the Firmware Update Command Class.

An illustration is given in Figure 10 and Figure 11.

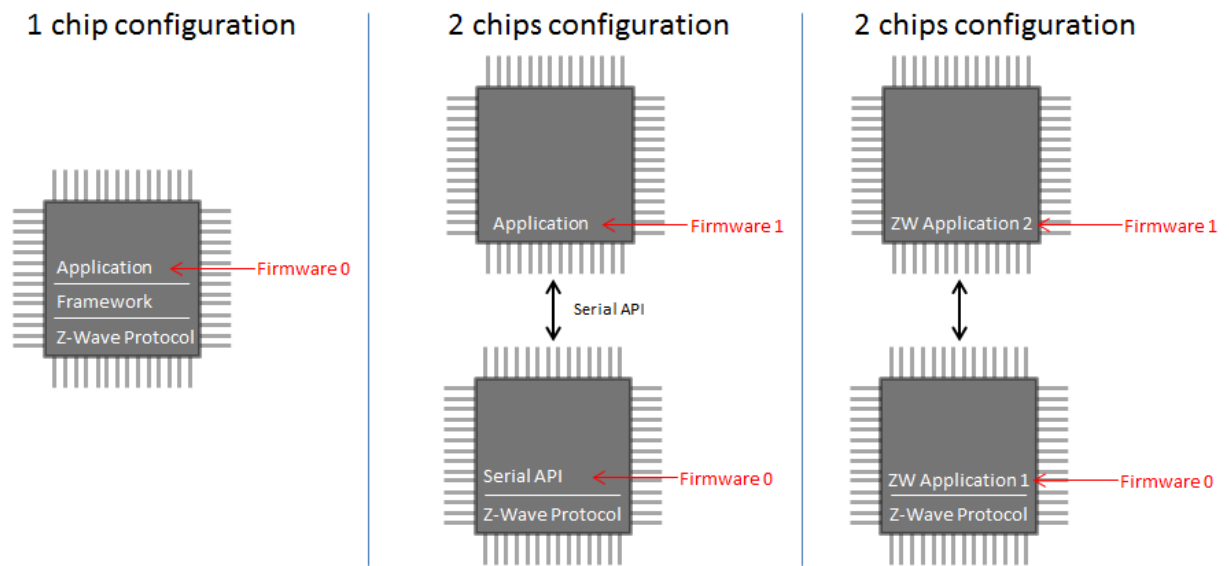


Figure 10, Version Report::Firmware numbering

chip without Z-Wave functionality

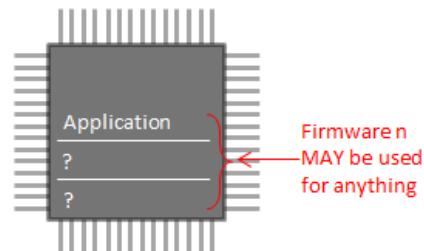


Figure 11, Version Report::Firmware numbering (2)

Further, version 2 of the Version Report command introduces a Hardware Version field as well as new version fields for each additional firmware image implemented by the actual device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_VERSION							
Command = VERSION_REPORT							
Z-Wave Protocol Library Type							
Z-Wave Protocol Version							
Z-Wave Protocol Sub Version							
Firmware 0 Version							
Firmware 0 Sub Version							
Hardware Version							
Number of firmware targets							
Firmware 1 Version							
Firmware 1 Sub Version							
...							
Firmware N Version							
Firmware N Sub Version							

Z-Wave Library Type (8 bits)

For a description, refer to section 4.30.4.

Z-Wave Protocol Version and Z-Wave Protocol Sub Version

These fields advertise information specific to Software Development Kits (SDK) provided by Silicon Labs . Refer to Silicon Labs SDK documentation.

Firmware 0 Version (8 bits)

Returns the Firmware 0 Version. Firmware 0 is dedicated to the Z-Wave chip firmware. The manufacturer MUST assign a version number. Previously called Application Version.

CC:0086.02.12.11.007

Firmware 0 Sub Version (8 bits)

Returns the Firmware 0 Sub Version. Firmware 0 is dedicated to the Z-Wave chip firmware. The manufacturer MUST assign a sub version number. Previously called Application Sub Version.

CC:0086.02.12.11.008

Hardware Version (8 bits)

- CC:0086.02.12.11.009 The Hardware Version field MUST report a value which is unique to this particular version of the product.
CC:0086.02.12.11.00A The value MUST be updated to a new value every time the hardware is modified for a given product. The
CC:0086.02.12.12.001 value 0x00 SHOULD NOT be used for the Hardware Version.
- CC:0086.02.12.11.00B It MUST be possible to uniquely determine the hardware characteristics from the Hardware Version field in combination with the Manufacturer ID, Product Type ID and Product ID fields of Manufacturer Specific Info Report of the Manufacturer Specific Command Class.
This information allows a user to pick a firmware image version that is guaranteed to work with this particular version of the product.

Note that the Hardware Version field is intended for the hardware version of the entire product, not just the version of the Z-Wave radio chip.

Number of Firmware Targets (8 bits)

- CC:0086.02.12.11.00C The Number of Firmware Targets field MUST report the number of firmware Version + Sub Version fields following this field. The Firmware 0 Version fields are not included.
CC:0086.02.12.11.00D The field MUST be zero if the device only implements a Firmware 0 target, the Z-Wave chip.

Firmware Version (N * 8 bits)

- CC:0086.02.12.11.00E Returns the Firmware n Version. The manufacturer MUST assign a unique Firmware n Version.

Firmware Sub Version (N * 8 bits)

- CC:0086.02.12.11.00F Returns the Firmware n Sub Version. The manufacturer MUST assign a unique Firmware n Sub Version.

4.32 Version Command Class, version 3

The Version Command Class, version 3 allows supporting nodes to advertise capabilities related to the Version Command Class and optionally provide a detailed list of information regarding implementation on the Z-Wave chip.

4.32.1 Compatibility considerations

The Version Command Class, version 3 is backwards compatible with the Version Command Class, version 2. A node supporting the Version Command Class, version 3 **MUST** also support the Version Command Class, version 2.

All commands and fields not described in this version remain unchanged from version 2.

The Version Command Class, version 3 introduces the possibility for a node to advertise its supported Version Command Class capabilities with the following commands:

- Version Capabilities Get Command
- Version Capabilities Report Command

Supporting nodes **MUST** advertise support for commands present in previous versions of this Command Class for backwards compatibility.

The following commands are optionally implemented by supporting nodes:

- Version Z-Wave Software Get Command
- Version Z-Wave Software Report Command

The Version Z-Wave Software Get/Report commands are used to provide detailed information on the actual software running on the Z-Wave radio SoC.

4.32.2 Version Capabilities Get Command

This command is used to request which version commands are supported by a node.

The Version Capabilities Report Command **MUST** be returned in response to this command.

This command **MUST NOT** be issued via multicast addressing.

A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_VERSION							
Command = VERSION_CAPABILITIES_GET							

4.32.3 Version Capabilities Report Command

This command is used to advertise the version commands supported by the sending node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_VERSION							
Command = VERSION_CAPABILITIES_REPORT							
Reserved					ZWS	CC	V

V (Version) (1 bit)

This field is used to advertise support for the version information queried with the Version Get Command. This field **MUST** be set to 1.

CC (Command Class) (1 bit)

This field is used to advertise support for the Command Class version information queried with the Version Command Class Get Command. This field **MUST** be set to 1.

ZWS (Z-Wave Software) (1 bit)

This field is used to advertise support for the detailed Z-Wave software version information queried with the Version Z-Wave Software Get Command.

The value 1 **MUST** indicate that the sending node supports the Version Z-Wave Software Get Command. The value 0 **MUST** indicate that the sending node does not support the Version Z-Wave Software Get Command.

4.32.4 Version Z-Wave Software Get Command

This command is used to request the detailed Z-Wave chip software version information of a node.

A node advertising no support for detailed Z-Wave software version information in the Version Capabilities Report Command **MUST** ignore this command.

The Version Z-Wave Software Report Command **MUST** be returned in response to this command unless it is to be ignored.

This command **MUST NOT** be issued via multicast addressing.

A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_VERSION							
Command = VERSION_ZWAVE_SOFTWARE_GET							

4.32.5 Version Z-Wave Software Report Command

This command is used to advertise the detailed Z-Wave chip software version information of a node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_VERSION							
Command = VERSION_ZWAVE_SOFTWARE_REPORT							
SDK version 1 (MSB)							
SDK version 2							
SDK version 3 (LSB)							
Application Framework API Version 1 (MSB)							
Application Framework API Version 2							
Application Framework API Version 3 (LSB)							
Application Framework Build Number 1 (MSB)							
Application Framework Build Number 2 (LSB)							
Host Interface Version 1 (MSB)							
Host Interface Version 2							
Host Interface Version 3 (LSB)							
Host Interface Build Number 1 (MSB)							
Host Interface Build Number 2 (LSB)							
Z-Wave Protocol Version 1 (MSB)							
Z-Wave Protocol Version 2							
Z-Wave Protocol Version 3 (LSB)							
Z-Wave Protocol Build Number 1 (MSB)							
Z-Wave Protocol Build Number 2 (LSB)							
Application Version 1 (MSB)							
Application Version 2							
Application Version 3 (LSB)							
Application Build Number 1 (MSB)							
Application Build Number 2 (LSB)							

SDK version (24 bits)

This field is used to advertise the SDK version used for building the Z-Wave chip software components for the node.

The value 0 MUST indicate that this field is unused. Other values MUST indicate the SDK version.

Application Framework API Version (24 bits)

This field is used to advertise the application framework API version used by the node.

The value 0 MUST indicate that this field is unused. Other values MUST indicate the application framework API version.

Application Framework Build Number (16 bits)

This field is used to advertise the application framework build number running on the node.

The value 0 MUST indicate that this field is unused. Other values MUST indicate the application framework build number.

This field MUST be set to 0 if the Application Framework API Version field is set to 0.

Host Interface Version (24 bits)

This field is used to advertise the version of the API exposed to a host CPU.

The value 0 MUST indicate that this field is unused. Other values MUST indicate the actual API version. This field MUST be set to 0 by a node running on a single chip.

Host Interface Build Number (16 bits)

This field is used to advertise the build number of the API software exposed to a host CPU.

The value 0 MUST indicate that this field is unused. Other values MUST indicate the actual API build number.

This field MUST be set to 0 if the Host Interface Version field is set to 0.

Z-Wave Protocol Version (24 bits)

This field is used to advertise the Z-Wave protocol version used by the node.

This field MUST be set to the same value as the Z-Wave Protocol Version and Z-Wave Protocol Sub Version fields present in the Version Report Command.

Byte 1 of this field MUST be set to the same value as the Z-Wave Protocol Version field present in the Version Report Command.

Byte 2 of this field MUST be set to the same value as the Z-Wave Protocol Sub Version field present in the Version Report Command.

Z-Wave Protocol Build Number (16 bits)

This field is used to advertise the actual build number of the Z-Wave protocol software used by the node.

The value 0 MUST indicate that this field is unused. Other values MUST indicate the Z-Wave protocol build number.

Application Version (24 bits)

This field is used to advertise the version of application software used by the node on its Z-Wave chip.

The value 0 MUST indicate that this field is unused. Other values MUST indicate the application version.

This field MUST be set to 0 by a node if its application is running on a host CPU.

If a supporting node's application is running on the Z-Wave chip:

- Byte 1 of this field MUST be set to the same value as the Z-Wave Application field present in the Version Report Command.
- Byte 2 of this field MUST be set to the same value as the Z-Wave Application Sub Version field present in the Version Report Command.

Application Build Number (16 bits)

This field is used to advertise the actual build of the application software used by the node on its Z-Wave chip.

The value 0 MUST indicate that this field is unused. Other values MUST indicate the application build number.

4.32.5.1 Version fields

Fields indicating version numbers are composed of 3 bytes. Version fields MUST be used and interpreted in the following manner:

If used, Byte 1 of a version field MUST represent the major version digit.

If used, Byte 2 of a version field MUST represent the minor version digit.

If used, Byte 3 of a version field MUST represent the patch version digit.

For example, if the node used Z-Wave SDK version 6.51.09, it MUST set the SDK version field to 0x063309. Unused bytes MUST be set to 0.

4.33 Wake Up Command Class, version 1

The Wake Up Command Class allows a battery-powered device to notify another device (always listening), that it is awake and ready to receive any queued commands.

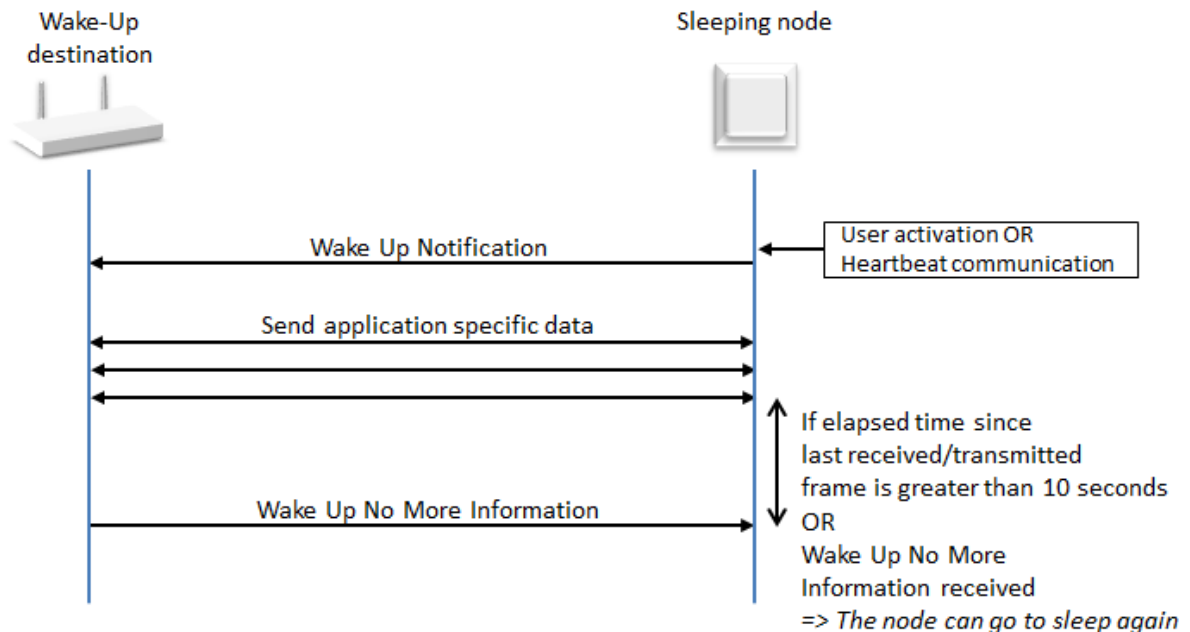


Figure 12, Wake Up sequence

CC:0084.01.00.12.001

This Command Class SHOULD be implemented by battery powered devices. Wake Up Notification commands SHOULD be handled immediately by the destination node in order to minimize battery consumption.

4.33.1 Terminology

The Wake Up period is defined as the period during which a Wake Up node is supposed to be awake after issuing a Wake Up Notification Command to its Wake Up destination.

The Wake Up period starts when the supporting node issues a Wake Up Notification and it ends either 10 seconds after the last received/transmitted frame or at the reception of a Wake Up No More Information Command by the Wake Up destination.

4.33.2 Interoperability considerations

CC:0084.01.00.31.001

The Wake Up Notification Command Class MUST NOT be covered by any association group. This also means that the Wake Up Notification Command Class MUST NOT be covered by the Lifeline association group.

CC:0084.01.00.32.001

A node sending periodical or triggered unsolicited commands at the same time it wakes up SHOULD send the unsolicited commands before the Wake Up Notification Command.

CC:0084.01.00.32.002

A supporting node SHOULD NOT send unsolicited commands during the Wake Up period.

4.33.3 Wake Up Interval Set Command

This command is used to configure the Wake Up interval and destination of a node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_WAKE_UP							
Command = WAKE_UP_INTERVAL_SET							
Seconds 1 (MSB)							
Seconds 2							
Seconds 3 (LSB)							
NodeID							

Seconds (24 bits)

This field is used to specify the time in seconds between Wake Up periods for the receiving node.

CC:0084.01.04.11.001 The first byte MUST be the most significant byte.

CC:0084.01.04.11.002 Values in the range 1..16777215 MUST indicate the time between Wake Up periods.
The value 0 MUST indicate that the receiving node MUST Wake Up when initiated by an event determined by the application e.g. a pushbutton activation.

NodeID (8 bits)

This field is used to specify the Wake Up destination NodeID.

CC:0084.01.04.11.003 A receiving node MUST assume the indicated NodeID as Wake Up destination. i.e. the Wake Up Notification Command MUST be sent to the NodeID specified in this field.

4.33.4 Wake Up Interval Get Command

This command is used to request the Wake Up Interval and destination of a node.

CC:0084.01.05.11.001 The Wake Up Interval Report Command MUST be returned in response to this command.

CC:0084.01.05.11.002 This command MUST NOT be issued via multicast addressing.
CC:0084.01.05.11.003 A receiving node MUST NOT return a response if this command is received via multicast addressing.
The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_WAKE_UP							
Command = WAKE_UP_INTERVAL_GET							

4.33.5 Wake Up Interval Report Command

This command is used to advertise the current Wake Up interval and destination.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_WAKE_UP							
Command = WAKE_UP_INTERVAL_REPORT							
Seconds 1 (MSB)							
Seconds 2							
Seconds 3 (LSB)							
NodeID							

Seconds (24 bits)

This field is used to advertise the time in seconds between Wake Up periods at the sending node.

CC:0084.01.06.11.001

The first byte **MUST** be the most significant byte.

CC:0084.01.06.11.002

Values in the range 1..16777215 **MUST** indicate the time between Wake Up periods. The value 0 **MUST** indicate that the receiving node **MUST** Wake Up when initiated by an event determined by the application e.g. a pushbutton activation.

NodeID (8 bits)

This field is used to advertise the Wake Up destination NodeID configured at the sending node.

CC:0084.01.06.11.003

The sending node **MUST** send the Wake Up Notification Commands to the advertised NodeID in this field.

4.33.6 Wake Up Notification Command

This command allows a node to notify its Wake Up destination that it is awake.

CC:0084.01.07.12.001

CC:0084.01.07.11.001

The sending node **SHOULD** start a timer allowing it power down again in case no Wake Up No More Information Command is received. If implemented, the timer **MUST** comply with the timing requirements defined in the Z-Wave Plus Device Types Specification [9].

CC:0084.01.07.13.001

CC:0084.01.07.11.002

CC:0084.01.07.12.002

A node **MAY** send this command as broadcast if no Wake Up destination NodeID is configured. A receiving node **MUST NOT** return a Wake Up No More Information Command in response to this command issued via broadcast. Upon receiving this command via broadcast, a receiving node **SHOULD** configure a relevant Wake Up destination issuing a Wake Up Interval Set Command to the node that issued this command via broadcast.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_WAKE_UP							
Command = WAKE_UP_NOTIFICATION							

4.33.7 Wake Up No More Information Command

CC:0084.01.08.13.001 This command is used to notify a supporting node that it MAY return to sleep to minimize power consumption.

CC:0084.01.08.12.001 A node receiving the Wake Up No More Information Command SHOULD return a MAC layer Ack frame for the Wake Up No More Information Command before returning to sleep. Not acknowledging the command will cause a number of retransmissions.

CC:0084.01.08.12.002 It is RECOMMENDED that the handling of the Wake Up No More Information Command and the associated time window for MAC layer acknowledgement is left to the protocol layer in order to simplify application design.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_WAKE_UP							
Command = WAKE_UP_NO_MORE_INFORMATION							

4.34 Wake Up Command Class, version 2

The Wake Up Command Class version 2 enables read back of the Wake Up Interval capabilities in a node.

4.34.1 Compatibility considerations

CC:0084.02.00.21.001

A node supporting Wake Up Command Class, version 2 **MUST** also support Wake Up Command Class, version 1.

All commands not mentioned in this version remain unchanged from the Wake Up Command Class, version 1.

CC:0084.02.00.22.001

A version 2 supporting node may receive a Wake Up Interval Set outside the range of supported intervals from a version 1 controlling node. A version 2 supporting node **SHOULD NOT** ignore a Wake Up Interval Set with an invalid time period and set the Wake Up Interval to a supported value.

4.34.2 Wake Up Interval Set Command

The Wake Up Interval Set Command is used to configure the wake up interval of a device and the NodeID of the device receiving the Wake Up Notification Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_WAKE_UP							
Command = WAKE_UP_INTERVAL_SET							
Seconds 1 (MSB)							
Seconds 2							
Seconds 3 (LSB)							
NodeID							

All fields not described below remain unchanged from version 1

Seconds (24 bits)

This field is used to specify the time in seconds between Wake Up periods for the receiving node.

- CC:0084.02.04.11.001 The first byte **MUST** be the most significant byte.
- CC:0084.02.04.11.002 Values in the range 1..16777215 **MUST** indicate the time between Wake Up periods. The value 0 **MUST** indicate that the receiving node **MUST** Wake Up when initiated by an event determined by the application e.g. a pushbutton activation.
- CC:0084.02.04.11.003 A receiving node **MUST** accept any interval value within the interval limits advertised by the Wake Up Interval Capabilities Report Command.
- CC:0084.02.04.12.001 A receiving node **SHOULD NOT** ignore this command if this field contains a non-valid value and it **SHOULD** apply the following:
- If the received value is higher than the minimum valid value (advertised in the Wake Up Interval Capabilities Report Command), the receiving node **SHOULD** round down the value to the next valid value but **SHOULD NOT** set it to 0 if 0 is the minimum value.
 - If the value specified in this field is lower than the minimum valid value, the receiving node **SHOULD** apply the minimum valid value.
- CC:0084.02.04.12.002
- CC:0084.02.04.12.003
- CC:0084.02.04.12.004 A receiving node **SHOULD** accept the value 0, even if the Minimum Wake Up Interval advertised by the Wake Up Interval Capabilities Report Command is larger than 0. If accepted, the value 0 **MUST** disable the timer-based transmission of Wake Up Notification Commands.
- CC:0084.02.04.12.005 The node **SHOULD** be able to issue a Wake Up Notification in response to some local activation, e.g. a button press.

4.34.3 Wake Up Interval Capabilities Get Command

This command is used to request the Wake Up Interval capabilities of a node.

- CC:0084.02.09.11.001 The Wake Up Interval Capabilities Report Command **MUST** be returned in response to this command.
- CC:0084.02.09.11.002 This command **MUST NOT** be issued via multicast addressing.
- CC:0084.02.09.11.003 A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_WAKE_UP							
Command = WAKE_UP_INTERVAL_CAPABILITIES_GET							

4.34.4 Wake Up Interval Capabilities Report Command

This command is used to advertise the Wake Up Interval capabilities of a node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_WAKE_UP							
Command = WAKE_UP_INTERVAL_CAPABILITIES_REPORT							
Minimum Wake Up Interval Seconds 1 (MSB)							
Minimum Wake Up Interval Seconds 2							
Minimum Wake Up Interval Seconds 3 (LSB)							
Maximum Wake Up Interval Seconds 1 (MSB)							
Maximum Wake Up Interval Seconds 2							
Maximum Wake Up Interval Seconds 3 (LSB)							
Default Wake Up Interval Seconds 1 (MSB)							
Default Wake Up Interval Seconds 2							
Default Wake Up Interval Seconds 3 (LSB)							
Wake Up Interval Step Seconds 1 (MSB)							
Wake Up Interval Step Seconds 2							
Wake Up Interval Step Seconds 3 (LSB)							

CC:0084.02.0A.11.001 Byte 1 MUST be the most significant byte for all fields in this command.

Minimum Wake Up Interval Seconds (24 bits)

CC:0084.02.0A.11.002 This field is used to advertise the minimum Wake Up Interval supported by the sending node. The field MUST be in the range 0..16777215

CC:0084.02.0A.11.00B The value 0 MUST indicate that the receiving node MUST supports Wake Up when initiated by an event determined by the application e.g. a pushbutton activation.

CC:0084.02.0A.11.00C Values in the range 1..16777215 MUST indicate Minimum supported Wake Up interval in seconds

Maximum Wake Up Interval Seconds (24 bits)

CC:0084.02.0A.11.003 This field is used to advertise the maximum Wake Up Interval supported by the sending node. The field MUST comply with Table 22

Table 22, Wake Up Interval Capabilities Report::Maximum Wake Up Interval Seconds encoding

Decimal	Description
0	This value is used to indicate that there is no minimum / maximum / default wake up interval. In this case, the sending node is activated by e.g. user interaction in form of a button press. If this field is set to 0, the Minimum and Default Wake Up Interval fields MUST also be 0.
<min interval> ..16777215	Values in this range indicate the maximum wake up interval in seconds supported by the sending node. This field MUST NOT be set to a lower value than the Minimum Wake Up Interval Seconds value. This field MAY be set to the same value as the Minimum Wake Up Interval Seconds value, which means the sending node only supports one value.

Default Wake Up Interval Seconds (24 bits)

This field is used to advertise the default Wake Up Interval value for the sending node.

This field MUST be set to a value included in the range defined by the Minimum Wake Up Interval and the Maximum Wake Up Interval

The Default Wake Up Interval SHOULD be 0 if the Minimum Wake Up Interval is 0.

Wake Up Interval Step Seconds (24 bits)

This field is used to advertise the resolution of valid Wake Up Intervals values for the sending node. The field MUST comply with Table 23

Table 23, Wake Up Interval Capabilities Report::Wake Up Interval Step Seconds encoding

Decimal	Description
0	<p>This value is used to indicate that no interval steps are possible.</p> <p>The battery-operated device only supports the minimum and maximum Wake Up Interval values.</p> <p>This field MUST be set to 0 if the maximum and the minimum interval are equal.</p>
1..16777215	<p>Values in this range indicate the Wake Up Interval step in seconds supported by the sending node.</p> <p>This field's value MUST NOT exceed the difference between the Minimum and Maximum Wake Up Intervals.</p> <p>This field SHOULD have a value so that the difference between the maximum and minimum Wake Up Interval is a multiple of this field's value.</p> <p><i>Examples:</i></p> <p>If a device has minimum wake up interval of 5 minutes (300 seconds) and a maximum wake up interval of 10 minutes (600 seconds), the wake up interval step MUST NOT exceed 5 minutes (300 seconds) as this would be larger than the difference of the minimum and maximum interval. A Wake Up Interval Step of 100 seconds indicates that the nodes supports the following Wake Up Intervals :</p> <p>300 seconds 400 seconds 500 seconds 600 seconds</p>

4.35 Z/IP Naming and Location Command Class, version 1

The Z/IP Naming and Location Command Class is used to manage the Name and a Location text strings of a Z/IP resource.

4.35.1 Compatibility Considerations

A Z/IP resource is identified by an IPv6 address and an End Point ID. Capable Z/IP nodes SHOULD store the naming and location information locally in non-volatile storage. In addition, a Z/IP Resource Directory (RD) [18] MAY store the naming and location of all Z/IP Resources; also sleeping resources. The Z/IP RD manages node properties retrieved from actual nodes as well as extended information which is not always available in classic Z-Wave nodes. Information managed by the Z/IP RD may be announced to Z/IP Clients via service discovery mechanisms such as mDNS [19].

Commands defined in this command class MUST be encapsulated in Z/IP Packets.

A client modifying the name and/or location information MUST ensure that the combined length of the Z/IP Name and Z/IP Location is 62 bytes or less in order to comply with DNS-SD limitations.

4.35.2 Z/IP Name Set Command

This command is used to set the name of a Z/IP Resource.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_NAMING							
Command = ZIP_NAMING_NAME_SET							
Name 1							
...							
Name N							

Name (N bytes)

This field is used to specify the Name portion of the resource name.

The combined Name and Location strings MUST NOT be longer than 62 bytes.

The Name String MUST be UTF-8 encoded. Since a UTF-8 character may require two or more bytes, the maximum length of a name string depends on the composition of the string.

The Name String MUST NOT contain any appended termination characters. The number of bytes MUST be determined from the command length.

The name MUST NOT contain the dot (also known as period) character “.”.
 The escape sequence “\.” (backslash followed by dot) MAY be used for encoding the dot character.
 A user interface SHOULD allow users to enter names that contain dot characters.
 If a dot character is entered, the user interface MUST replace the dot character with the escape sequence “\.” before issuing this command.

The name MUST NOT contain the underscore character “_”.
 The name MUST NOT end with the dash character “-”.

Node names MUST be case insensitive.

4.35.3 Z/IP Name Get Command

This command is used to request the name from a Z/IP Resource

The Z/IP Name Report Command MUST be returned in response to this command.

This command MUST NOT be issued via multicast addressing.
 A receiving node MUST NOT return a response if this command is received via multicast addressing.
 The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_NAMING							
Command = ZIP_NAMING_NAME_GET							

4.35.4 Z/IP Name Report Command

This command is used to advertise the name of a Z/IP Resource.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_NAMING							
Command = ZIP_NAMING_NAME_REPORT							
Name 1							
...							
Name N							

Name (N bytes)

Refer to 4.35.2.

It may be attractive to use dots in names. The escape sequence “\.” (backslash followed by dot) MAY be used for encoding the dot character in names.
 A receiving device SHOULD display the escape sequence “\.” as “.” in user interfaces facing end users.

4.35.5 Z/IP Location Set Command

This command is used to set the location of a Z/IP Resource.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_NAMING							
Command = ZIP_NAMING_LOCATION_SET							
Location 1							
...							
Location N							

Location (N bytes)

This field is used to specify the Location portion of the resource name.

The combined Name and Location strings MUST NOT be longer than 62 bytes.

The Location string MUST be UTF-8 encoded. Since a UTF-8 character may require two or more bytes, the maximum length of a location string depends on the composition of the string.

The Location string MUST NOT contain any appended termination characters. The number of bytes MUST be determined from the message length.

The location string MAY contain the dot (also known as the period) character “.”.

A user interface MUST NOT replace the dot character with the escape sequence “\.” before issuing this Command.

The location string MUST NOT contain the underscore character “_”.

Each location sub-string (separated by the dot character “.”) MUST NOT end with the dash character “-”.

Node locations MUST be case insensitive.

4.35.6 Z/IP Location Get Command

This command is used to request the location from a Z/IP Resource

The Z/IP Location Report Command MUST be returned in response to this command.

This command MUST NOT be issued via multicast addressing.

A receiving node MUST NOT return a response if this command is received via multicast addressing.

The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_NAMING							
Command = ZIP_NAMING_LOCATION_GET							

4.35.7 Z/IP Location Report Command

This command is used to advertise the location of a Z/IP Resource.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_NAMING							
Command = ZIP_NAMING_LOCATION_REPORT							
Location 1							
...							
Location N							

Location (N bytes)

Refer to 4.35.5.

The escape sequence “\.” (backslash followed by dot) MAY be used for encoding the dot character in locations.

A receiving node SHOULD display the escape sequence “\.” as “.” in user interfaces facing end users.

4.36 Z-Wave Plus Info Command Class, version 1 [OBSOLETED]

THIS COMMAND CLASS VERSION HAS BEEN OBSOLETED

New implementations MUST use the Z-Wave Plus Info Command Class Version 2.

CC:005E.01.00.11.001

4.37 Z-Wave Plus Info Command Class, version 2

The Z-Wave Plus Info Command Class is used to differentiate between Z-Wave Plus, Z-Wave for IP and Z-Wave devices. Furthermore this command class provides additional information about the Z-Wave Plus device in question.

The Z-Wave Plus Info Command Class defines two icon types. Icon types allow for a meaningful, homogeneous representation in user and installer Graphical User Interfaces (GUI), respectively. A Z-Wave Plus product **MUST** specify valid icon types. Icon types do not affect the operation of a product or how it is certified. The actual graphical appearance of icons is out of scope of this specification. Any app may define its own icon library mapping to the Icon Type identifiers.

4.37.1 Compatibility Considerations

4.37.1.1 Node Information Frame (NIF)

A supporting node **MUST** always advertise the Z-Wave Plus Info Command Class in its NIF, regardless of the inclusion status and security bootstrapping outcome.

The Z-Wave Plus Info Command Class **MUST** be advertised as the first supported command class in Node Information Frame (NIF).

4.37.2 Multi Channel considerations

A Multi Channel device implements a Root Device and a number of Multi Channel End Points.

The Z-Wave Plus Info Command Class **MUST** be supported for each End Point in order to advertise individual icons for each End Point.

This means that the Z-Wave Plus Version, Role Type and Node Type information is advertised in a redundant fashion. The advertised Z-Wave Plus Version, Role Type and Node Type information values **MUST** be identical for the Root Device and all Multi Channel End Points.

4.37.3 Z-Wave Plus Info Get Command

The Z-Wave Plus Info Get Command is used to get additional information of the Z-Wave Plus device in question.

The Z-Wave Plus Info Report Command **MUST** be returned in response to this command.

This command **MUST NOT** be issued via multicast addressing.

A receiving node **MUST NOT** return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZWAVEPLUS_INFO							
Command = ZWAVEPLUS_INFO_GET							

4.37.4 Z-Wave Plus Info Report Command

The Z-Wave Plus Info Report Command is used to report version of Z-Wave Plus framework used and additional information of the Z-Wave Plus device in question.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZWAVEPLUS_INFO							
Command = ZWAVEPLUS_INFO_REPORT							
Z-Wave Plus Version							
Role Type							
Node Type							
Installer Icon Type MSB							
Installer Icon Type LSB							
User Icon Type MSB							
User Icon Type LSB							

Z-Wave Plus Version (8 bits)

The Z-Wave Plus Version field enables a future revision of the Z-Wave Plus framework where it is necessary to distinguish it from the previous frameworks.

CC:005E.02.02.11.006

This field MUST be set to 1 if the sending node is compliant with Z-Wave Plus [10].
This field MUST be set to 2 if the sending node is compliant with Z-Wave Plus v2 [20].

Role Type (8 bits)

The Role Type field indicates the role the Z-Wave Plus device in question possess in the network and functionalities supported. The full functionality is determined in conjunction with the Device Type.

Node Type (8 bits)

The Node Type field indicates the type of node the Z-Wave Plus device in question possess in the network.

The table below shows the current list of Node Types:

Table 24, Node Type identifiers

Value	Identifier	Node Types
0x00	NODE_TYPE_ZWAVEPLUS_NODE	Z-Wave Plus node
0x02	NODE_TYPE_ZWAVEPLUS_FOR_IP_GATEWAY	Z-Wave Plus for IP gateway

CC:005E.02.02.11.002

All other values are reserved and MUST NOT be used by a sending node. Reserved values MUST be ignored by a receiving node.

Z-Wave Plus for IP gateway

The device provides access to and from IP networks and allows IP hosts to discover resources in the Z-Wave network via Z/IP Discovery.

CC:005E.02.02.11.003

Two IP services **MUST** be available to represent Z-Wave resources in the IP domain: ICMP Echo (Ping) and UDP port 4123 (Z-Wave control protocol).

The device is typically a stand-alone device based on a residential IPv6 router. The Z-Wave interface is presented as an IP network interface to external clients but all IP communication to classic Z-Wave nodes is terminated by the gateway and forwarded in classic Z-Wave frames.

Installer Icon Type (16 bits)

The Installer Icon Type field indicates the icon to use in Graphical User Interfaces for network management, e.g. in a floor plan. Installer Icons provide a finer granularity than user icons, e.g. a light on/off resource may be a built-in wall outlet, a plug-in module or an entire power strip. A sensor may be a single sensor icon or multiple sensors in one casing.

For further details, refer to the User Icon Type section below.

User Icon Type (16 bits)

The User Icon Type field indicates the icon to use in Graphical User Interfaces for end users. User Icons provide a basic granularity, e.g. a light resource is always shown as a light bulb whether the physical device is a built-in wall outlet, a plug-in module or the output of a power strip.

CC:005E.02.02.11.004

In case of multi-endpoint devices, e.g. power strips or multi-sensors, the info command class **MUST** be supported and Icon Types **MUST** be specified for each endpoint.

CC:005E.02.02.13.001

The User Icon Type **MAY** differ for individual endpoints.

CC:005E.02.02.12.001

Icon types are defined by the Z-Wave Alliance [13]. The 16 bit identifier values defined by [13] allows the GUI designer to know how a given product would like to be represented using resources from the GUI designer's own favorite icon library. The icon graphics found in [13] are only suggested graphics. Output device icons **SHOULD** be tailored to the actual geographical region.

CC:005E.02.02.12.002

CC:005E.02.02.11.005

If a GUI does not know the specific icon type, it **SHOULD** use the generic icon type as a fallback alternative. The generic icon type **MUST** be derived by setting the least significant 8 bits of the icon type identifier to zero.

APPENDIX A ASCII CODES

The standard ASCII table defines 128 character codes (from 0 to 127), of which, the first 32 are control codes (non-printable), and the remaining 96 character codes are printable characters. The table below shows the hexadecimal values of the ASCII character codes, e.g. the ASCII code for the capital letter “A” is equal to 0x41:

Table 25, The standard ASCII Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NRK	SYN	ETB	CRN	EM	SUB	ESC	FS	GS	RS	U
2		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	

In addition to the 128 standard ASCII codes (the ones listed above ranging from 0 to 127), most systems have another 128 extra codes which form what is known as extended ASCII (with ranges from 128 to 255). The OEM Extended ASCII character set is included in all PC-compatible computers as the default character set when the system boots before loading any operating system and under MS-DOS. It includes some foreign signs, some marked characters and also pieces to draw simple panels. The table below shows the hexadecimal values of the OEM Extended ASCII character codes, e.g. the ASCII code for the capital letter “Æ” is equal to 0x92:

Table 26, OEM Extended ASCII Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
9	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	ℳ	f
A	á	í	ó	ú	ñ	Ñ	ª	º	¿	¬	½	¼	¡	«	»	
B	▤	▥	▦													
C	L	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
D	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
E	α	β	Γ	Π	Σ	σ	μ	τ	Φ	Θ	Ω	δ	∞	φ	ε	∩
F	≡	±	≥	≤			÷	≈	°	·	·	√	n	²	■	

Below are listed codes for players, radios etc. as an alternative to the OEM Extended ASCII codes. Undefined values MUST be ignored.

Table 27, Players Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	·	□	▶	▯	■	·	▶▶	◀◀	◊							
9																
A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬		®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

APPENDIX B CRC-CCITT SOURCE CODE

The checksum algorithm implements a CRC-CCITT using initialization value equal to 0x1D0F and 0x1021 (normal representation) as the poly.

```

/***** ZW_crc.h *****/
/*****
#ifndef _ZW_CRC_H_
#define _ZW_CRC_H_

#include <ZW_typedefs.h>
/*****
/*      EXPORTED FUNCTIONS      */
/*****

WORD ZW_CreateCrc16( BYTE *pHeadeAddr, BYTE bHeaderLen, BYTE *pPayloadAddr, BYTE bPayloadLen );

/*===== ZW_CheckCrc16 =====
**  CRC-CCITT (0x1D0F) calculation / check
**
**  In: byte string excluding 16bit check field
**  Out: CRC-16 value
**  or
**  In: byte string including 16bit check field
**  Out: zero when OK
**  -----*/

WORD ZW_CheckCrc16( WORD crc, BYTE *pDataAddr, WORD bDataLen );

#endif /* _ZW_CRC_H_ */

```

```

/***** ZW_crc.c *****/
#include <ZW_typedefs.h>
#include <ZW_crc.h>
#include <ZW_uart_api.h>
#define POLY 0x1021 /* crc-ccitt mask */

/* CRC calculation */

/*===== ZW_CheckCrc16 =====
** CRC-CCITT (0x1D0F) calculation / check
**
** In: byte string excluding 16bit check field
** Out: CRC-16 value
** or
** In: byte string including 16bit check field
** Out: zero when OK
**
** and
** In: The crc input should normally be set to the initialization
** value = 0x1D0F.
** It can also be used to carry over crc value between separate
** calculations of multiple parts of a frame, e.g. header and body.
**-----*/
WORD ZW_CheckCrc16( WORD crc, BYTE *pDataAddr, WORD bDataLen ) {
    BYTE WorkData;
    BYTE bitMask;
    BYTE NewBit;

    while(bDataLen--){
        WorkData = *pDataAddr++;
        for (bitMask = 0x80; bitMask != 0; bitMask >>= 1) {
            /* Align test bit with next bit of the message byte, starting with msb. */
            NewBit = ((WorkData & bitMask) != 0) ^ ((crc & 0x8000) != 0);
            crc <<= 1;
            if (NewBit) {
                crc ^= POLY;
            }
        } /* for (bitMask = 0x80; bitMask != 0; bitMask >>= 1) */
    }
    return crc;
}

WORD ZW_CreateCrc16( BYTE *pHeaderAddr, BYTE bHeaderLen, BYTE *pPayloadAddr, BYTE bPayloadLen){
    WORD crc;
    crc = 0x1D0F;
    crc = ZW_CheckCrc16(crc, pHeaderAddr, bHeaderLen);
    crc = ZW_CheckCrc16(crc, pPayloadAddr, bPayloadLen);
    return crc;
}

```

REFERENCES

- [1] Silicon Labs, SDS10242, Software Design Spec., Z-Wave Device Class Specification.
- [2] Silicon Labs, SDS12657, Z-Wave Command Class Specification A-M.
- [3] Silicon Labs, SDS12652, Z-Wave Command Class Specification N-Z.
- [4] IETF RFC 4861, Neighbor Discovery for IP version 6 (IPv6),
<http://tools.ietf.org/pdf/rfc4861.pdf>
- [5] IETF RFC 3122, Extensions to IPv6 Neighbor Discovery for Inverse Discovery Specification,
<http://tools.ietf.org/pdf/rfc3122.pdf>
- [6] IETF RFC 2119, Key words for use in RFCs to Indicate Requirement Levels,
<http://tools.ietf.org/pdf/rfc2119.pdf>
- [7] IETF RFC 2460, Internet Protocol, Version 6 (IPv6) Specification,
<http://tools.ietf.org/pdf/rfc2460.pdf>
- [8] IETF RFC 4291, IP Version6 Addressing Architecture,
<http://tools.ietf.org/pdf/rfc4291.pdf>
- [9] Silicon Labs, SDS11846, Z-Wave Plus Role Types Specification.
- [10] Silicon Labs, SDS11847, Z-Wave Plus Device Types Specification.
- [11] Graphical UI elements,
http://en.wikipedia.org/wiki/Graphical_user_interface_elements
- [12] Silicon Labs, SDS13425, Z-Wave Plus Assigned Manufacturer IDs.
- [13] Silicon Labs, SDS13738, Z-Wave Plus Assigned Icon Types.
- [14] Silicon Labs, SDS13548, List of defined Z-Wave Command Classes
- [15] Silicon Labs, SDS13783, Z-Wave Transport-Encapsulation Command Class Specification
- [16] Silicon Labs, SDS13781, Z-Wave Application Command Class Specification
- [17] Silicon Labs, SDS13784, Z-Wave Network-Protocol Command Class Specification
- [18] Silicon Labs, SDS11633, Software Design Spec., Z/IP Resource Directory.
- [19] Silicon Labs, SDS11756, Software Design Spec., Z/IP DNS Discovery Support.
- [20] Silicon Labs, SDS14224, Z-Wave Plus v2 Device Type Specification
- [21] Silicon Labs, SDS14220, Indicator Command Class, list of assigned Indicator and Property IDs
- [22]