Software Design Specification

Z-Wave Command Class Specification

Document No.:	SDS11060
Version:	7
Description:	This document describes the Command Classes and associated Commands used by Z Wave enabled products ensuring that compliant products will be interoperable.
Written By:	JFR;JRM
Date:	2009-05-05
Reviewed By:	CHL;JRM;SGR;ABR
Restrictions:	Partners Only

Approved by:

Date	CET	Initials	Name	Justification
2009-05-05	16:20:12	NTJ	Niels Thybo Johansen	

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20070920	JFR	ALL	All Command Classes from SDS10242-12 added.
1	20070929	JFR	Section 3.23	Latitude and longitude only 16 bits in Geographic Command Class.
1	20071213	JFR	Appendix A	Manufacturer ID table updated.
1	20071227	JFR	Section 3.7	Association Command Configuration Command Class added.
1	20071227	JFR	Section 3.2 Section 3.3	Alarm Sensor Command Class added. Alarm Silence Command Class added.
1	20071227	JFR	Section 3.64	Sensor Configuration Command Class added.
1	20071227	JFR	Section 3.72	Thermostat Setback Command Class added.
2	20080121	JFR		Composite Command Class discontinued Association Command Class version 2 discontinued
3	20080315	JFR	Section 3.18	Configuration Command Class version 2 added
3	20080812	JFR	Section 3.64	Security Command Class added.
3	20080812	JFR	Section 3.1 Section 3.2 Section 3.3	Advanced Z/IP Client Command Class added Advanced Z/IP Server Command Class added Advanced Z/IP Services Command Class added
3	20080812	JFR	Section 3.81 Section 3.82 Section 3.83	Z/IP Client Command Class added Z/IP Server Command Class added Z/IP Services Command Class added
3	20080812	JFR	Section 3.42 Section 3.40 Section 3.7	Multi Channel Command Class, version 2 added as an extension to Multi Instance Command Class, version 1. Multi Channel Association Command Class, version 2 added as an extension to Multi Instance Association Command Class, version 1. Association Command Class, version 2 added.
3	20080814	JFR	Section 3.43	Multilevel Sensor Command Class extended to version 3.
3	20080814	JFR	Section 3.67	Basic Tariff Information Command Class added.
4	20080818	JFR	Section 3.7	Added clarification A) in Association Command Configuration Command Class.
4	20080826	JRM	Section 3.62	Secure Network Inclusion timeouts clarified.
4	20080827	JFR	Section 3.23	Firmware Update Meta Data Command Class, version 2 added.
4	20081118	JFR	Section 3.48	Clarified announcement of No Operation Command Class in Node Information Frame (NIF)
5	20081202	JFR	Section 3.74 & 3.75	Time Command Class is split into 2 versions; a simple one for transferring time and date info between Z-Wave devices (version 1) and a more complex one including time zones and daylight saving (version 2). Notice discontinuation of the old Time Command Class version 1.
5	20081202	JFR	Section 3.70 Section 3.73	Thermostat Mode Command Class version 2 added Thermostat Setpoint Command Class version 2 added
5	20090129	JFR	Section 3.14 & 3.46	Warning against using Toggle Switch Command Classes in new devices
5	20090212	JFR	Section 3.79.4 Appendix A	Clarification when using Wake Up Notification Command broadcast Manufacturer ID's updated
6	20090218	JFR	Section 3.36 Section 3.44.1	Meter Command Class version 2 added Text "100…254 (0x63…0xFE)" changed to "100…254 (0x64…0xFE)"
6	20090320	JRM	Section 3.64.1.3 Section 3.64.1 Section 3.64.2 Section 3.64.2.2 Section 3.64.2.1 Section 3.64.1 Section 3.64.2.1 Section 3.64.1 Section 3.64.2.1 Section 3.64.2.1.1	Added security frame flow diagram to illustrate sequencing and updated description. Sequence counter text updated. Network streaming made mandatory Updated to only support security 0 using low power to improve usability, but security scheme is extendable to offer stronger key exchange at a later stage. Notice: New security 0 value! Updated to only support security 0 using normal power to improve usability, but security scheme is extendable to offer stronger key exchange at a later stage. Nonce Timers described in more detail, and changed from 3 seconds to a defined variable Added / described inclusion timer in more detail. Timers must be started as soon as message is sent, not after Acknowledge is received. State diagram for inclusion timers added

7	20090404	ABR JFR	Section 3.81 Section 3.82 Section 3.83 Section 3.83.1	Z/IP Client Cmd Class renamed to Z/IP Tunneling Client Cmd Class Z/IP Server Cmd Class renamed to Z/IP Tunneling Server Cmd Class Z/IP Services Cmd Class renamed to Z/IP Tunneling Services Cmd Class Sequence number removed from Z/IP Packet Cmd.
7	20090404	JFR	Section 3.27 Section 3.28	HRV Status Command Class added HRV Control Command Class added
7	20090429	JFR	Section 3.46	Multilevel Switch Command Class version 3 added
7	20090430	JFR	Section 3.41	Clarify that Multi Channel Command Class version 2 must be used instead of Multi Instance Command Class version 1.

Table of Contents

Table of Figures

Table of Tables

1 ABBREVIATIONS

Abbreviation	Explanation
AMR	Automatic Meter Reading
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange. An ASCII code is the numerical representation of a character.
AV	Audio/Video
DHCP	Dynamic Host Configuration Protocol.
DNS	Dynamic Host Service
DST	Daylight Savings Time
HRV	Heat Recovery Ventilation
ID	Identifier
IP	Internet Protocol
IPV4	Internet Protocol version 4
IPV6	Internet Protocol version 6
LF	Linefeed character.
LSB	Less significant byte
MSB	Most significant byte
NIF	Node Information Frame
PIR	Pyroelectric Infrared Motion Sensor
SUC	Static Update Controller
TZO	Time Zone Offset
Unicode	Unicode is a standard for encoding of characters. For more information please visit http://www.unicode.org/
UTC	Universal Time (sometimes also called “Zulu Time”) was called Greenwich Mean Time (GMT) before 1972
WMC	Windows Vista Media Center and Media Center 2005 remote controls

2 INTRODUCTION

This document describes the command classes and associated commands that must be used when designing and implementing Z-Wave™ products. A subset of command classes is typically mandatory for a given type of device. All commands are handled by the application layer of the Z-Wave protocol. This document should be read in conjunction the Z-Wave Device Class Specification [1].

2.1 Purpose

The purpose of this document is to describe the command classes used by the application layer of the Z-Wave protocol.

2.2 Audience and prerequisites

The audience of this document is Z-Wave partners and Zensys.

2.3 Precedence of definitions

In terms of reviewing products for Z-Wave Compliance, definitions in this document have precedence over the header file ZW_classcmd.h distributed as part of the Z-Wave Developer's Kit. However, the assignment of all device and command class hex identifiers can only be found in the header file ZW_classcmd.h.

Device and Command Class Specifications approved as final version (ver. 1.00) during the device/command class development process have precedence over this document temporarily until integrated into this document.

2.4 Terms used in this document

This document describes mandatory and optional aspects of the required compliance of a Z-Wave product to the Z-Wave standard.

The words “shall” and “must” specify aspects that are mandatory for compliance. Equally, “must not” has to be adhered to for compliance. Products that are in violation any such statement are considered to be ***not*** Z-Wave compliant.

The words “may”, “could”, and “may not” leave the choice to the implementer. “Recommended” also leaves the choice formally to the manufacturer, but provides additional guidance.

3 COMMAND CLASSES

Interoperability between devices is based on Command Classes. If a controlling device and a slave device understand the same Command Class then these devices are able to communicate.

The Command Class range is shown below:

Command Class	Description
0x00	No Operation. Used by Z-Wave Protocol and optional by the application.
0x01 – 0x1F	Reserved for the Z-Wave protocol
0x20 – 0xEE	Application Command Classes
0xEF	Support/Control Mark
0xF0	Non interoperable
0xF1 – 0xFF	Extended Application Command Classes

Table 1, Command Class identifier range

A Command Class can contain up to 255 different Commands. If the Command Class field is set to 0xF1 through 0xFF then there is another Command Class byte added. This allows for future extensions of the Command Classes. The strategy of having an Extended Command Class followed by the actual command identifier provides the possibility of having more than 4000 Command Classes.

All commands have a common header consisting of a Command Class identifier and a Command identifier. Further the command can have from zero to n bytes of command data. The figures below show the generic command frame for the two possible formats:

7	6	5	4	3	2	1	0
Command Class (0x20 – 0xEE)							
Command							
Command Data 1							
..							
Command Data n							

Figure 1, Generic command format

7	6	5	4	3	2	1	0
Command Class MSB (0xF1 – 0xFF)							
Command Class LSB (0x00 – 0xFF)							
Command							
Command Data 1							
..							
Command Data n							

Figure 2, Generic extended command format

All command classes have a version number. The following rules apply to avoid interoperability issues when introducing the same command class with different versions:

1. A node must not discard a frame based on the length field. All frames must be interpreted by the command class identifier and the command identifier. Thereby can a version 1 command class implementation in a device interpret the version 1 part of the received version 2 command.
2. All implementations of a command class version higher then 1 must initialize all parameters associated with the version higher then 1. Thereby can a version 2 command class implementation in a device interpret a received version 1.
3. A device supporting a Command Class having a version higher than 1 must support the Version Command Class to be able to identify the supported version. In case the device doesn’t support the Version Command Class then it can be assumed that all command classes are equal to version 1.
4. It is allowed to make devices only supporting an older version of the command class despite a newer version exists as long as the generic/specific device specification does not require a specific version implemented.

The number of data fields transmitted can be determined from the length field returned by the ApplicationCommandHandler. The length field is used in cases where the same command has a variable number of command data fields.

Reserved values and reserved bits

Values of fields in commands that are marked as “reserved” must not be used by devices sending commands and shall be ignored by devices receiving commands.

Bits in commands that are marked as “reserved” shall be set to 0 by devices sending commands and shall be ignored by devices receiving commands.

Command Class

The Command Class field indicates what group of commands the command is part of. The currently defined Command Classes are split in two tables depending on the purpose. The first table shows the current list of general purpose Command Classes applicable for many different device types:

General Command Class	ZW_classcmd.h
Advanced Z/IP Client	COMMAND_CLASS_ZIP_ADV_CLIENT
Advanced Z/IP Server	COMMAND_CLASS_ZIP_ADV_SERVER
Advanced Z/IP Services	COMMAND_CLASS_ZIP_ADV_SERVICES
Alarm	COMMAND_CLASS_ALARM
Application Status	COMMAND_CLASS_APPLICATION_STATUS
Association	COMMAND_CLASS_ASSOCIATION
Association Command Configuration	COMMAND_CLASS_ASSOCIATION_COMMAND_CONFIGURATION
Battery	COMMAND_CLASS_BATTERY
Clock	COMMAND_CLASS_CLOCK
Configuration	COMMAND_CLASS_CONFIGURATION
Controller Replication	COMMAND_CLASS_CONTROLLER_REPLICATION
Firmware Update Meta Data	COMMAND_CLASS_FIRMWARE_UPDATE_MD
Geographic Location	COMMAND_CLASS_GEOGRAPHIC_LOCATION
Grouping Name	COMMAND_CLASS_GROUPING_NAME
Hail	COMMAND_CLASS_HAIL
Indicator	COMMAND_CLASS_INDICATOR
IP Configuration	COMMAND_CLASS_IP_CONFIGURATION
Language	COMMAND_CLASS_LANGUAGE
Manufacturer Proprietary	COMMAND_CLASS_MANUFACTURER_PROPRIETARY
Manufacturer Specific	COMMAND_CLASS_MANUFACTURER_SPECIFIC
Mark (Support/control mark)	COMMAND_CLASS_MARK
Multi Channel	COMMAND_CLASS_MULTI_CHANNEL
Multi Channel Association	COMMAND_CLASS_MULTI_CHANNEL_ASSOCIATION
Multi Command	COMMAND_CLASS_MULTI_COMMAND
Multi Instance	COMMAND_CLASS_MULTI_INSTANCE
Multi Instance Association	COMMAND_CLASS_MULTI_INSTANCE_ASSOCIATION
No Operation	COMMAND_CLASS_NO_OPERATION
Node Naming and Location	COMMAND_CLASS_NODE_NAMING
Non interoperable	COMMAND_CLASS_NON_INTEROPERABLE
Proprietary	COMMAND_CLASS_PROPRIETARY
Remote Association Activate	COMMAND_CLASS_REMOTE_ASSOCIATION_ACTIVATE
Remote Association Configuration	COMMAND_CLASS_REMOTE_ASSOCIATION
Screen Attributes	COMMAND_CLASS_SCREEN_ATTRIBUTES
Screen Meta Data	COMMAND_CLASS_SCREEN_MD
Security	COMMAND_CLASS_SECURITY
Time	COMMAND_CLASS_TIME
Time Parameters	COMMAND_CLASS_TIME_PARAMETERS
User Code	COMMAND_CLASS_USER_CODE
Version	COMMAND_CLASS_VERSION
Wake Up	COMMAND_CLASS_WAKE_UP
Z/IP Tunneling Client	COMMAND_CLASS_ZIP_TUN_CLIENT
Z/IP Tunneling Server	COMMAND_CLASS_ZIP_TUN_SERVER
Z/IP Tunneling Services	COMMAND_CLASS_ZIP_TUN_SERVICES

Table 2, General purpose Command Class identifiers

The second table shows the current list of Command Classes targeted for certain types of devices:

Device Related Command Class	ZW_classcmd.h
Alarm Sensor	COMMAND_CLASS_SENSOR_ALARM
Alarm Silence	COMMAND_CLASS_SILENCE_ALARM
All Switch	COMMAND_CLASS_SWITCH_ALL
Basic	COMMAND_CLASS_BASIC
Basic Tariff Information	COMMAND_CLASS_BASIC_TARIFF_INFO
Basic Window Covering	COMMAND_CLASS_BASIC_WINDOW_COVERING
Binary Sensor	COMMAND_CLASS_SENSOR_BINARY
Binary Switch	COMMAND_CLASS_SWITCH_BINARY
Binary Toggle Switch	COMMAND_CLASS_SWITCH_TOGGLE_BINARY
Climate Control Schedule	COMMAND_CLASS_CLIMATE_CONTROL_SCHEDULE
Door Lock	COMMAND_CLASS_DOOR_LOCK
Energy Production	COMMAND_CLASS_ENERGY_PRODUCTION
HRV Status Command Class	COMMAND_CLASS_HRV_STATUS
HRV Control Command Class	COMMAND_CLASS_HRV_CONTROL
Lock	COMMAND_CLASS_LOCK
Meter	COMMAND_CLASS_METER
Move To Position Window Covering	COMMAND_CLASS_MTP_WINDOW_COVERING
Multilevel Sensor	COMMAND_CLASS_SENSOR_MULTILEVEL
Multilevel Switch	COMMAND_CLASS_SWITCH_MULTILEVEL
Multilevel Toggle Switch	COMMAND_CLASS_SWITCH_TOGGLE_MULTILEVEL
Powerlevel	COMMAND_CLASS_POWERLEVEL
Protection	COMMAND_CLASS_PROTECTION
Pulse Meter	COMMAND_CLASS_METER_PULSE
Scene Activation	COMMAND_CLASS_SCENE_ACTIVATION
Scene Actuator Configuration	COMMAND_CLASS_SCENE_ACTUATOR_CONF
Scene Controller Configuration	COMMAND_CLASS_SCENE_CONTROLLER_CONF
Sensor Configuration	COMMAND_CLASS_SENSOR_CONFIGURATION
Simple AV Control	COMMAND_CLASS_SIMPLE_AV_CONTROL
Thermostat Fan	COMMAND_CLASS_THERMOSTAT_FAN_MODE
Thermostat Fan State	COMMAND_CLASS_THERMOSTAT_FAN_STATE
Thermostat Mode	COMMAND_CLASS_THERMOSTAT_MODE
Thermostat Operating State	COMMAND_CLASS_THERMOSTAT_OPERATING_STATE
Thermostat Setback	COMMAND_CLASS_THERMOSTAT_SETBACK
Thermostat Setpoint	COMMAND_CLASS_THERMOSTAT_SETPOINT

Table 3, Device related Command Class identifiers

Refer to ZW_classcmd.h source code file for the assigned Command Class identifiers. In ZW_classcmd.h additional constants and variables are defined to support the implementation.

Command (8 bit)

The command field contains the specific command that should be executed.

Command Data (0 – n*8 bit)

The command data field contains data related to the command. Simple commands, such as get commands, contain no command data. Other commands, such as set or report commands can contain several bytes of command data.

The following subchapters contain a description of the Command Classes listed in the tables below.

3.1 Advanced Z/IP Client Command Class, version 1

The Advanced Z/IP Client Command Class is the client part when carrying IP packets between Z/IP nodes.

3.1.1 Z/IP Subnet Request Report Command

Z/IP Gateway → Z/IP Node

The Z/IP Subnet Request Report Command is sent as a result of receiving a Z/IP Subnet Request Get command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_ADV_CLIENT							
Command = COMMAND_ZIP_SUBNET_REQUEST_REPORT							
IP Version							
IP Subnet address 1							
...							
IP Subnet address 15							

IP Version (8 bits)

Specify the IP version

IP Version	Value
IPv4	4
IPv6	6

All IP versions not mentioned in the table should be considered invalid and the frame should be ignored.

IP Subnet address (120 bits)

Specify the IP subnet address of the Z-Wave network.

IP Version	IP Subnet location
IPv4	bytes 1..3 (24 bits)
IPv6	bytes 1..15 (120 bits)

Depending on the IP version, the signaled subnet is either 24 bits (IPv4) or 120 bits (IPv6).
The number of host addresses in a Z/IP network is always 255 (232), thus mapping into an 8 bit host address.

3.2 Advanced Z/IP Server Command Class, version 1

The Advanced Z/IP Server Command Class is the server part when carrying IP packets between Z/IP nodes.

3.2.1 Z/IP Subnet Request Get Command

Z/IP Gateway ← Z/IP Node

The Z/IP Subnet Request Get Command can be used by a Z/IP node to query the IP subnet address of the Z/IP network.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_ADV_SERVER							
Command = COMMAND_ZIP_SUBNET_REQUEST_GET							

(no payload is carried in this command)

After locating a Z/IP Gateway, a Z/IP node may query the IP subnet of the local Z/IP network from the Z/IP Gateway. The Z/IP gateway must return a `Z/IP Subnet Request Report` command in response to the `Z/IP Subnet Request Get` command.

A Z/IP node must send a `Z/IP Subnet Request Get` command to the Z/IP gateway on receipt of a `Z/IP Gateway Set` command.

3.3 Advanced Z/IP Services Command Class, version 1

The Advanced Z/IP Services Command Class can be used to initialize Z/IP nodes and to carry IP packets between Z/IP nodes; including the Z/IP gateway.

3.3.1 Z/IP IP Datagram Segment Command

The Z/IP IP Datagram Segment Command can be used to carry a segment of an IP datagram.

First segment:

7	6	5	4	3	2	1	0
Command Class= COMMAND_CLASS_ZIP_ADV_SERVICES							
Command = COMMAND_ZIP_IP_DATAGRAM_SEGMENT							
First segment == '1'	Last segment	Last datagram	*reserved*			Sequence count	
IP Header compression control							
Payload 1							
...							
Payload N							

Up to 46 payload bytes may be carried in a 'first segment' frame. The number of payload bytes transmitted can be determined from the length field in the frame.

Following segment(s):

7	6	5	4	3	2	1	0
Command Class= COMMAND_CLASS_ZIP_ADV_SERVICES							
Command = COMMAND_ZIP_IP_DATAGRAM_SEGMENT							
First segment == '0'	Last segment	Last datagram	*reserved*			Sequence count	
Payload 1							
...							
Payload N							

Up to 47 payload bytes may be carried in a non-'first segment' frame. The number of payload bytes transmitted can be determined from the length field in the frame.

First segment (1 bit)

The `First segment` flag signals that this is the first segment of a datagram.
It may be the last segment at the same time if the datagram is short enough to fit into one segment.

First segment	Value
First segment	'1'
Not first segment	'0'

Last segment (1 bit)

The `Last segment` flag signals that the actual segment terminates the datagram.

First segment	Value
Last segment	'1'
Not last segment	'0'

Last datagram (1 bit)

The `Last datagram` flag signals that the actual datagram was the last datagram in the senders TX queue.

First segment	Value
Last datagram	'1'
Not last datagram	'0'

Sequence Count (2 bits)

The receiving end may receive multiple copies of a frame due to retransmissions caused by missing Acks. The `Sequence Count` field is a modulo 4 counter controlling the reception of datagram segments. The first segment of a segmented datagram must always hold the value '00'.

First segment	Value
First segment	'00'
Other segments	previousFrame.SeqCount + 1

IP Header Compression control

If `First segment = '1'`, an 8-bit field holds the IP Header Compression key (identifiers may be OR'ed).

Identifier	Encoding	Comment
0x00	No header compression	
0x01 (bit 0)	SIP is Z/IP local	Source IP address is omitted. Reconstruct SIP from Z/IP PAN subnet and Z-Wave node ID.
0x02 (bit 1)	DIP is Z/IP local	Destination IP address is omitted. Reconstruct DIP from Z/IP PAN subnet and Z-Wave node ID.
0x04 (bit 2)	DPORT is 4123	Destination port is omitted. Re-insert "4123" into DPORT field in receiving node.
0x08 .. 0xFF	*Reserved*	must be zero.

A receiving node should allocate enough memory for reconstruction of the header in order to avoid re-arranging all other received bytes in the datagram. The amount of bytes to allocate depends on the IP version (IPv4 or IPv6).

Reserved

Reserved bits must be set to zero.

3.1 Alarm Command Class, version 1

The Alarm Command Classallows applications to report alarm or service conditions. Since these parameters are not standardized across devices it is recommended that the alarms/service parameters be described in the user manual (or an installer manual).

3.1.1 Alarm Get Command

The Alarm Get Command is used to get the value of an alarm.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ALARM							
Command = ALARM_GET							
Alarm Type							

Alarm Type (8 bit)

The Alarm Type field specifies which alarm is being requested. The alarm types are specific for each application.

3.1.2 Alarm Report Command

The Alarm Report Command is used to report the type and level of an alarm. The Alarm Report Command can be sent unsolicited or requested by the Alarm Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ALARM							
Command = ALARM_REPORT							
Alarm Type							
Alarm Level							

Alarm Type (8 bit)

Refer to explanation under the Alarm Get Command.

Alarm Level (8 bit)

The alarm level is application specific.

3.2 Alarm Sensor Command Class, version 1

The Alarm Sensor Command Class can be used to realize Sensor Alarms.

3.2.1 Alarm Sensor Get Command

The Alarm Sensor Get Command can be used to request the status of a sensor.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SENSOR_ALARM							
Command = SENSOR_ALARM_GET							
Sensor Type							

Sensor Type (8 bit)

Sensor type specifies what type of sensor this command originates from. Refer to the table below with respect to defined sensors. The sensor type value 0xFF returns the first found supported sensor type in the bit mask (starting from bit 0 in Bit Mask 1) by the Alarm Sensor Supported Report. New sensor types/values can be requested from Zensys.

Sensor Type	Value
General Purpose Alarm	0x00
Smoke Alarm	0x01
CO Alarm	0x02
CO_2 Alarm	0x03
Heat Alarm	0x04
Water Leak Alarm	0x05
Return first Alarm on supported list	0xFF

3.2.2 Alarm Sensor Report Command

The Alarm Sensor Report Command can be sent unsolicited when the alarm state changes or as a result of receiving a Alarm Sensor Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SENSOR_ALARM							
Command = SENSOR_ALARM_REPORT							
Source Node ID							
Sensor Type							
Sensor State							
Seconds 1 (MSB)							
Seconds 2 (LSB)							

Source Node ID (8 bit)

Specify the source node ID, which detected the alarm condition. In a Zensor Net is it not possible to determine the source node ID because the frame is broadcast forwarded without this information on protocol level.

Sensor Type (8 bit)

See description under Alarm Sensor Get. The Sensor Type equal to 0xFF cannot be return by the report.

Sensor State (8 bit)

The Sensor State parameter returns the current alarm state. The value 0x00 indicates no alarm and 0xFF indicates alarm. Furthermore it can return values from 0x01 to 0x64 to indicate severity of the alarm in percentage.

The values 101…254 (0x65…0xFE) are reserved and shall be ignored by receiving devices.

Seconds 1..2 (16 bit)

The field Seconds indicates time the remote alarm must be active since last received report. The value 0x0000 indicates that the time field must be ignored.

3.2.3 Alarm Sensor Supported Get Command

The Alarm Sensor Supported Get Command is used to request the supported sensor types from the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SENSOR_ALARM							
Command = SENSOR_ALARM_SUPPORTED_GET							

3.2.4 Alarm Sensor Supported Report Command

The Alarm Sensor Supported Report is used to report the supported sensor types from the device. It can be sent as a result of receiving a Alarm Sensor Supported Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SENSOR_ALARM							
Command = SENSOR_ALARM_SUPPORTED_REPORT							
Number of Bit Masks							
Bit Mask 1							
…							
Bit Mask N							

Number of Bit Masks (8 bit)

Indicates the Number of Bit Masks fields used in bytes.

Bit Mask 1 .. Bit Mask N (N * Bytes)

The Bit Mask fields describe the supported sensor types by the device. The bit 0 in Bit Mask 1 field is used to indicate whether Sensor Type = 0 (General Alarm) is supported or not. The sensor type is supported if the bit is 1 and the opposite if 0. The bit 1 in Bit Mask 1 field is used by Sensor Type = 1 (Smoke Alarm) and so forth. It is only necessary to send the Bit Mask fields from 1 and up to the one indicating the last supported sensor type. The number of Bit Mask fields transmitted can be determined from the length field in the frame.

3.3 Alarm Silence Command Class, version 1

The Alarm Silence Command Class can be used to nuisance silence to temporarily disable the sounding of the alarm but still keep the alarm operating.

3.3.1 Alarm Silence Set Command

The Alarm Silence Set Command can be used to remotely silence the sensor alarm.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SILENCE_ALARM							
Command = SENSOR_ALARM_SET							
Mode							
Seconds 1 (MSB)							
Seconds 2 (LSB)							
Number of Bit Masks							
Bit Mask 1							
…							
Bit Mask N							

Mode (8 bit)

Sensor type specifies what type of sensor this command originates from. Refer to the table below with respect to defined sensors. New sensor types/values can be requested from Zensys.

Mode	Value
Disable sounding of all sensor alarms independent of bit mask	0x00
Disable sounding of all sensor alarms independent of bit mask which have received the alarm via the Sensor Alarm Report command	0x01
Disable sounding of all sensor alarms according to bit mask	0x02
Disable sounding of all sensor alarms according to bit mask which have received the alarm via the Alarm Sensor Report Command	0x03

Seconds 1..2 (16 bit)

The field Seconds indicates the duration sounding of the alarm must be disable but still keep the alarm operating. If silence is engaged, the alarm will come back on when the duration expires unless the originating sensor clears the alarm. The value 0x0000 indicates that the time field must be ignored.

Number of Bit Masks (8 bit)

Indicates the Number of Bit Masks fields used in bytes.

Bit Mask 1 .. Bit Mask N (N * Bytes)

The Bit Mask fields describe the sensor types to disable sounding from. The bit 0 in Bit Mask 1 field is used to indicate whether Sensor Type = 0 (General Alarm) is to be disabled or not. The sensor type must be disabled if the bit is 1 and the opposite if 0. The bit 1 in Bit Mask 1 field is used by Sensor Type = 1 (Smoke Alarm) and so forth. It is only necessary to send the Bit Mask fields from 1 and up to the one indicating the last sensor type to be disabled.

3.4 All Switch Command Class, version 1

The All Switch Command Class is used to switch all devices on or off. Devices can be excluded/included from the all on/all off functionality. The application determines which devices there are included in the all on/all off functionality as default.

3.4.1 All Switch Set Command

The All Switch Set Command is used to tell a device if it should be included or excluded from the all on/all off functionality.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_ALL							
Command = SWITCH_ALL_SET							
Mode							

Mode (8 bit)

The mode field is used to set the all on/all off functionality of the device.

Mode	Description
0x00	Indicate that the switch is excluded from the all on/all off functionality.
0x01	Indicate that the switch is excluded from the all on functionality but not all off.
0x02	Indicate that the switch is excluded from the all off functionality but not all on.
0xFF	Indicates that the switch is included in the all on/all off functionality.

3.4.2 All Switch Get Command

The All Switch Get Command can be used to ask a device if it is included or excluded from the all on/all off functionality.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_ALL							
Command = SWITCH_ALL_GET							

3.4.3 All Switch Report Command

The All Switch Report Command is used to report if the device is included or excluded from the all on/all off functionality. The All Switch Report Command can be sent unsolicited or as a result of receiving an All Switch Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_ALL							
Command = SWITCH_ALL_REPORT							
Mode							

Mode (8 bit)

Refer to explanation under the All Switch Command.

3.4.4 All Switch On Command

The All Switch On Command can be used to inform a switch that it should be turned on.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_ALL							
Command = SWITCH_ALL_ON							

3.4.5 All Switch Off Command

The All Switch Off Command can be used to inform a switch that it should be turned off.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_ALL							
Command = SWITCH_ALL_OFF							

3.5 Application Status Command Class, version 1

All devices should as far as possible support the Application Status Command Class. This 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.

3.5.1 Application Busy Command

The Application Busy Command is 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 bit)

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 bit)

The time in seconds a node should wait before retrying the request.

3.5.2 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 is 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 bit)

The status field can have the following values

Status	Description
0	Supported command rejected by the application in the receiving node

3.6 Association Command Class, version 1

The Association Command Class defines the commands necessary to create and maintain associations in a device. The associations are a list of devices the device wants to control on application level. Some devices may wish to have several groupings of associated nodes so each can be controlled by different events. The groupings are addressed by a Grouping Identifier with up to 255 different groupings of nodes associated to different events.

To configure a controller from another node requires support of the Association Command Class because it needs information about which nodes to be used in the routing table.

A routing slave has not a routing table as found in a controller. This requires that the routing slave in addition is configured with return routes for the wanted associations to obtain a reliable and robust network.

3.6.1 Association Set Command

The Association Set Command is used to add nodes to a given grouping identifier. The node receiving the set command should add the nodes received to the nodes already associated by this grouping until the grouping is full. Remember that routing slaves also must have assigned return routes by a controller using the API call ZW_AssignReturnRoute [2] to all the associated nodes. Delete all return routes by the API call ZW_DeleteReturnRoute before assignment. This is of course necessary for all the associated nodes out of direct range but should always be done default to all the associated nodes to create a reliable and robust network. It's optional whether the return routes are assigned before or after the Association Set command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION							
Command = ASSOCIATION_SET							
Grouping identifier							
Node ID1							
..							
NodeIDn							

Grouping identifier (8 bit)

This grouping identifier is used to instruct how nodes are grouped together. The grouping identifier values must be a sequence starting from 1. This field can be ignored in case the node only supports one grouping.

NodeID1 .. NodeIDn

These fields contain a list of node IDs that should be associated with the grouping.

3.6.2 Association Get Command

The Association Get Command is used to request the current association status of the node. The node receiving this command should answer with Association Report Command.

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

Grouping identifier (8 bit)

This grouping identifier is used to identify how nodes are grouped together. The grouping identifier values must be a sequence starting from 1. This field can be ignored in case the node only supports one grouping.

3.6.3 Association Report Command

The Association Report Command should be used to report all nodes associated with the matching grouping identifier. Be aware that it's only possible to get information about how many groupings a given node are associated to, but not the total number of groupings. Therefore the Grouping Identifiers should be allocated with care starting from 0x01 to avoid unnecessary overhead in finding the groupings with associations. A remote with up to 6 different groupings there are controlled by 6 buttons numbered 1...6 could use the same numbers as grouping identifiers. The Association Report Command can be sent unsolicited or as a result of receiving an Association Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION							
Command = ASSOCIATION_REPORT							
Grouping Identifier							
Max Nodes Supported							
Reports to Follow							
NodeID1							
…							
NodeIDn							

Grouping identifier (8 bit)

This grouping identifier is used to indicate which of the groupings the node list belongs to.

Max Nodes Supported (8 bit)

Maximum number of nodes grouping identifier above supports.

Reports to Follow (8 bit)

This value indicates how many report frames there is left before the entire node IDs associated with the given grouping identifier is transferred.

NodeID1 .. NodeIDn

These fields contain a list of node IDs that are associated with the node sending the report.

3.6.4 Association Remove Command

The Association Remove Command is used to remove nodes from a given grouping at the node receiving the command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION							
Command = ASSOCIATION_REMOVE							
Grouping identifier							
Node ID1							
..							
Node IDn							

Grouping identifier

This grouping identifier is used to determine from which grouping the supplied node ID’s should be removed.

NodeID1 .. NodeIDn

These fields contain a list of node ID’s that should be removed from the specified grouping identifier. In case no node ID’s are supplied the whole grouping should be cleared.

3.6.5 Association Supported Groupings Get Command

The Association Supported Groupings Get Command is used request the number of groupings that this node supports.

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

3.6.6 Association Supported Groupings Report Command

The Association Supported Groupings Report Command is used to report the maximum number of groupings the given node supports. The Association Supported Groupings Report Command can be sent unsolicited or as a result of receiving an Association Supported Groupings Get Command.

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

Supported Groupings (8 bit)

The number of groupings this node supports.

3.7 Association Command Class, version 2

The Association Command Class defines the commands necessary to create and maintain associations in a device. The following is a list of the command that has been changed or added in version 2. The commands not mentioned here will remain the same.

3.7.1 Association Remove Command

The Association Remove Command is used to remove nodes from a given grouping at the node receiving the command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ASSOCIATION							
Command = ASSOCIATION_REMOVE							
Grouping identifier							
Node ID1							
..							
Node IDn							

Grouping identifier

This grouping identifier is used to determine from which grouping the supplied node ID's should be removed.

NodeID1 .. NodeIDn

These fields contain a list of node ID's that should be removed from the specified grouping identifier. In case no node ID's are supplied the whole grouping should be cleared.

In version 2 grouping identifier in conjunction with sequence of NodeID's are interpreted as follows:

	Grouping identifier	Number of node ID's in list
Clear all node ID's in grouping X	$1 \leq X \leq N$	0
Clear specified node ID's in all groupings	0	>0
Clear all node ID's in all groupings	0	0

3.7.2 Association Specific Group Get Command

The Association Specific Group Get Command is used request the current active group from a node. This can be used to set up an association to a specific group using a remote.

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

3.7.3 Association Specific Group Report Command

The Association Specific Group Report Command is used to report the current active group. The Association Specific Group Report command can be sent unsolicited or as a result of receiving an Association Specific Group Get command.

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

Group (8 bit)

The current active group number (1-255). Set to 0 if undefined (the RS might not support the feature).

3.8 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 Node IDs in a group as defined in the Association Command Class in a 1st node.

Mandatory requirement: The device must implement the Association Command Class as ‘supported’

Mandatory requirement: The Association Command Class and the Command Configuration Command Class must be linked through the following dependencies

A) 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.

B) Nodes added to an association through a Command Configuration Set must also be reported in an Association Report and Association Composite Report

C) All commands associated to a grouping identifier//Node ID pair will be removed as a result of an Association Remove. The related command records will be released

D) Command(s) associated to a grouping identifier/Node ID/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 can 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), no new commands can 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 can not 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.

3.8.1 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 [1].

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

3.8.2 Command Records Supported Report Command

The Command Records Supported Report Command is used to report information regarding the Command records. The Command Records Supported Report Command can be sent unsolicited or as a result of receiving a Command Records Supported Get Command.

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 2nd network node.
1	The specific commands associated with the grouping can be controlled and monitored from a 2nd network node. This option includes also the level field used by the command *Transfer Scene* in the *Controller Replication Command* Class. In this case the level value is transferred via the command *Basic Set* in the *Basic Command Class*.

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 *Basic Set* in the *Basic Command Class*. Level field originates from the command *Transfer Scene* in the *Controller Replication Command* Class.

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 Switch Multilevel Set, but not a Switch Multilevel Start Level Change.

Switch Multilevel Set has a command length of 3

Switch Multilevel Start Level Change has a command length of 4

Free Command Records 1-2 (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 1-2 (16 bits)

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

3.8.3 Command Configuration Set Command

The Command Configuration Set Command is 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 multiple command records address same Grouping identifier and Node ID, it is the responsibility of the originator of the Command Configuration Set, that the related commands all specifies unique instances for Multi instance devices.

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 bit)

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

Node ID (8 bit)

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

Command length (8 bit)

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 bit)

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 bit)

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 byte1 .. Command byte n

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.

3.8.4 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.

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 bit)

This group identifier is 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 bit)

This field specifies the node ID within the grouping.

3.8.5 Command Configuration Report Command

The Command Configuration Report Command is 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 bit)

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

Node ID (8 bit)

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

Reports to follow (4 bit)

The value indicates how many report frames there is left before the entire list of commands is transferred.

First (1 bit)

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

Command length (8 bit)

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 bit)

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 bit)

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 byte1 .. Command byte n

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

3.9 Basic Command Class, version 1

All devices will if possible support the Basic Commands Class. This class contains a small number of very basic commands that can be used to control the basic functionality of a device. The Commands include the possibility to set a given level, get a given level and report a level.

The basic Commands enables a controller application to use the basic Commands on a device that is unknown to the controller and thereby give the user control over the main functionality of the device. The actual usage of the basic Commands is described for each generic/specific device.

In case you have a device with multi instances one of the instances should be accessible using the Basic Command Class directly. The Basic Command Class can also be encapsulated in the Multi instance Command Class to address the individual instances. It is required for devices to support Basic Commands that are encapsulated in the Multi instance Command. A controlling device is not required to be able to send Basic Commands that are encapsulated in the Multi instance Command.

Typically, generic and/or specific device classes define mappings from the Basic Command Class to specific Commands. A device shall implement the mappings as specified in the generic and/or specific device classes that are published by the device in its node info frame. Specifications for the mapping of the Basic Command Class in the corresponding specific device class have precedence over specifications in the generic device class.

NOTE: To avoid unintentionally operation of devices it is recommended not using broadcasts for the Basic Command Class because many devices will support this command class.

3.9.1 Basic Set Command

The Basic Set Command will be used by the different devices, to set dim level, temperature, state, water level, speed etc., in a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_BASIC							
Command = BASIC_SET							
Value							

Value (8 bit)

The value field can be used to set different levels in the device.

A controlling device may use any of the values 0…255 (0x00…0xFF) in a Set Command. Since the Z-Wave compliant mapping of the Basic Command may define that certain values are ignore by the target device that received a Basic Set Command from a controlling device, a controlling device shall not assume that the device will always act on any Basic Set Command. It is the responsibility of the device that is controlled with a Basic Set Command to ignore reserved / undefined values.

A device that is controlled by Basic Commands shall implement Basic Set as specified in the corresponding generic / specific device class. If there is no further specification beyond the definition of a mapping to another Command defined, the device shall provide exactly the same behavior, as if the set Command would have been given with that Command.

3.9.2 Basic Get Command

The Basic Get Command will be used by the different devices to get dim level, temperature, state, water level, speed etc., from a device.

Devices shall implement Basic Get and Basic Report as specified in the corresponding generic / specific device class. If there is no further explanation beyond the definition of a mapping to another Command defined, the device shall provide exactly the same behavior, as if the set Command would have been given with that Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_BASIC							
Command = BASIC_GET							

3.9.3 Basic Report Command

The Basic Report Command will be used by the different devices to report dimming level, temperature, state, water level, speed etc., from a device. The Basic Report Command can be sent unsolicited or requested by the Basic Get Command.

A device shall respond to a Basic Get always with a Basic Report, i.e. not with the Report Command of the mapped-to device class.

A controlling device must be able to accept any of the values 0…255 (0x00…0xFF) in a Basic Report. It is beyond the scope of the Z-Wave specification if and which values received with a Basic Report a controlling device may ignore.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_BASIC							
Command = BASIC_REPORT							
Value							

Value (8 bit)

The value field can be use to report different levels in the device.

3.10 Basic Window Covering Command Class, version 1

It is not recommended to use this command class for new devices.

Instead it is recommended to base devices of this category on the Multi-level switch generic device class with the Multi-position Motor specific device class.

This section contains Commands that can be used to control a Basic Window Covering Command Class.

3.10.1 Basic Window Covering Start Level Change Command

The Basic Window Covering Start Level Change Command is used to start moving drapes, shades, blinds in a given direction. The speed of the movement is implementation specific.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_BASIC_WINDOW_COVERING							
Command = BASIC_WINDOW_COVERING_START_LEVEL_CHANGE							
Reserved	Open/ Close	Reserved					

Open/Close (1 bit)

If the Open/Close bit is set to 0 the window covering should open. If field is set to 1 the window covering should close.

Reserved

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

3.10.2 Basic Window Covering Stop Level Change Command

The Basic Window Covering Stop Level Change Command is used to stop moving drapes, shades, blinds in a given direction.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_BASIC_WINDOW_COVERING							
Command = BASIC_WINDOW_COVERING_STOP_LEVEL_CHANGE							

3.11 Battery Command Class, version 1

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

3.11.1 Battery Level Get Command

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

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

3.11.2 Battery Level Report Command

The Battery Level Report Command is used to report the battery level of a battery operated device. The Battery Level Report Command can be sent unsolicited or requested by the Battery Level Get Command.

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

Battery Level (8 bit)

The battery level is reported as a percentage of the full battery. The field can take values from 0 to 100% (0x00 – 0x64). The value 0xFF indicates a battery low warning.

3.12 Binary Sensor Command Class, version 1

The Binary Sensor Command Class can be used to realize binary sensors, such as movement sensors.

3.12.1 Binary Sensor Get Command

The Binary Sensor Get Command can be used to request the status of a sensor.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SENSOR_BINARY							
Command = SENSOR_BINARY_GET							

3.12.2 Binary Sensor Report Command

The Binary Sensor Report Command can be sent unsolicited or requested by the Binary Sensor Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SENSOR_BINARY							
Command = SENSOR_BINARY_REPORT							
Sensor Value							

Sensor Value (8 bit)

If the Sensor value is 0x00 indicates that the sensor is idle and 0xFF indicates that the sensor has detected an event.

3.13 Binary Switch Command Class, version 1

This chapter describes the Commands that can be used to make binary switches. These Commands allow applications to set and get the status of a binary switch.

3.13.1 Binary Switch Set Command

The Binary Switch Set Command can be used to set a device on or off (enable or disable).

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_BINARY							
Command = SWITCH_BINARY_SET							
Switch Value							

Value (8 bit)

The value can be either 0x00 (off/disable) or 0xFF (on/enable). The values from 1 to 99 (0x01 – 0x63) shall mapped to 0xFF upon receipt of the Command in the device.

All other values are reserved and shall be ignored by the receiving device.

Controlling devices may send any of the values 0x00…0x63 and 0xFF to the device. Controlling devices must not send any of the reserved values to the device.

Note:
Based on this specification of the Binary Switch Commands and the Basic Command Class mappings for the corresponding generic / specific devices, sending a single Basic Set Command to control a mixed group of Binary and Multilevel Switches is possible. Upon receipt of such a Command, enables that the Binary Switches would simply turn ON, while Multilevel Switches would turn to the selected level.

3.13.2 Binary Switch Get Command

The Binary Switch Get Command can be used to get the status of the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_BINARY							
Command = SWITCH_BINARY_GET							

3.13.3 Binary Switch Report Command

The Binary Switch Report Command can be sent unsolicited or requested by the Binary Switch Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_BINARY							
Command = SWITCH_BINARY_REPORT							
Value							

Value (8 bit)

The value can be either 0x00 (off/disable) or 0xFF (on/enable). Devices must not respond with a Binary Switch Report with any other value.

3.14 Binary Toggle Switch Command Class, version 1

Do not use this command class for new devices.

Use instead Binary Switch Generic Device Class for such devices.

This chapter describes the Commands that can be used to make binary toggle switches. These Commands allow applications to set and get the status of a binary toggle switch.

3.14.1 Binary Toggle Switch Set Command

The Binary Toggle Switch Set Command can be used to toggle a device e.g. from on to off and from off to on.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_TOGGLE_BINARY							
Command = SWITCH_TOGGLE_BINARY_SET							

3.14.2 Binary Toggle Switch Get Command

The Binary Switch Get Command can be used to request the state of the load controlled by the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_TOGGLE_BINARY							
Command = SWITCH_TOGGLE_BINARY_GET							

3.14.3 Binary Toggle Switch Report Command

The Binary Toggle Switch Report Command can be sent unsolicited or requested by the Binary Toggle Switch Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_TOGGLE_BINARY							
Command = SWITCH_TOGGLE_BINARY_REPORT							
Value							

Value (8 bit)

The value can be either 0x00 (off) or 0xFF (on).

3.15 Climate Control Schedule Command Class, version 1

The Climate Control Schedule Command Class allows devices to exchange schedules and overrides, which specify when to perform a setback on the setpoint.

Note: The setpoint is the temperature a device will try to maintain. The setback is a deviation from the setpoint. When a setback is in use the device will apply the setback to the setpoint, resulting in a different temperature. When using schedules and overrides it is possible to define several setbacks occurring at specific times.

Schedules shall be exchanged using the Schedule Commands.
Overrides of schedules shall be exchanged using the Schedule Override Commands.
Detection of updated schedules shall be done using the Schedule Changed Commands.

The Climate Control Schedule uses the Schedule State type to define each setback. The Schedule State type has the following format:

7	6	5	4	3	2	1	0
Schedule State							

Schedule State (8 bit)

The values are as follows:

Schedule State		Description
Hexadecimal	**Decimal**	
0x80 … 0xFF 0x00 0x01 … 0x78	-128 … -1 0 1 … 120	The setback in 1/10 degrees (Kelvin) Example: 0 = 0 degrees setback 1 = 0.1 degrees is added to the setpoint 2 = 0.2 degrees is added to the setpoint -1 = 0.1 degrees is subtracted from the setpoint -2 = 0.2 degrees is subtracted from the setpoint
0x79	121	Frost Protection
0x7A	122	Energy Saving Mode
0x7B – 0x7E	123 – 126	Reserved
0x7F	127	Unused State

When converting between Celsius and Fahrenheit proper rounding must be applied with at least two decimals in the internal calculations of a device to avoid rounding errors.

When displaying converted Fahrenheit values it is recommended that the displayed value is rounded to nearest quarter of a degree.

3.15.1 Schedule Set Command

The Schedule Set Command is used to set the climate control schedule in a device for a specific weekday. A climate control schedule defines when to use a setback on the setpoint in a device. A schedule can hold a maximum of 9 switchpoints. A switchpoint defines one setback from the current setpoint.

The entire list of switchpoints in the Command must be ordered by time, ascending from 00:00 towards 23:59. Switchpoints which have a Schedule State set to “Unused” shall be placed last. No duplicates shall be allowed for Switchpoints which have a Schedule State different from “Unused”.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CLIMATE_CONTROL_SCHEDULE							
Command = SCHEDULE_SET							
Reserved					Weekday		
Switchpoint 0 Byte 1							
Switchpoint 0 Byte 2							
Switchpoint 0 Byte 3							
Switchpoint 1 Byte 1							
Switchpoint 1 Byte 2							
Switchpoint 1 Byte 3							
…							
Switchpoint 8 Byte 1							
Switchpoint 8 Byte 2							
Switchpoint 8 Byte 3							

Weekday (3 bit)

The possible values are:

Binary	Decimal	Description
0b000	0	Reserved
0b001	1	Monday
0b010	2	Tuesday
0b011	3	Wednesday
0b100	4	Thursday
0b101	5	Friday
0b110	6	Saturday
0b111	7	Sunday

Switchpoint (24 bit)

7	6	5	4	3	2	1	0	
Reserved			Hour					Byte 1
Reserved		Minute						Byte 2
Schedule State								Byte 3

Hour (5 bit):

Specifies the hour during a day when this switchpoint shall be used. Possible values are:

Binary	Decimal	Description
0b00000	0	0 hour
0b00001	1	1st hour
0b00010	2	2nd hour
…	…	…
0b10111	23	23rd hour
0b11000	24	Reserved
…	…	Reserved
0b11111	31	Reserved

Minute (6 bit):

Specifies the minute during the specified hour when this switchpoint shall be used. Possible values are:

Binary	Decimal	Description
0b000000	0	0 minute
0b000001	1	1st minute
0b000010	2	2nd minute
…	…	…
0b111011	59	59th minute
0b111100	60	Reserved
…	…	Reserved
0b111111	63	Reserved

Schedule State (8 bit):

Schedule State uses the Schedule State type format, see section 3.15. If Schedule State has the value of “Unused”, then the Hour and Minute field shall be ignored. Once a Schedule State of “Unused” is encountered, the parsing of switchpoints shall stop.

3.15.2 Schedule Get Command

The Schedule Get Command is used to request the climate control schedule in a device for a specific weekday.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CLIMATE_CONTROL_SCHEDULE							
Command = SCHEDULE_GET							
Reserved					Weekday		

Weekday (3 bit)

The possible values are:

Binary	Decimal	Description
0b000	0	Reserved
0b001	1	Monday
0b010	2	Tuesday
0b011	3	Wednesday
0b100	4	Thursday
0b101	5	Friday
0b110	6	Saturday
0b111	7	Sunday

3.15.3 Schedule Report Command

The Schedule Report Command is used to report the climate control schedule in a device for a specific weekday.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CLIMATE_CONTROL_SCHEDULE							
Command = SCHEDULE_REPORT							
Reserved					Weekday		
Switchpoint 0 Byte 1							
Switchpoint 0 Byte 2							
Switchpoint 0 Byte 3							
Switchpoint 1 Byte 1							
Switchpoint 1 Byte 2							
Switchpoint 1 Byte 3							
…							
Switchpoint 8 Byte 1							
Switchpoint 8 Byte 2							
Switchpoint 8 Byte 3							

Weekday (3 bit)

The possible values are:

Binary	Decimal	Description
0b000	0	Reserved
0b001	1	Monday
0b010	2	Tuesday
0b011	3	Wednesday
0b100	4	Thursday
0b101	5	Friday
0b110	6	Saturday
0b111	7	Sunday

Switchpoint (24 bit)

7	6	5	4	3	2	1	0	
Reserved			Hour					Byte 1
Reserved		Minute						Byte 2
Schedule State								Byte 3

Hour (5 bit):

Specifies the hour during a day when this switchpoint shall be used. Possible values are:

Binary	Decimal	Description
0b00000	0	0 hour
0b00001	1	1st hour
0b00010	2	2nd hour
…	…	…
0b10111	23	23rd hour
0b11000	24	Reserved
…	…	Reserved
0b11111	31	Reserved

Minute (6 bit):

Specifies the minute during the specified hour when this switchpoint shall be used. Possible values are:

Binary	Decimal	Description
0b000000	0	0 minute
0b000001	1	1st minute
0b000010	2	2nd minute
…	…	…
0b111011	59	59th minute
0b111100	60	Reserved
…	…	Reserved
0b111111	63	Reserved

Schedule State (8 bit):

Schedule State uses the Schedule State type format, see section 3.15.
If Schedule State has the value of “Unused”, then the Hour and Minute field shall be ignored. Once a Schedule State of “Unused” is encountered, the parsing of switchpoints shall stop.

3.15.4 Schedule Changed Get Command

The Schedule Changed Get Command is used to check if the climate control schedule has changed.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CLIMATE_CONTROL_SCHEDULE							
Command = SCHEDULE_CHANGED_GET							

3.15.5 Schedule Changed Report Command

The Schedule Changed Report Command is used to report if the climate control schedule has changed.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CLIMATE_CONTROL_SCHEDULE							
Command = SCHEDULE_CHANGED_REPORT							
ChangeCounter							

ChangeCounter (8 bit)

The ChangeCounter is a timestamp for a climate control schedule and it is kept in devices which exchange climate control schedules.

One device holds a climate control schedule and other devices uses this climate control schedule. Whenever the climate control schedule changes, the device which holds it shall internally update its ChangeCounter, and the other devices shall regularly use the Schedule Changed Get Command on the device which holds the Climate Control Schedule to see if the ChangeCounter is different from last time – indicating a change in a climate control schedule.

The possible values are:

Hexadecimal	Decimal	Description
0x00	0	The climate control schedule change mechanism is temporarily disabled by the override function.
0x01 … 0xFF	1 - 255	The climate control schedule change mechanism is enabled. ChangeCounter is incremented by one every time climate control schedule changes. When ChangeCounter eventually reach 0xFF, then the next increment, will rollover to 0x01.

When a device is fresh and has no climate control schedule it shall retrieve a climate control schedule using the Schedule Get Command and it shall also use the Schedule Changed Get to get the first copy of the current ChangeCounter, thus avoiding getting the climate control schedule initially twice.

When a device is awake after sleep mode it should use this Command to detect if the schedule has been changed.

3.15.6 Schedule Override Set Command

The Schedule Override Set Command is used to set the override in a device.

The purpose of an override is to inform a device to ignore its current climate control schedule and assume the setting provided by the Override Type and Override State fields.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CLIMATE_CONTROL_SCHEDULE							
Command = SCHEDULE_OVERRIDE_SET							
Reserved						Override Type	
Override State							

Override Type (2 bit)

The override type field can assume the following values:

Binary	Decimal	Description
0b00	0	No override
0b01	1	Temporary override
0b10	2	Permanent override
0b11	3	Reserved

Note: The difference between a temporary and a permanent override is that a temporary override only overrides the current switchpoint in the climate control schedule.

Both temporary and permanent overrides may be cancelled in the device, which receives the SCHEDULE_OVERRIDE_SET. This cancellation shall be notified in an unsolicited SCHEDULE_OVERRIDE_REPORT as specified in the following sequence diagram:

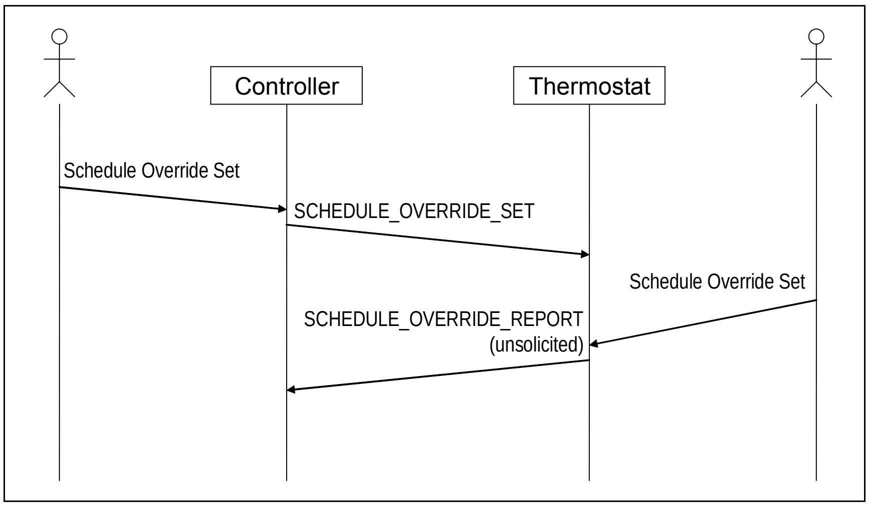


Figure 3, Sequence diagram for cancellation of a Schedule Override Set

Override State (8 bit)

The Override State uses the Schedule State type format, see section 3.15

3.15.7 Schedule Override Get Command

The Schedule Override Get Command is used to request the override, currently in use in a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CLIMATE_CONTROL_SCHEDULE							
Command = SCHEDULE_OVERRIDE_GET							

3.15.8 Schedule Override Report Command

The Schedule Override Report Command is used to report the override, currently in use in a device. This report can also be sent unsolicited.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CLIMATE_CONTROL_SCHEDULE							
Command = SCHEDULE_OVERRIDE_REPORT							
Reserved						Override Type	
Override State							

Override Type (2 bit)

The override type field can assume the following values:

Binary	Decimal	Description
0b00	0	No override
0b01	1	Temporary override
0b10	2	Permanent override
0b11	3	Reserved

Note: The difference between a temporary and a permanent override is that a temporary override only overrides the current switchpoint in the climate control schedule.

Both temporary and permanent overrides may be cancelled in the device, which receives the SCHEDULE_OVERRIDE_SET. This cancellation shall be notified in an unsolicited SCHEDULE_OVERRIDE_REPORT as shown on figure in section 3.15.6

Override State (8 bit)

The Override State uses the Schedule State type format, see section 3.15

3.16 Clock Command Class, version 1

The Clock Command Class can be used to implement a simple clock functionality that can be used to displaying time or creating timers.

3.16.1 Clock Set Command

The Clock Set Command is used to set the clock in a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CLOCK							
Command = CLOCK_SET							
Weekday			Hour				
Minute							

Weekday (3 bit)

The weekday field can take the following values 0 = Unused (24 hour clock), 1 = Monday, 2 = Tuesday, 3 = Wednesday, 4 = Thursday, 5 = Friday, 6 = Saturday and 7 = Sunday.

Hour (5 bit)

The hour field can take values from 0 to 23.

Minute (8 bit)

The minute field can take values from 0 to 59.

3.16.2 Clock Get Command

The Clock Get Command is used to request the clock report from a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CLOCK							
Command = CLOCK_GET							

3.16.3 Clock Report Command

The Clock Report Command is used to report the actual weekday and clock in a device. The Clock Report Command can be sent unsolicited or requested by the Clock Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CLOCK							
Command = CLOCK_REPORT							
Weekday			Hour				
Minute							

Weekday (3 bit)

The weekday field can take the following values 0 = Unused (24 hour clock), 1 = Monday, 2 = Tuesday, 3 = Wednesday, 4 = Thursday, 5 = Friday, 6 = Saturday and 7 = Sunday.

Hour (5 bit)

The hour field can take values from 0 to 23.

Minute (8 bit)

The minute field can take values from 0 to 59.

3.17 Configuration Command Class, version 1

With the Configuration Command Class it's possible to change the default factory settings in a device. This could for example be the dimming rate in a lighting dimmer device. When implementing this class in a controller it's recommended to be able to set all parameters manually. Since the content of the configuration parameters are not standardized in the Z-Wave framework, it's the vendors responsibility to document this functionality in the products user manual (or an installer manual).

NOTE: All Z-Wave enabled devices must be able to operate based on the default factory setting.

3.17.1 Configuration Set Command

The Configuration Set Command is used to set the value of configuration parameter(s).

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CONFIGURATION							
Command = CONFIGURATION_SET							
Parameter Number							
Default	Reserved				Size		
Configuration Value 1							
Configuration Value 2							
…							
Configuration Value N							

Parameter Number (8 bit)

The parameter number field specifies which configuration parameter is being set. The parameter numbers are specific for each application.

Default (1 bit)

If the default bit is set to 1 the device is set to default factory setting and the configuration values is ignored. If the default bit is set to 0 then the configuration values is used.

Reserved (4 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Size (3 bit)

The size field indicates the number of bytes that is used for the configuration value. This field can take values 1 (001b), 2 (010b) or 4 (100b).

Configuration Value 1 … Configuration Value N (variable)

The configuration value is a signed field. The field can be 1, 2 or 4 bytes in size. The first byte is the most significant byte. The table below shows signed decimal values together with their hexadecimal equivalents.

Signed 1 byte decimal value	Hexadecimal
127	0x7F
25	0x19
2	0x02
1	0x01
0	0x00
-1	0xFF
-2	0xFE
-25	0xE7
-128	0x80

Signed 2 bytes decimal value	Hexadecimal
32767	0x7FFF
1025	0x0401
2	0x0002
1	0x0001
0	0x0000
-1	0xFFFF
-2	0xFFFE
-1025	0xFBFF
-32768	0x8000

3.17.2 Configuration Get Command

This Configuration Get Command is used to get the value of a configuration parameter.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CONFIGURATION							
Command = CONFIGURATION_GET							
Parameter Number							

Parameter Number (8 bit)

The parameter number field specifies which configuration parameter is being requested. The parameter numbers are specific for each application.

3.17.3 Configuration Report Command

This Configuration Report Command is used to report the actual value of a given configuration parameter in the device. The Configuration Report Command can be sent unsolicited or requested by the Configuration Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CONFIGURATION							
Command = CONFIGURATION_REPORT							
Parameter Number							
Reserved					Size		
Configuration Value 1							
Configuration Value 2							
..							
Configuration Value n							

Refer to explanation under the Configuration Set Command.

3.18 Configuration Command Class, version 2

Configuration Command Class v2 enables the device to exchange up to 65.535 product specific configuration parameters in the Z-Wave Interoperability community. In addition it is possible to set multiple configuration parameters with one Command.

It is mandatory for Z-Wave™ devices supporting Configuration Command Class v2 to also support Configuration Command Class v1; in order to communicate with Z-Wave™ enabled devices that only support v1. Therefore Z-Wave™ devices advertising Configuration Command Class v2 in the NIF shall implicit support Configuration Command Class v1.

NOTE! All Z-Wave™ enabled devices must be able to operate based on default factory settings. Supported configuration parameters must be a sequence starting from one (1).

3.18.1 Configuration Bulk Set Command

The Configuration Bulk Set Command is used to set the value of configuration parameter(s).

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CONFIGURATION							
Command = CONFIGURATION_BULK_SET							
Parameter Offset MSB							
Parameter Offset LSB							
Number of Parameters							
Default	Hand-shake	Reserved			Size		
Parameter 1 – Configuration Value 1 (MSB)							
…							
Parameter 1 – Configuration Value N (LSB)							
…							
Parameter N – Configuration Value 1 (MSB)							
…							
Parameter N – Configuration Value N (LSB)							

Parameter Offset MSB + LSB (16 bit)

The parameter offset is a 16 bit value to address the offset to a parameter. E.g. if parameter offset = 900, then depended on the parameter number field, the first parameter to be addressed is 900.

Number of Parameters (8 bit)

The Number of Parameters field specifies the number of configuration parameters that will be addressed with reference to the parameter offset. Valid values are 1-255 e.g. Number of Parameters = 3 and Parameter Offset = 900 then parameters 900, 901 and 902 are being addressed.

Default (1 bit)

If the default bit is set to 1 the device i.e. all configuration parameters are set to default factory settings and the parameter values are ignored. If the default bit is set to 0 then the configuration values are used. Note that when required the configuration values can be overwritten by default values at any time.

Handshake (1 bit)

If the Handshake bit is set to 1 the receiving device must reply with a Configuration Bulk Report containing the same configuration parameter numbers (Parameter Offset + Number of Parameters), reporting that data has been stored in non-volatile memory for the requested configuration parameters and the receiver is now ready for next Command. In case the Configuration Bulk Report is missing despite successful transmission of Configuration Bulk Set, the transmitter must wait one second before sending the next Configuration Bulk Set Command.
If the Handshake bit is set to 0, then no Configuration Bulk Report is requested immediately after the Set Command.

Reserved (3 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Size (3 bit)

The size field indicates the number of bytes that are used for the configuration value. This field can take values 1 (001b), 2 (010b) or 4 (100b).

Parameter 1 – Configuration Value 1 … Parameter N – Configuration Value N (variable)

The parameter[x] – data[x] is a signed field. The field can be 1, 2 or 4 bytes in size. The first byte is the most significant byte. The table below shows signed decimal values together with their hexadecimal equivalents.

Decimal	Hexadecimal
127	0x7F
1	0x01
0	0x00
-1	0xFF
-128	0x80

Decimal	Hexadecimal
32767	0x7FFF
1	0x0001
0	0x0000
-1	0xFFFF
-32768	0x8000

3.18.2 Configuration Bulk Get Command

The Configuration Bulk Get Command is used to get the value of configuration parameter(s).

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CONFIGURATION							
Command = CONFIGURATION_BULK_GET							
Parameter Offset MSB							
Parameter Offset LSB							
Number of Parameters							

Parameter Offset MSB + LSB (16 bit)

See Configuration Bulk Set Command description.

Number of Parameters (8 bit)

See Configuration Bulk Set Command description.

3.18.3 Configuration Bulk Report Command

The Configuration Bulk Report Command is used to report the actual value of the requested configuration parameter(s). The Configuration Bulk Report Command can be send as a result of receiving a Configuration Bulk Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CONFIGURATION							
Command = CONFIGURATION_BULK_REPORT							
Parameter Offset MSB							
Parameter Offset LSB							
Number of Parameters							
Reports to follow							
Default	Hand-shake	Reserved			Size		
Parameter 1 – Configuration Value 1 (MSB)							
…							
Parameter 1 – Configuration Value N (LSB)							
…							
Parameter N – Configuration Value 1 (MSB)							
…							
Parameter N – Configuration Value N (LSB)							

Parameter Offset MSB + LSB (16 bit)

The Parameter Offset is a 16 bit value to address the offset to a parameter. E.g. if Parameter Offset = 900, then depended on the Number of Parameters field, the first parameter in the report is 900.

Number of Parameters (8 bit)

The Number of Parameters field specifies the number of configuration parameters that are presented in this report with reference to the Parameter Offset. Valid values are 1-255 e.g. if Number of Parameters = 2 and Parameter Offset = 900, then Configuration Parameter 900 and 901 are being reported.

Report to follow (8 bit)

This value indicates how many report frames there are left before all requested configuration parameters values have been transferred. 0 indicates no more Configuration Bulk Reports to follow.

Default (1 bit)

If default parameter is 1, then all returned configuration parameter values are in factory default state.

Handshake (1 bit)

If Handshake is 1, then this report indicates the receiver is ready for next frame.

Size (3 bit)

The size field indicates the number of bytes that are used for the configuration value. This field can take values 1 (001b), 2 (010b) or 4 (100b). All parameters in the same report must be of the same size.

Parameter 1 – Configuration Value 1 … Parameter N – Configuration Value N (variable)

See Configuration Bulk Set Command description.

3.19 Controller Replication Command Class, version 1

The Controller Replication Command Class contains Commands that can be used to copy scene and group data to another controller. The Command Class must only be used in conjunction with a controller shift or when including a new controller to the network. It's optional to use this command class during a controller shift or when including a new controller to the network. The API call ZW_ControllerChange is used in a controller shift and ZW_AddNodeToNetwork when including a new controller to the network [2].

Devices supporting this Command Class should accept all the Commands. If some Commands are not used in the particular implementation, then they should be ignored.

The API call ZW_ReplicationSend must be used by the sending controller when transferring the group and scene command classes to another controller. The API call ZW_ReplicationReceiveComplete must be used by the receiving controller as acknowledge on application level because the data must first be stored in non-volatile memory before it can receive the next group or scene data.

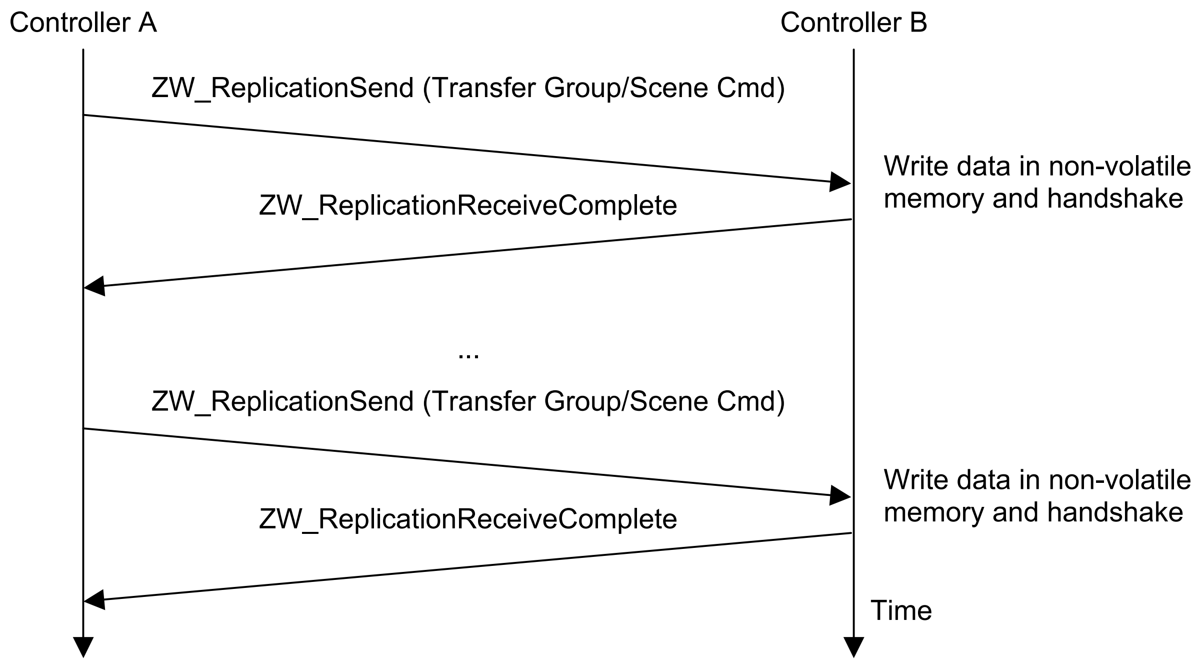

Controller A
Controller B
ZW_ReplicationSend (Transfer Group/Scene Cmd)
Write data in non-volatile memory and handshake
ZW_ReplicationReceiveComplete
...
ZW_ReplicationSend (Transfer Group/Scene Cmd)
Write data in non-volatile memory and handshake
ZW_ReplicationReceiveComplete
Time

Figure 4, Controller Replication sequence

A controller not supporting the Controller Replication Command Class must implement the acknowledge on application level when receiving Controller Replication Commands to avoid that the sending controller is locked due to a missing acknowledge on application level. The receiving controller will then ignore the content of the Controller Replication Commands but acknowledge on application level using the API call ZW_ReplicationReceiveComplete.

3.19.1 Transfer Group Command

The Transfer Group Command is used to replicate mappings between Group ID and Node ID.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CONTROLLER_REPLICATION							
Command = CTRL_REPLICATION_TRANSFER_GROUP							
Sequence Number							
Group ID							
Node ID							

Sequence Number (8 bit)

The sequence number is used by the Z-Wave protocol and should not be filled in by the sending device or evaluated by the receiving device.

Group ID (8 bit)

Group ID of the group that the node is member of.

Node ID (8 bit)

Node ID of slave device.

3.19.2 Transfer Group Name Command

The Transfer Group Name Command is used to replicate group names.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CONTROLLER_REPLICATION							
Command = CTRL_REPLICATION_TRANSFER_GROUP_NAME							
Sequence Number							
Group ID							
Group Name 1							
Group Name 2							
Group Name 3							
…							
Group Name n							

Sequence Number (8 bit)

The sequence number is used by the Z-Wave protocol and should not be filled in by the sending device or evaluated by the receiving device.

Group ID (8 bit)

Group ID associated with a specific group.

Group Name (n bytes)

The Group Name fields contain the assign group name in ASCII characters.

3.19.3 Transfer Scene Command

The Transfer Scene Command is used to replicate mappings between Scene ID and Node ID.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CONTROLLER_REPLICATION							
Command = CTRL_REPLICATION_TRANSFER_SCENE							
Sequence Number							
Scene ID							
Node ID							
Level							

Sequence Number (8 bit)

The sequence number is used by the Z-Wave protocol and should not be filled in by the sending device or evaluated by the receiving device.

Scene ID (8 bit)

The scene ID is the parameter used to link together the different devices that takes part of a scene.

Node ID (8 bit)

The Node ID for a device that is part of the scene.

Level (8 bit)

The level is the parameter used for the specified scene.

3.19.4 Transfer Scene Name Command

The Transfer Scene Name Command is used to replicate scene names.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CONTROLLER_REPLICATION							
Command = CTRL_REPLICATION_TRANSFER_SCENE_NAME							
Sequence Number							
Scene ID							
Scene Name 1							
Scene Name 2							
Scene Name 3							
…							
Scene Name n							

Sequence Number (8 bit)

The sequence number is used by the Z-Wave protocol and should not be filled in by the sending device or evaluated by the receiving device.

Scene ID (8 bit)

Scene ID associated with a specific scene.

Scene Name (n bytes)

The Scene Name fields contain the assign scene name in ASCII characters.

3.20 Door Lock Command Class, version 1

The Door Lock Command Class is used to secure/unsecure a lock type as well as setting the configuration of an advanced Z-Wave™ door lock device.

3.20.1 Door Lock Operation Set Command

The Door Lock Operation Set Command is used to set the operation mode of the lock in a Z-Wave™ enabled door lock device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_DOOR_LOCK							
Command = DOOR_LOCK_OPERATION_SET							
Door Lock Mode							

Door Lock Mode (8 bits)

Door Lock Mode will set the door lock device in unsecured or secured mode as well as other peripheral settings.

Hexadecimal	Description
0x00	Door Unsecured 1)
0x01	Door Unsecured with timeout 2)
0x10	Door Unsecured for inside Door Handles 1)
0x11	Door Unsecured for inside Door Handles with timeout 2)
0x20	Door Unsecured for outside Door Handles 1)
0x21	Door Unsecured for outside Door Handles with timeout 2)
0xFF	Door Secured

1) Constant mode. Door will be unsecured until set back to secured mode by Command.
2) Timeout mode. Fallback to secured mode after timeout has expired (set by Door Lock Configuration Set).

3.20.2 Door Lock Operation Get Command

The Door Lock Operation Get Command is used to request the state of the door lock device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_DOOR_LOCK							
Command = DOOR_LOCK_OPERATION_GET							

3.20.3 Door Lock Operation Report

The Door Lock Operation Report Command can be used by a door lock device to send a report either unsolicited or requested by the Door Lock Operation Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_DOOR_LOCK							
Command = DOOR_LOCK_OPERATION_REPORT							
Door Lock Mode							
Outside Door Handles Mode				Inside Door Handles Mode			
Door Condition							
Lock Timeout Minutes							
Lock Timeout Seconds							

Door Lock Mode (8 bits)

See Door Lock Operation Set Command description. In case timeout mode is not supported, this field must display other valid mode from the Door Lock Operation Set Command e.g. 0x10 instead of 0x11.

Outside/Inside Door Handles Mode (4 bits)

These parameters indicate the activity of the door handles i.e. which handle(s) has opened the door lock.

Bit	Outside Door Handles Mode (4 bits)	Inside Door Handles Mode (4 bits)
0	0 = Handle 1 inactive; 1 = Handle 1 active	0 = Handle 1 inactive; 1 = Handle 1 active
1	0 = Handle 2 inactive; 1 = Handle 2 active	0 = Handle 2 inactive; 1 = Handle 2 active
2	0 = Handle 3 inactive; 1 = Handle 3 active	0 = Handle 3 inactive; 1 = Handle 3 active
3	0 = Handle 4 inactive; 1 = Handle 4 active	0 = Handle 4 inactive; 1 = Handle 4 active

Door Condition (8 bits)

The Door Condition field indicates the hardware status of the door lock device such as bolt and latch states.

Bit	Description
0	0 = Door Open; 1 = Door Closed
1	0 = Bolt Locked; 1 = Bolt Unlocked
2	0 = Latch Open; 1 = Latch Closed
3-7	Reserved

Lock Timeout Minutes (8 bits)

If a Lock Timeout has been set in DOOR_LOCK_CONFIG_SET Command and the door lock state has been set to unsecure with timeout by means of DOOR_LOCK_OPERATION_SET, this field shall display the remaining minutes the lock is in unsecured state. In case no timeout has been set this field shall always display zero. This field shall be set to 0xFE when timeout is not supported by the door lock device.

Lock Timeout Seconds (8 bits)

If a Lock Timeout has been set in DOOR_LOCK_CONFIG_SET Command and the door lock state has been set to unsecure with timeout by means of DOOR_LOCK_OPERATION_SET, this field shall display the remaining seconds the lock is in unsecured state. In case no timeout has been set this field shall always display zero. This field shall be set to 0xFE when timeout is not supported by the door lock device.

3.20.4 Door Lock Configuration Set

The Door Lock Configuration Set Command is used to set the configuration of the door lock device including operation type and operation timers.

Note: All Z-Wave enabled devices must be able to operate based on factory default settings i.e. an end-user should not be forced to setup the configuration parameters for the door lock to operate.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_DOOR_LOCK							
Command = DOOR_LOCK_CONFIGURATION_SET							
Operation Type							
Outside Door Handles State				Inside Door Handles State			
Lock Timeout Minutes							
Lock Timeout Seconds							

Operation Type (1 byte)

The Operation Type field can be set to either constant or timed operation. When timed operation is set, the Lock Timer Minutes and Lock Timer Seconds fields must be set to valid values.

Hexadecimal	Description
0x01	Constant operation
0x02	Timed operation
0x03 – 0XFF	*Reserved*

Outside/Inside Door Handles State (4 bits)

The Door Handles field is to enable or disable the door handlers that are implemented in the door lock device. For example there could be an inside as well as an outside door handler, whereas the inside door handler can be handled by Z-Wave Commands and the outside door handler will only unlock the door when successful authentication has been verified by e.g. a keypad.

Bit	Outside Door Handles State (4 bits)	Inside Door Handles State (4 bits)
0	0 = Handle 1 disabled; 1 = Handle 1 enabled	0 = Handle 1 disabled; 1 = Handle 1 enabled
1	0 = Handle 2 disabled; 1 = Handle 2 enabled	0 = Handle 2 disabled; 1 = Handle 2 enabled
2	0 = Handle 3 disabled; 1 = Handle 3 enabled	0 = Handle 3 disabled; 1 = Handle 3 enabled
3	0 = Handle 4 disabled; 1 = Handle 4 enabled	0 = Handle 4 disabled; 1 = Handle 4 enabled

Lock Timeout Minutes (1 byte)

Number of minutes the lock stays unsecured. If the Operation Type is set to Timed Operation, this field can be set accordingly to the valid values 1-254 decimal. If timeout is not supported for the door lock device then this shall be indicated with 0xFE. All other values not mentioned are reserved.

Lock Timeout Seconds (1 byte)

Number of seconds the lock stays unsecured. If the Operation Type is set to Timed Operation, this field can be set accordingly to the valid 1-59 decimal. If timeout is not supported for the door lock device then this shall be indicated with 0xFE. All other values not mentioned are reserved.

3.20.5 Door Lock Configuration Get Command

The Door Lock Configuration Get Command is used to request the configuration parameter of the door lock device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_DOOR_LOCK							
Command = DOOR_LOCK_CONFIGURATION_GET							

3.20.6 Door Lock Configuration Report Command

The Door Lock Configuration Report Command can be used by a door lock device send a report requested by the Door Lock Configuration Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_DOOR_LOCK							
Command = DOOR_LOCK_CONFIGURATION_REPORT							
Operation Type							
Outside Door Handles State				Inside Door Handles State			
Lock Timeout Minutes							
Lock Timeout Seconds							

See parameter description for DOOR_LOCK_CONFIGURATION_SET.

3.21 Energy Production Command Class, version 1

The Energy Production Command Class is used to retrieve various production data from the device.

3.21.1 Energy Production Get Command

The Energy production Get Command is used to request various production data from the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ENERGY_PRODUCTION							
Command = ENERGY_PRODUCTION_GET							
Parameter Number							

Parameter Number (8 bit)

The parameter number specifies the kind of production data to retrieve. Currently the following parameter numbers are defined:

Parameter Number	Description
0x00	Instant energy production
0x01	Total energy production
0x02	Energy production today
0x03	Total production time

This list may evolve in the future. In case a parameter number is not supported then it should be ignored.

3.21.2 Energy Production Report Command

The Energy Production Report is used to retrieve various production data from the device. The Energy Production Report Command can be send requested by the Energy Production Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ENERGY_PRODUCTION							
Command = ENERGY_PRODUCTION_REPORT							
Parameter Number							
Precision			Scale		Size		
Value 1							
Value 2							
…							
Value N							

Parameter Number (8 bit)

Refer to description under the Energy Production Get Command.

Precision (3 bit)

The precision field describes what the precision of the value is. The number indicates the number of decimals. The decimal value 1025 with precision 2 is therefore equal to 10.25.

Scale (2 bit)

The scale field indicates the scale used for the specified parameter number:

Parameter Number	Scale	Description
0x00	0x00	W
0x01	0x00	Wh
0x02	0x00	Wh
0x03	0x00	Seconds
	0x01	Hours

Size (3 bit)

The size field indicates the number of bytes that is used for the value. This field can take values from 1 (001b), 2 (010b) or 4 (100b).

Value 1 … Value N (variable)

The value is a signed field. The field can be 1, 2 or 4 bytes in size. The first byte is the most significant byte. The table below shows signed decimal values together with their hexadecimal equivalents.

Signed 1 byte decimal value	Hexadecimal
127	0x7F
25	0x19
2	0x02
1	0x01
0	0x00
-1	0xFF
-2	0xFE
-25	0xE7
-128	0x80

Signed 2 bytes decimal value	Hexadecimal
32767	0x7FFF
1025	0x0401
2	0x0002
1	0x0001
0	0x0000
-1	0xFFFF
-2	0xFFFE
-1025	0xFBFF
-32768	0x8000

3.22 Firmware Update Meta Data Command Class, version 1

The Firmware Update Meta Data Command Class used to update a device in the Z-Wave network with new firmware via the 40kbps RF communication link. It is recommended to enable the firmware update by some kind of physical authentication (e.g. activation of a pushbutton) on the device in question to avoid unintentional firmware updates.

The Firmware Update Meta Data Command Class is actually not updating the firmware in the device, but only used to store the firmware data temporary in memory e.g. the external EEPROM. The device will now calculate the checksum of the downloaded firmware data and compare it with the checksum received. In case the checksum is successfully verified the application will jump to a bootloader taking care of erasing the Flash memory and transferring the firmware data to the Flash memory. Refer to [3] regarding how to erase/write to Flash. The bootloader must be able to reside in Flash during this process and is therefore not altered when the firmware is updated. When the bootloader is finished, it will reset the module and the newly downloaded firmware is then executed. The bootloader must be located at the top of the memory map but the start address will be vendor specific depending on the complexity of the bootloader. Bootloader may support recovery of protocol/application data when updating firmware. Notice that protocol/application data can change location in the memory map depending on the library/application used. The Version Command Class can be used to verify that the intended firmware version is installed. Device should beside Version Command Class also support Manufacturer Specific Command Class to enable other devices to select the correct firmware data for the device in question.

A bootloader solution can be avoided in case the Z-Wave Single Chip has an appropriate interface to a host processor. In a two chip scenario the host processor can be used as temporary storage of the firmware data received. Afterwards can the host processor program the Flash on the Z-Wave Single Chip via the SPI interface and thereby avoiding a bootloader on the Z-Wave Single Chip.

The API call ZW_SendDataMeta must be used when streaming data to ensure that this traffic doesn't prevent control data from getting through in the network, especially important for 9.6kbps nodes because they can't detect 40kbps RF communication. Refer to [2] regarding a detailed description of the API call ZW_SendDataMeta.

Notice that it is not allowed to update firmware in a ZW0102 based module because it only supports 9.6kbps.

3.22.1 Firmware Meta Data Get Command

The Firmware Meta Data Get Command is used to request the current firmware in the device. This call is typically used to check whether or not the firmware needs to be updated.

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

3.22.2 Firmware Meta Data Report Command

The Firmware Meta Data Report Command is used to return status for the current firmware in the device. The Firmware Meta Data Report Command can only be sent requested by the Firmware Meta Data Get Command.

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

Manufacturer ID (16 bit)

The Manufacturer ID is a unique ID identifying the manufacturer of the device. The Manufacturer ID is assigned by Zensys upon request. Refer to Appendix A for a list of assigned Manufacturer ID’s. The first byte is the most significant byte.

Firmware ID (16 bit)

The Firmware ID is a unique identification of the firmware file when combined with the Manufacturer ID. Return zero as Firmware ID, in case no value is stored.The manufacturer assigns the Firmware ID. The first byte is the most significant byte.

Checksum (16 bit)

The checksum field is used to ensure consistency of the firmware data currently downloaded. Return zero as checksum, in case no value is stored. The first byte is the most significant byte. The checksum algorithm is implementation specific.

3.22.3 Firmware Update Meta Data Request Get Command

The Firmware Update Meta Data Request Get Command is used to request a firmware update. It is recommended to send the Command in conjunction with some kind of physical authentication on the device to be updated.

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 bit)

The Manufacturer ID is a unique ID identifying the manufacturer of the device. The Manufacturer ID is assigned by Zensys upon request. Refer to Appendix A for a list of assigned Manufacturer ID's. The first byte is the most significant byte.

Firmware ID (16 bit)

The Firmware ID is a unique identification of the firmware file when combined with the Manufacturer ID. The manufacturer assigns the Firmware ID. The first byte is the most significant byte. Bootloader will check that a valid firmware file is available based on the Manufacturer ID and Firmware ID.

Checksum (16 bit)

The checksum field is used to ensure consistency of the firmware data to be downloaded. The checksum algorithm is implementation specific.

3.22.4 Firmware Update Meta Data Request Report Command

The Firmware Update Meta Data Request Report Command is used to return status for the requested firmware update. The Firmware Update Meta Data Request Report Command is returned to the node ID which issued the Firmware Update Meta Data Request Get Command. The Firmware Update Meta Data Request Report Command can only be sent requested by the 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_REPORT							
Status							

Status (8 bit)

The Status identifier can return the following values:

Status	Description
0x00	Invalid combination of Manufacturer ID and Firmware ID. The device to be firmware updated will not start to request the firmware data.
0x01	Requires some kind of physical authentication (e.g. activation of a pushbutton) on the device to be updated. The device to be firmware updated will not start to request the firmware data.
0xFF	Valid combination of Manufacturer ID and Firmware ID. The device to be firmware updated will start to request the firmware data from the node ID specified in the Firmware Update Meta Data Request Get Command.

3.22.5 Firmware Update Meta Data Get Command

The Firmware Update Meta Data Get Command is used to request the Firmware Update Meta Data Report Command in case a valid firmware is available. The Firmware Update Meta Data Get Command is used as handshake to avoid buffer overflow in the receiving device e.g. when storing the firmware data in the external EEPROM. The Firmware Update Meta Data Get Command will optionally be able to request multiple Firmware Update Meta Data Report Commands to improve throughput. Any missing reports can also be requested individually.

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

Number of Reports (8 bit)

Number of Firmware Update Meta Data Report Commands to be received without requesting each Firmware Update Meta Data Report Command by the Firmware Update Meta Data Get Command.

Report number 1 .. 2 (15 bit)

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.

3.22.6 Firmware Update Meta Data Report Command

The Firmware Update Meta Data Report Command is used to retrieve firmware data from a device. The Firmware Update Meta Data Report Command(s) can only be sent requested by the Firmware Update Meta Data Get Command.

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 field indicates if the requested Firmware Update Meta Data Report Command is the last one or the transfer is not complete. The field is equal to 0x01 when the last report is transmitted otherwise the field has the value 0x00. This field makes it possible to start requesting Firmware Update Meta Data Report Commands without knowing the total number of Firmware Update Meta Data Report Commands to be transferred.

Report number 1 .. 2 (15 bit)

Firmware Update Meta Data Report Command number received. This allows the receiving device to check if all requested reports are received. The report number values must be a sequence starting from 1. The first byte (Report number 1) is the most significant byte.

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

Offset = (report number – 1) x 44

The reports are typically transferred as a sequence starting from 1. The device must respond to any valid request for any report number to deal with reports that did not arrive properly.

Data 1 .. Data N (variable)

The Data fields contain the requested binary firmware data starting from address 0x0000. Each frame must contain 44 bytes to be able to calculate the offset. This result in a frame size equal to 48 bytes due to the payload limitations with respect to a routed single cast frame using 4 hops [2]. The last frame can be reduced in size according to the remaining binary firmware data to be transferred. The number of data fields transmitted in the last frame can be determined from the length field in the frame.

3.22.7 Firmware Update Meta Data Status Report Command

The Firmware Update Meta Data Status Report Command is used to return the firmware update status. The Firmware Update Meta Data Status Report Command is returned when the firmware update is completed or abandoned by the device receiving the firmware. It is not allowed to send a Firmware Update Meta Data Get Command after the 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 bit)

The Status identifier can return the following values:

Status	Description
0x00	Module was unable to receive the requested firmware data without checksum error. Number of retries and request sequence of missing frames are implementation specific.
0x01	Module was unable to receive the requested firmware data. Number of retries and request sequence of missing frames are implementation specific.
0xFF	Firmware update written successfully to EEPROM. Module will now power cycle itself to install the new firmware.

3.22.8 Detailed description of frame flow

A device (Node A) wants to check the current firmware version loaded in another device (Node B). To obtain the firmware version Node A must request it as shown on the figure below:

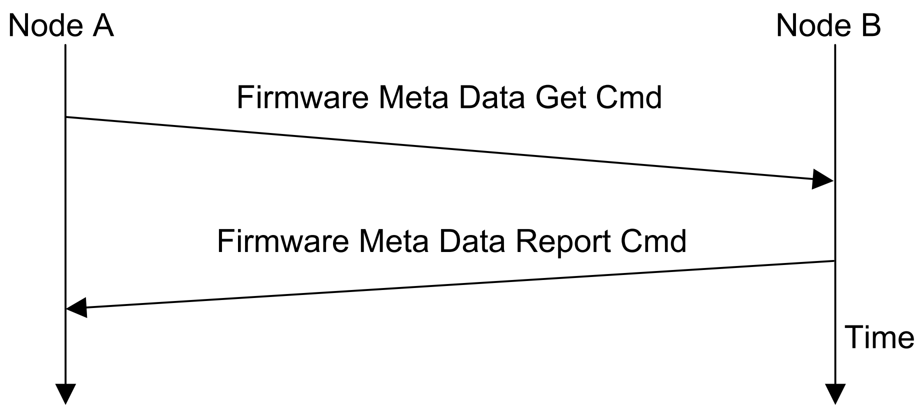


Figure 5, Requesting firmware version in a device

To make a firmware update of Node B must Node A request Node B to start the process. To avoid unintentional firmware updates is it recommended that some kind of physical authentication on the device to be updated is exercised. Node B returns a report to notify Node A whether or not the firmware update request is accepted. In case the firmware update request is accepted by Node B it will start to retrieve the firmware data. The complete frame flow is shown below:

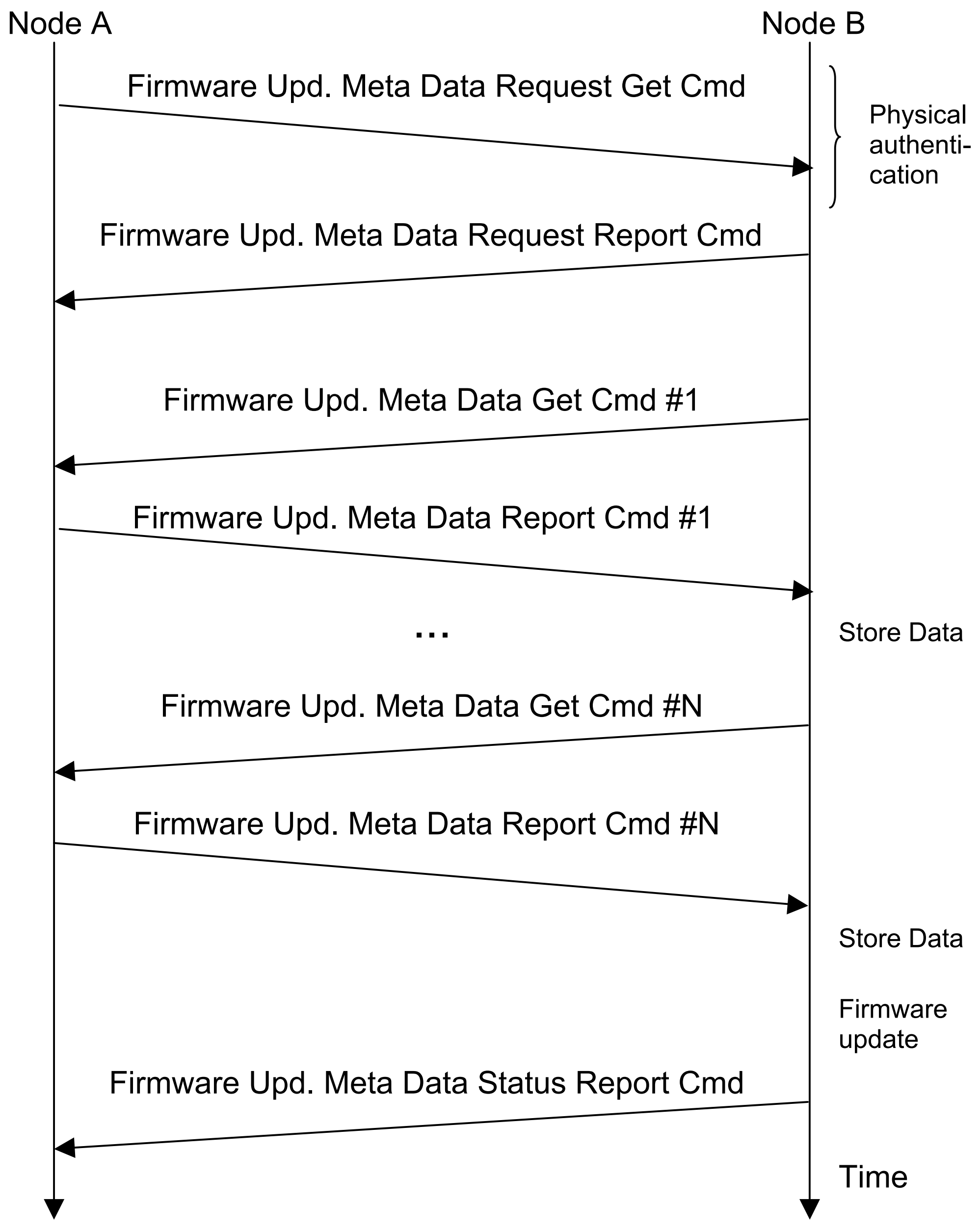

Node A
Node B
Firmware Upd. Meta Data Request Get Cmd
Physical authenti-cation
Firmware Upd. Meta Data Request Report Cmd
Firmware Upd. Meta Data Get Cmd #1
Firmware Upd. Meta Data Report Cmd #1
...
Store Data
Firmware Upd. Meta Data Get Cmd #N
Firmware Upd. Meta Data Report Cmd #N
Store Data
Firmware update
Firmware Upd. Meta Data Status Report Cmd
Time

Figure 6, Firmware update of a device

3.23 Firmware Update Meta Data Command Class, version 2

The Firmware Update Meta Data Report is enhanced in Firmware Update Meta Data Command Class, version 2. The commands not mentioned here will remain the same as in version 1.

3.23.1 Firmware Update Meta Data Report Command

The Firmware Update Meta Data Report command is used to retrieve firmware data from a device. The Firmware Update Meta Data Report command(s) can only be sent as a result of receiving a Firmware Update Meta Data Get command.

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							

Last (1 bit)

The Last field indicates if the requested Firmware Update Meta Data Report command is the last one or the transfer is not complete. The field is equal to 0x01 when the last report is transmitted otherwise the field has the value 0x00. This field makes it possible to start requesting Firmware Update Meta Data Report commands without knowing the total number of Firmware Update Meta Data Report commands to be transferred.

Report number 1 .. 2 (15 bit)

Firmware Update Meta Data Report command number received. This allows the receiving device to check if all requested reports are received. The report number values must be a sequence starting from 1. The first byte (Report number 1) is the most significant byte.

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

Offset = (report number – 1) x 42

The reports are typically transferred as a sequence starting from 1. The device must respond to any valid request for any report number to deal with reports that did not arrive properly.

Data 1 .. Data N (variable)

The Data fields contain the requested binary firmware data starting from address 0x0000. Each frame must contain 42 bytes to be able to calculate the offset. This result in a frame size equal to 48 bytes due to the payload limitations with respect to a routed single cast frame using 4 hops [1]. The last frame can be reduced in size according to the remaining binary firmware data to be transferred. The number of data fields transmitted in the last frame can be determined from the length field in the frame.

Checksum (16 bit)

The checksum field is used to ensure consistency of the command class identifier, command identifier, report number and firmware data downloaded. The checksum is derived from the bytes starting with the command class identifier and until the last data field (Data N). The first byte is the most significant byte. The checksum algorithm is implementation specific but is the same as use in the Firmware Meta Data Report.

3.24 Geographic Location Command Class, version 1

This Geographic Location Command Class is used to read latitude and longitude from a device in a Z-Wave network. The latitude and longitude can also be set according to the geographic location in question. Date and geographic location can be used to calculate sunrise and sunset for e.g. automatic lighting control.

3.24.1 Geographic Location Set Command

The Geographic Location Get Command is used to set latitude and longitude.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_GEOGRAPHIC_LOCATION							
Command = GEOGRAPHIC_LOCATION_SET							
Longitude Degrees							
Long. Sign	Longitude Minutes						
Latitude Degrees							
Lat. Sign	Latitude Minutes						

Longitude (16 bits)

The longitude determines one's location on the earth's surface, East or West of the Greenwich Meridian. The Greenwich Meridian is located at the Greenwich observatory, in Greenwich, England to be the geographic point for where East and West meet. Therefore, Greenwich Meridian is indicated as 0° longitude. Longitude values for points East of the Meridian are always positive, while points West of the Meridian are always negative. Valid ranges are for degrees (from -180 to 180) and minutes (0-59). Other values will be interpreted as 0.

Latitude (16 bits)

The latitude determines one's location on the earth's surface, North or South of the Equator. Latitude is measured between -90° South, and +90° North of the Equator point (0°). Valid ranges are for degrees (from -90 to 90) and minutes (0-59). Other values will be interpreted as 0.

3.24.2 Geographic Location Get Command

The Geographic Location Get Command is used to request latitude and longitude from a device in a Z-Wave network.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_GEOGRAPHIC_LOCATION							
Command = GEOGRAPHIC_LOCATION_GET							

3.24.3 Geographic Location Report Command

The Geographic Location Report Command returns latitude and longitude from a device in a Z-Wave network. The Geographic Location Report Command can be send requested by the Geographic Location Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_GEOGRAPHIC_LOCATION							
Command = GEOGRAPHIC_LOCATION_REPORT							
Long. Sign	Longitude Degrees						
Longitude Minutes							
Lat. Sign	Latitude Degrees						
Latitude Minutes							

Refer to description under the Geographic Location Set Command.

3.25 Grouping Name Command Class, version 1

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

3.25.1 Grouping Name Set Command

The Grouping Name Set Command is 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 2							
…							
Grouping Name x							

Grouping Identifier (8 bit)

The field specifies the Grouping ID.

Reserved (5 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Char. Presentation (3 bit)

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

Char. Presentation	Description
0	Using standard ASCII codes, see **Appendix B** (values 128-255 are ignored)
1	Using standard and OEM Extended ASCII codes, see **Appendix B**
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.

This list may evolve in the future. Undefined values of the character presentation identifier must be ignored.

Grouping Name 1-x (variable)

Grouping name using specified character representation.

The Grouping name can have a maximum of 16 characters and a minimum of 0 characters. The number of character fields transmitted can be determined from the frame length. If a frame with more than 16 characters is received only the first 16 characters must be accept. The remaining characters must be ignored.

3.25.2 Grouping Name Get Command

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

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

Grouping Identifier (8 bit)

The field specifies the grouping identifier.

3.25.3 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 2							
…							
Grouping Name x							

Grouping Identifier (8 bit)

The field specifies the grouping identifier.

Reserved (5 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Char. Presentation (3 bit)

Refer to description under the Grouping Name Set Command

Grouping Name 1-x (variable)

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.

3.26 Hail Command Class, version 1

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

3.26.1 Hail Command

The Hail Command is used to sent unsolicited hail's to other devices in a Z-Wave network.

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

3.27 HRV Status Command Class, version 1

The residential Heat Recovery Ventilation (HRV) Status Command Class is used to read out a number of parameters in the ventilation system.

3.27.1 HRV Status Get Command

The residential HRV Status Get Command is used to request specific parameters from the ventilation system.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_STATUS							
Command = HRV_STATUS_GET							
Status Parameter							

Status Parameter (8 bit)

The status parameter is used to indicate which status parameter that is requested.

Status Parameter	Value
Outdoor Air temperature	0
Supply Air (to room) temperature	1
Exhaust Air (from room) temperature	2
Discharge Air temperature	3
Room temperature	4
Relative Humidity in room	5
Remaining filter life	6

3.27.2 HRV Status Report Command

The residential HRV Status Report Command is used to report a specific status parameter in response to a HRV Status Get.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_STATUS							
Command = HRV_STATUS_REPORT							
Status Parameter							
Precision			Scale		Size		
Value 1							
….							
Value n							

Status Parameter (8 bit)

The status parameter is used to indicate which status parameter that is reported. Refer to 3.27.1 HRV Status Get for possible values.

Precision (3 bit)

The precision field describes what the precision of the setpoint value is. The number indicates the number of decimals. The decimal value 1025 with precision 2 is therefore equal to 10.25.

Scale (2 bit)

The scale field indicates the scale used the list of possible scales are given below:

Status Parameter	Scale	Value
Outdoor Air temperature	Celsius (C)	0
	Fahrenheit (F)	1
Supply Air (to room) temperature	Celsius (C)	0
	Fahrenheit (F)	1
Exhaust Air (from room) temperature	Celsius (C)	0
	Fahrenheit (F)	1
Discharge Air temperature	Celsius (C)	0
	Fahrenheit (F)	1
Room temperature	Celsius (C)	0
	Fahrenheit (F)	1
Relative Humidity in room	Percentage (%)	0
Remaining filter life	Percentage (%)	0

Size (3 bit)

The size field indicates the number of bytes that is used for the sensor value. This field can take values from 1 (001b), 2 (010b) or 4 (100b).

Value 1 … Value n

The value is a signed field. The value can be 1, 2 or 4 bytes in size. This first byte is the most significant byte. The table below show examples of signed decimal value with their hexadecimal equivalents.

Signed 1 byte decimal value	Hexadecimal
127	0x7F
25	0x19
2	0x02
1	0x01
0	0x00
-1	0xFF
-2	0xFE
-25	0xE7
-128	0x80

Signed 2 bytes decimal value	Hexadecimal
32767	0x7FFF
1025	0x0401
2	0x0002
1	0x0001
0	0x0000
-1	0xFFFF
-2	0xFFFE
-1025	0xFBFF
-32768	0x8000

3.27.3 HRV Status Supported Get Command

The HRV Status Supported Get Command is used to request a bitmap of the supported status parameters.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_STATUS							
Command = HRV_STATUS_SUPPORTED_GET							

3.27.4 HRV Status Supported Report Command

The HRV Status Supported Report Command is used to report a bitmap indicating the supported status parameters.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_STATUS							
Command = HRV_STATUS_SUPPORTED_REPORT							
Bit Mask 1							
…							
Bit Mask N							

Bit Mask 1 ... Bit Mask N (N * Byte)

The Bit Mask fields describe the supported status parameters from the ventilation system. Bit 0 in the Bit Mask 1 field is used to indicate if the status parameter “Outdoor Air Temperature” is supported, 0 indicating not supported and 1 indicating supported. Bit 1 in the Bit Mask 1 field is used to indicate if the status parameter “Supply Air Temperature” is supported and so forth. All available status parameters are given in Section 3.27.1 HRV Status Get.

It is only necessary to send the Bit Mask fields 1 and up to the one indicating the last support status parameter. The number of Bit Mask fields transmitted can be determined from the length field in the frame.

3.28 HRV Control Command Class, version 1

The Heat Recovery Ventilation (HRV) Control Command Class is introducing control of Heat Recovery Ventilation systems via the Z-Wave interface.

3.28.1 HRV Mode Set

The HRV Mode Set Command is used to set the desired mode in the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_CONTROL							
Command = HRV_CONTROL_MODE_SET							
Reserved			Mode				

Mode (5 bit)

The mode identifier can be set to the following values:

Mode	Name	Description
0	Off	The HRV system is in the off state, frost protection can occur.
1	Demand / Automatic	The HRV system is controlled based on sensor input.
2	Schedule	The HRV system is controlled based on predefined input from the factory and/or user/installer.
3	Energy Savings Mode	The HRV system will be put into a reduced heat / ventilation mode.
4	Manual	The HRV system is in manual mode. The command HRV Manual Control Set can be used to manually control the device.

This list may evolve in the future. If a mode is not supported then it must be ignored.

Reserved (3 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

3.28.2 HRV Mode Get Command

The HRV Mode Get Command is used to request the current mode from the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_CONTROL							
Command = HRV_CONTROL_MODE_GET							

3.28.3 HRV Mode Report Command

The HRV Mode Report Command is used to report the mode from the device. It can be sent either unsolicited or requested by the Mode Get command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_CONTROL							
Command = HRV_CONTROL_MODE_REPORT							
Reserved			Mode				

Mode (8 bit)

Refer to description under the HRV Mode Set command.

Reserved (3 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

3.28.4 HRV Bypass Set Command

The HRV Bypass Set Command is used to set the bypass mode when the ventilation system is set to manual mode. If the system is not in manual mode while receiving this command it must be ignored.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_CONTROL							
Command = HRV_CONTROL_BYPASS_SET							
Bypass							

Bypass (8 bit)

The value can be either 0x00 (close) or 0xFF (open).

Furthermore it can take a percentage value between 1 to 99 (0x01 - 0x63) if the ventilation system supports modulated bypass, the percentage value will represent the aperture of the bypass. If the system does not support the full range of aperture steps, the supported values must be mapped linearly over the entire scale.

If ventilation system does support modulated bypass the values from 1 to 99 must be interpreted as fully open.

The value 254 (0xFE) is used to set the bypass into automatic mode.

3.28.5 HRV Bypass Get Command

The HRV Bypass Get Command is used to request the current bypass setting.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_CONTROL							
Command = HRV_CONTROL_BYPASS _GET							

3.28.6 HRV Bypass Report Command

The HRV Bypass Report command is used to report the current bypass parameters.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_CONTROL							
Command = HRV_CONTROL_BYPASS_REPORT							
Bypass							

Bypass (8 bit)

See description under Section 3.28.4 HRV Bypass Set.

3.28.7 HRV Ventilation Rate Set Command

The HRV Ventilation Rate Set Command is used to set the ventilation rate when the ventilation system is set to manual mode. If the system is not in manual mode while receiving this command it must be ignored.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_CONTROL							
Command = HRV_CONTROL_VENTILATION_RATE_SET							
Ventilation Rate							

Ventilation Rate (8 bit)

The value can be either 0x00 (off) or 0xFF (on). Furthermore it can take a percentage value between 1 to 99 (0x01 - 0x63). A ventilation system is not required to support all possible steps between 1 and 99.

3.28.8 HRV Ventilation Rate Get Command

The HRV Ventilation Rate Get Command is used to request the current ventilation rate setting.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_CONTROL							
Command = HRV_CONTROL_ VENTILATION_RATE _GET							

3.28.9 HRV Ventilation Rate Report Command

The HRV Ventilation Rate Report Command is used to report the current ventilation rate setting.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_CONTROL							
Command = HRV_CONTROL_ VENTILATION_RATE _REPORT							
Ventilation Rate							

Ventilation Rate (8 bit)

See description under Section 3.28.7 HRV Ventilation Rate Set.

3.28.10 HRV Mode Supported Get Command

The HRV Mode Supported Get Command is used to request the supported modes from the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_CONTROL							
Command = HRV_CONTROL_MODE_SUPPORTED_GET							

3.28.11 HRV Mode Supported Report Command

The HRV Mode Supported Report Command is used to report the supported modes from the device. It can be sent either unsolicited or requested by the Mode Supported Get command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_HRV_CONTROL							
Command = HRV_CONTROL_MODE_SUPPORTED_REPORT							
Reserved				Manual Control Supported			
Bit Mask 1							
…							
Bit Mask N							

Reserved (5 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Manual Control Supported (3 bits)

The manual control supported bits describes the supported manual control modes of the ventilation system. The mode is supported if the bit is 1; if the bit is 0 then the mode is not supported.

The bits are mapped to the following controls:

Bit	Name
0	Bypass Open / Close
1	Bypass Auto
2	Modulated Bypass
3	Ventilation Rate

E.g. a ventilation system supporting only open and close would report Manual Control Supported = 0x01.

Bit Mask 1 ... Bit Mask N (N * Bytes)

The Bit Mask fields describe the supported modes by the device. The bit 0 in Bit Mask 1 field is used to indicate whether Mode = 0 (Off) is supported or not. The mode is supported if the bit is 1; if the bit is 0 then the mode is not supported. The bit 1 in Bit Mask 1 field is used by Mode = 1 (Demand / Automatic) and so forth. It is only necessary to send the Bit Mask fields from 1 and up to the one indicating the last supported mode. The number of Bit Mask fields transmitted can be determined from the length field in the frame.

All available modes are given in Section 3.28.1 HRV Mode Set.

3.29 Indicator Command Class, version 1

The Indicator Command Class is used to show the actual state, level etc. on a device.

3.29.1 Indicator Set Command

The Indicator Set Command can be used to set a device on or off (enable or disable).

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

Value (8 bit)

The value can be either 0x00 (off/disable) or 0xFF (on/enable). Furthermore it can take values from 1 to 99 (0x01 – 0x63). In case the indicator does not have the capability to show the range from 1 to 99 it is interpreted as 0xFF (on/enable).

3.29.2 Indicator Get Command

The Indicator Get Command can be used to get the state of the device.

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

3.29.3 Indicator Report Command

The Indicator Report Command can be sent unsolicited or requested by the Indicator Get Command.

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

Value (8 bit)

Refer to description under the Indicator Set Command.

3.30 IP Configuration Command Class, version 1

The IP Configuration Command Class used to configure network identifiers for IPV4 devices. The intended use of the command class is illustrated in the figure below.

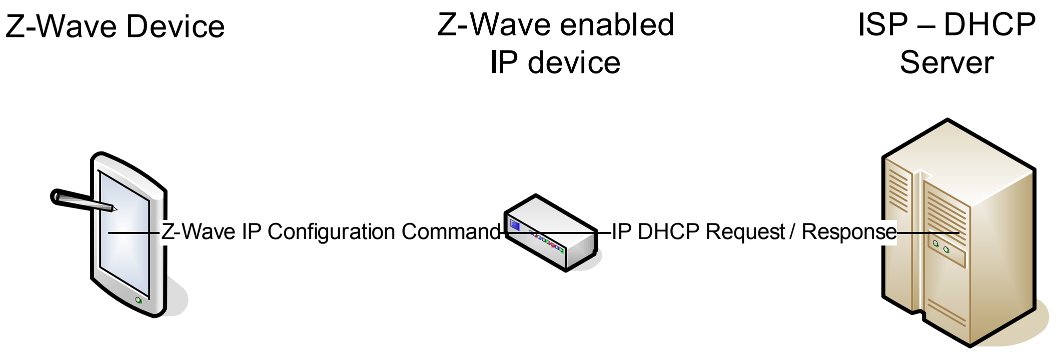


Figure 7, Configuration of network identifiers for IPV4 devices

In the figure the Z-Wave Remote to the left, sends an IP Configuration Command to the Z-Wave enabled IP device, telling it to acquire its configuration using DHCP. The Z-Wave enabled IP device will now perform a standard DHCP IP request to the DHCP server over an IP based network.

Another example might be where the Z-Wave Remote statically configures the Z-Wave enabled IP device with fixed IP, subnet, DNS etc. by sending an IP Configuration Command.

Note that this class is only intended for IPV4 and not IPV6 support.

3.30.1 IP Configuration Set Command

The IP Configuration Set Command is used to configure IPV4 settings in a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_IP_CONFIGURATION							
Command = IP_CONFIGURATION_SET							
Reserved						Auto IP	Auto DNS
IP Address 1							
IP Address 2							
IP Address 3							
IP Address 4							
Subnet Mask 1							
Subnet Mask 2							
Subnet Mask 3							
Subnet Mask 4							
Gateway 1							
Gateway 2							
Gateway 3							
Gateway 4							
DNS1 1							
DNS1 2							
DNS1 3							
DNS1 4							
DNS2 1							
DNS2 2							
DNS2 3							
DNS2 4							

Reserved (6 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Auto IP (1 bit)

If Auto IP bit is set, the following fields are ignored: IP Address, Subnet Mask, and Gateway. And are allocated by DHCP or BOOTP instead.

Auto DNS (1 bit)

The Auto DNS if set indicates to ignore DNS1 and DNS2 and allocate DNS by DHCP instead. Note that some devices might not support Auto DNS without Auto IP set.

IP Address 1…4 (32 bit)

The IP Address indicates the static IP address of the device itself. The first byte is the most significant byte.

Subnet mask 1…4 (32 bit)

The Subnet Mask determines the portion of the IP address that represents the subnet. The first byte is the most significant byte.

Gateway 1…4 (32 bit)

The Gateway indicates the default gateway that serves as an access point to another network. The first byte is the most significant byte.

DNS1 1…4 (32 bit)

The DNS1 allows the use of domain name system (DNS) server names instead of using numerical IP addresses for management packet routing. In case the device will not need DNS, and should not query it from DHCP then leave field as all zeroes. The first byte is the most significant byte.

DNS2 1…4 (32 bit)

The DNS2 provides a secondary DNS server name. In case only one DNS server is available or the device will not need DNS then leave field as all zeroes. The first byte is the most significant byte.

3.30.2 IP Configuration Get Command

The IP Configuration Get Command is used to request the IPV4 settings in a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_IP_CONFIGURATION							
Command = IP_CONFIGURATION_GET							

3.30.3 IP Configuration Report Command

The IP Configuration Report Command is used to return IPV4 settings in a device. The IP Configuration Report Command can only be sent requested by the IP Configuration Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_IP_CONFIGURATION							
Command = IP_CONFIGURATION_REPORT							
Reserved						Auto IP	Auto DNS
IP Address1							
IP Address2							
IP Address3							
IP Address4							
Subnet Mask1							
Subnet Mask2							
Subnet Mask3							
Subnet Mask4							
Gateway1							
Gateway2							
Gateway3							
Gateway4							
DNS11							
DNS12							
DNS13							
DNS14							
DNS21							
DNS22							
DNS23							
DNS24							
LeaseTime1							
LeaseTime2							
LeaseTime3							
LeaseTime4							

Refer to explanation of parameters in IP Configuration Set Command description.

Lease Time 1…4 (32 bit)

The lease time specifies the time the IP address has been granted, if Auto IP is being used (in seconds). This field is optional; if the device does not know its lease period it should return 0 for the lease time fields.

3.30.4 IP Configuration DHCP Release Command

The IP Configuration DHCP Release Command is used to release the DHCP lease.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_IP_CONFIGURATION							
Command = IP_CONFIGURATION_RELEASE							

3.30.5 IP Configuration DHCP Renew Command

The IP Configuration DHCP Renew Command is used to force the renewal of the DHCP lease.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_IP_CONFIGURATION							
Command = IP_CONFIGURATION_RENEW							

3.31 Language Command Class, version 1

The Language Command Class is used to specify the language settings on a device.

3.31.1 Language Set Command

The Language Set Command is used to transfer data to a device. The device uses the default language in case the selected language is not supported.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_LANGUAGE							
Command = LANGUAGE_SET							
Language 1							
Language 2							
Language 3							
Country 1							
Country 2							

Language 1 ... 3

The code definition of the languages can be found in ISO 639-2:1998 ‘Codes for the representation of names of languages – Part 2: Alpha-3 code’. In the table below are some examples of the alpha-3 codes listed:

Language	Language 1	Language 2	Language 3
English	e	n	g
Flemish; Dutch	d	u	t
French	f	r	e
German	g	e	r
Italian	i	t	a
Polish	p	o	l
Russian	r	u	s
Walloon	w	l	n

Country 1 ... 2

The code definition of the countries can be found in ISO 3166-1 ‘Country Codes: Alpha-2 codes’. The country is optional and only defined in case it’s necessary to differ geographical. The number of data fields transmitted can be determined from the length field in the frame. In the table below are some examples of the alpha-2 codes listed:

Language	Country 1	Country 2
Belgium	B	E
Italy	I	T
Netherlands	N	L
Poland	P	L
United Kingdom	G	B
United States	U	S

3.31.2 Language Get Command

The Language Get Command is used to request the current language setting in a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_LANGUAGE							
Command = LANGUAGE_GET							

3.31.3 Language Report Command

The Language Report returns the current language setting in a device. The Language Report is obtained by the Language Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_LANGUAGE							
Command = LANGUAGE_REPORT							
Language 1							
Language 2							
Language 3							
Country 1							
Country 2							

Refer to explanation under the Language Set Command.

3.32 Lock Command Class, version 1

The Lock Command Class is used to lock and unlock a “lock” type device, e.g. a door or window lock

3.32.1 Lock Set Command

The Lock Set Command is used to set the lock state in a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_LOCK							
Command = LOCK_SET							
Lock State							

Lock State (8 bit)

The lock state field is used to set the lock state of the device. The value 0 indicates that the device is unlocked. The value 1 indicates that the device is locked.

3.32.2 Lock Get Command

The Lock Get Command is used to request the lock state from a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_LOCK							
Command = LOCK_GET							

3.32.3 Lock Report Command

The Lock Report Command is used to report the lock state of a device. The Lock Report Command can be sent unsolicited or requested by the Lock Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_LOCK							
Command = LOCK_REPORT							
Lock State							

Lock State (8 bit)

The Lock state field is used to report the lock state of the device. The value 0 indicates that the device is unlocked. The value 1 indicates that the device is locked.

3.33 Manufacturer Proprietary Command Class, version 1

The Manufacturer Proprietary Command Class is used to transfer data between devices in the Z-Wave network. The data content must be vendor specific and must be non-value added with respect to the Home Automation application in general. An example could be data used to diagnose the hardware in a device.

Note: The Manufacturer Proprietary Command Class must not be used without written approval from Zensys.

3.33.1 Manufacturer Proprietary Command

The Manufacturer Proprietary Command is used to transfer data between devices in the Z-Wave network. The Command contains a manufacturer specific identifier to allow the receiving device to check if this Command can be interpreted. The vendor is responsible for establishing a Command structure to differ between the set of Commands supported.

Meta Data: In case more than one frame must be transferred sequentially then the API call ZW_SendDataMeta can be used to ensure that control data gets through in the Z-Wave network.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MANUFACTURER_PROPRIETARY							
Manufacturer ID 1							
Manufacturer ID 2							
Data 1							
…							
Data N							

Manufacturer ID (16 bit)

The Manufacturer ID is a unique ID identifying the manufacturer of the device. The Manufacturer ID is assigned by Zensys upon request. See Appendix A for a list of assigned ID's. The first byte (Manufacturer ID 1) is the most significant byte.

Data 1 .. Data N (variable)

The data fields can be used for data transfer etc. The number of data fields transmitted can be determined from the length field in the frame.

3.34 Manufacturer Specific Command Class, version 1

Class to advertise manufacturer specific information use the Manufacturer Specific Command Class. The Manufacturer ID is assigned by Zensys upon request.

3.34.1 Manufacturer Specific Info Get Command

A device will use the Manufacturer Specific Info Get Command to get the manufacturer information from another device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MANUFACTURER_SPECIFIC							
Command = MANUFACTURER_SPECIFIC_GET							

3.34.2 Manufacturer Specific Info Report Command

A device require the Manufacturer Specific Info Report to obtain the manufacturer specific information in another device. The Manufacturer Specific Info Report Command can be sent unsolicited or requested by the Manufacturer Specific Info Get Command.

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 bit)

The Manufacturer ID is a unique ID identifying the manufacturer of the device. The Manufacturer ID is assigned by Zensys upon request. See Appendix A for a list of assigned ID’s. The first byte is the most significant byte.

Product Type ID (16 bit)

The manufacturer assigns the Product Type ID. The first byte is the most significant byte.

Product ID (16 bit)

The manufacturer assigns the Product ID. The first byte is the most significant byte.

3.35 Meter Command Class, version 1

The Meter Command Class defines the Commands necessary to read accumulated values in physical units from a water meter or metering device (gas, electric etc.) and thereby enabling automatic meter reading capabilities.

Automatic meter reading (AMR), is the technology of automatically collecting data from water meter or energy metering devices and transferring that data to a central database for billing and/or analyzing.

3.35.1 Meter Get Command

The Meter Get Command is used to request the accumulated consumption in physical units from a metering device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_METER							
Command = METER_GET							

3.35.2 Meter Report Command

The Meter Report Command can be used by a metering device to send a report containing the accumulated consumption in physical units either unsolicited or requested by the Meter Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_METER							
Command = METER_REPORT							
Meter Type							
Precision			Scale		Size		
Meter Value 1							
Meter Value 2							
..							
Meter Value n							

Meter Type (8 bit)

Meter type specifies what type of metering device this Command originates from. Refer to the table below with respect to defined metering devices. New metering device types/values can be requested from Zensys.

Sensor Type	Value
Electric meter	0x01
Gas meter	0x02
Water meter	0x03
	0x04
	0x05
	0x06

Precision (3 bit)

The precision field describes what the precision of the value is. The number indicates the number of decimals. The decimal value 1025 with precision 2 is therefore equal to 10.25.

Scale (2 bit)

The Scale is used to indicate what unit the metering device uses. Refer to the table below with respect to defined scales for the relevant metering devices. New scales/values can be requested from Zensys.

Meter Type	Scale	Value
Electric meter	kWh	0x00
		0x01
		0x02
		0x03
Gas meter	Cubic meters	0x00
		0x01
		0x02
		0x03
Water meter	Cubic meters	0x00
	Cubic feet	0x01
	US gallons	0x02
		0x03

Size (3 bit)

The size field indicates the number of bytes that is used for the meter value. This field can take values from 1 (001b), 2 (010b) or 4 (100b).

Meter Value (variable)

The meter value is a signed field. The field can be 1, 2 or 4 bytes in size. The first byte is the most significant byte. The table below shows signed decimal values together with their hexadecimal equivalents.

Signed 1 byte decimal value	Hexadecimal
127	0x7F
25	0x19
2	0x02
1	0x01
0	0x00
-1	0xFF
-2	0xFE
-25	0xE7
-128	0x80

Signed 2 bytes decimal value	Hexadecimal
32767	0x7FFF
1025	0x0401
2	0x0002
1	0x0001
0	0x0000
-1	0xFFFF
-2	0xFFFE
-1025	0xFBFF
-32768	0x8000

Notice: The metering device receiving the Meter Report must always show the value even though the Metering Device Type and/or Scale are not supported.

3.36 Meter Command Class, version 2

The Meter Command Class is intended for Z-Wave enabled devices capable of reporting energy measurements in addition to any main functionality or features e.g. an appliance module reporting the current consumption of the connected load. This command class is not intended for residential utility submetering such as a water meter counting total consumption. Utility meters must refer to the Advanced Energy Control framework.

Meter Command Class (Version 2) is improved with the following functionalities:

- Commands to interview the device for supported Meter types
- 'Previously accumulated consumption' to allow for easy calculation of consumption since last measurement.
- Reset of accumulated consumption
- Add capability to communicate current consumption (W)
- 'Scale' entry for pulse meters along with W, kVAh and Cubic ft.

The commands not mentioned here will remain the same as specified for Meter Command Class (Version 1).

3.36.1 Meter Supported Get Command

The Meter Supported Get Command is used to request the supported scales in a sub meter.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_METER							
Command = METER_SUPPORTED_GET							

3.36.2 Meter Supported Report Command

The Meter Supported Report Command is used to report the supported scales in a sub meter as a reply to the Meter Supported Get Command. The Meter Supported Report Command must not be sent unsolicited.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_METER							
Command = METER_SUPPORTED_REPORT							
Meter Reset	Reserved		Meter Type				
Reserved				Scale Supported			

Meter Reset (1 bit)

The Meter Reset field set to “1” indicates support for the Meter Reset Command.

Meter Type (5 bit)

See Meter Type defined in Meter Report Command.

Scale Supported (4 bit)

Meter Type	Scale	Value	Accumulate / Instant measurement
Electric meter	kWh	Bit 0 asserted	Accumulated
	kVAh	Bit 1 asserted	Accumulated
	W	Bit 2 asserted	Instant
	Pulse count	Bit 3 asserted	Accumulated
Gas meter	Cubic meters	Bit 0 asserted	Accumulated
	Cubic feet	Bit 1 asserted	Accumulated
	Reserved	Bit 2 asserted	n/a
	Pulse count	Bit 3 asserted	Accumulated
Water meter	Cubic meters	Bit 0 asserted	Accumulated
	Cubic feet	Bit 1 asserted	Accumulated
	US gallons	Bit 2 asserted	Accumulated
	Pulse count	Bit 3 asserted	Accumulated

3.36.3 Meter Reset Command

The Meter Reset Command is used to reset ALL <u>accumulated values</u> stored in the meter device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_METER							
Command = METER_RESET							

3.36.4 Meter Get Command

The Meter Get Command is used to request a device supported measurement in physical units from a metering device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_METER							
Command = METER_GET							
Reserved			Scale		Reserved		

Scale (2 bit)

See description of Meter Report Command.

Note! A device receiving a Meter Get Command containing a non-supported Scale must ignore the command.

3.36.4.1 Backwards compatibility

When devices supporting Meter Command Class (Version 1) receives a Meter Get Command of Version 2 it must report its implemented scale.

When devices supporting Meter Command Class (Version 2) receives a Meter Get Command of Version 1 it must report its default scale. It is up to the manufacture to define which scale is the default scale and describe this in the product manual.

3.36.5 Meter Report Command

The Meter Report Command must be transmitted as a response to the Meter Get Command and in addition it can also be transmitted unsolicited in case the OEM application deems this necessary. The OEM application must however ensure not to transmit unsolicited report commands using a broadcast frame, in order to avoid unintentional transmission of an unsolicited command to a destination not ratifying support for it.

The Meter Report Command is used by the metering device to report the instant or accumulated consumption in physical units.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_METER							
Command = METER_REPORT							
Reser-ved	Rate Type		Meter Type				
Precision			Scale		Size		
Meter Value 1							
..							
Meter Value n							
Delta Time 1							
Delta Time 2							
Previous Meter Value 1							
..							
Previous Meter Value n							

Rate Type (2 bit)

Rate Type specifies if it is import or export values to be read. Setting the Rate Type to Import on the Meter Report Command is an indication that the Meter Value is a consumed measurement. In contrary when the Rate Type is set to Export the indication of the Meter Value is a produced measurement.

Rate Type	Value
Reserved	0x00
Import (consumed)	0x01
Export (produced)	0x02
Reserved	0x03

Meter Type (5 bit)

Meter Type specifies what type of metering device this command originates from. Refer to the table below with respect to defined metering devices. New metering device types/values can be requested from Zensys.

Meter Type	Value
Reserved	0x00
Electric meter	0x01
Gas meter	0x02
Water meter	0x03
Reserved	0x04-0x1F

Precision (3 bit)

The Precision field describes what the precision of the value is. The number indicates the number of decimals. The decimal value 1025 with precision 1 must be interpreted as 102.5 and if precision is 3 then the interpreted value is 1.025.

Scale (2 bit)

The Scale field is used to indicate what unit the metering device uses. Refer to the table below with respect to defined scales for the relevant metering devices. New scales/values can be requested from Zensys.

Meter Type	Scale	Value
Electric meter	kWh	0x00
	kVAh	0x01
	W	0x02
	Pulse count	0x03
Gas meter	Cubic meters	0x00
	Cubic feet	0x01
	Reserved	0x02
	Pulse count	0x03
Water meter	Cubic meters	0x00
	Cubic feet	0x01
	US gallons	0x02
	Pulse count	0x03

Size (3 bit)

The Size field indicates the number of bytes that is used for the Meter Value and the Previous Meter Value. This field can take values from 1 (001b), 2 (010b) or 4 (100b) meaning both the Meter Value and the Previous Meter Value can be 1, 2 or 4 bytes long respectively.

Meter Value (variable) / Previous Meter Value (variable)

The Meter Value and Previous Meter Value are signed fields and can be 1, 2 or 4 bytes long defined by the Size field i.e. if the Size field is 2 then both Meter Value and Previous Meter Value are 2 bytes long. The first byte is the most significant byte.

The table below shows signed decimal values together with the hexadecimal equivalents.

Signed 1 byte		Signed 2 bytes		Signed 4 bytes	
Dec	**Hex**	**Dec**	**Hex**	**Dec**	**Hex**
127	0x7F	32767	0x7FFF	2147483647	0x7FFFFFFF
..	..	..	..	..	..
63	0x3F	16383	0x3FFF	1073741823	0x3FFFFFFF
..	..	..	..	..	..
1	0x01	1	0x0001	1	0x00000001
0	0x00	0	0x0000	0	0x00000000
-1	0xFF	-1	0xFFFF	-1	0xFFFFFFFF
..	..	..	..	..	..
-63	0xC1	-16383	0xC001	-1073741823	0xC0000001
..	..	..	..	..	..
-128	0x80	-32768	0x8000	-2147483648	0x80000000

Delta Time (16 bit)

The Delta Time field will report the elapsed time in seconds between the ‘Meter Value’ and the ‘Previous Meter Value’ measurements. Valid values are 0 to 65536 seconds.

Delta Time	Value
No Previous Meter Value field included in the Meter Report	0x0000
1 sec. between Meter Value and Previous Meter Value	0x0001
..	..
65534 sec. between Meter Value and Previous Meter Value	0xFFFE
Unknown Delta Time between Meter Value and Previous Meter Value.	0xFFFF

3.36.5.1 Examples of Meter Report Commands

To retrieve accumulated power consumption from a device supporting Meter Command Class (Version 2), the Meter Get Command must indicate what scale is requested.

7	6	5	4	3	2	1	0
0x32 (COMMAND_CLASS_METER)							
0x01 (METER_GET)							
0			0 (kWh)		0		

As a response to the above Meter Get Command an example of accumulated power consumption can be reported with the below Meter Report indicating a current measurement of 12065.298 kWh and 10 minutes ago the previous measurement was 12060.678 kWh.

7	6	5	4	3	2	1	0
0x32 (COMMAND_CLASS_METER)							
0x02 (METER_REPORT)							
0	2 (export)		0 (Electric meter)				
3 (3 decimals)			0 (kWh)		4 (4 bytes)		
0x00 (Meter Value 1)							
0xB8 (Meter Value 2)							
0x1A (Meter Value 3)							
0x12 (Meter Value 4)							
0x02 (Delta Time 1)							
0x58 (Delta Time 2)							
0x00 (Previous Meter Value 1)							
0xB8 (Previous Meter Value 2)							
0x08 (Previous Meter Value 3)							
0x06 (Previous Meter Value 4)							

To retrieve instant power consumption from a device supporting Meter Command Class (Version 2), the Meter Get Command must indicate what scale is requested.

7	6	5	4	3	2	1	0
0x32 (COMMAND_CLASS_METER)							
0x01 (METER_GET)							
0			2 (W)		0		

An example of instant power consumption can be reported with the below Meter Report indicating a current measurement of 39.99 W.

7	6	5	4	3	2	1	0
0x32 (COMMAND_CLASS_METER)							
0x02 (METER_REPORT)							
0	2 (export)		0 (Electric meter)				
2 (2 decimals)			2 (W)		2 (2 bytes)		
0x0F (Meter Value 1)							
0x9F (Meter Value 2)							
0x00 (Delta Time 1)							
0x00 (Delta Time 2)							

3.37 Move To Position Window Covering Command Class, version 1

The Move To Position Window Covering Command Class is used to move the window covering to a given position and request current position.

3.37.1 Move To Position Set Command

The Move To Position Set Command is used to move the window covering to a given position. The speed of the movement is implementation specific.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MTP_WINDOW_COVERING							
Command = MOVE_TO_POSITION_SET							
Value							

Value (8 bit)

The value can be either 0x00 (close) or 0xFF (open). Furthermore it can take values from 1 to 99 (0x01 – 0x63). The unit of the value is percentage.

3.37.2 Move To Position Get Command

The Move To Position Get Command is used to request the actual position of a drape, shade, blind etc.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MTP_WINDOW_COVERING							
Command = MOVE_TO_POSITION_GET							

3.37.3 Move To Position Report Command

The Move To Position Report Command is sent requested by the Move To Position Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MTP_WINDOW_COVERING							
Command = MOVE_TO_POSITION_REPORT							
Value							

Value (8 bit)

Refer to explanation under the Move To Position Set Command.

3.38 Multi Command Command Class, version 1

This Multi Command Command Class is used to encapsulate multiple Commands in one Command. The purpose for this command class is to limit the number of transmissions to reduce network traffic and improve latency when a number of Commands must be executed sequentially in a device. This command class can also be useful for e.g. battery operated devices in order to extend battery lifetime.

A device that support the Multi Command Command Class may receive a combination of encapsulated and normal non-encapsulated requests and the correct response should be as follows:

a) If the request is sent encapsulated, the response must be returned encapsulated also.

b) If the request is sent as a normal frame, i.e. non-encapsulated, the response must also be sent non-encapsulated. Only if c) below is observed the response can be send encapsulated.

c) Before any setting, request, or response is send encapsulated to any other device, the sending device must ensure, that the destination will be able to understand the encapsulated Command.

For a device sending set or request Commands, this could for instance be through the device application software requesting the NIF from the destination and ensuring that the Multi Command Command Class is listed as “supported”.

For a device responding, this is automatically ensured in situation a) above. However, in situation b) above, or where a device is configured to send responses (e.g. reports) to a 3rd device, for instance via the Association Command Class, then the responding device must ensure the destination device understands the Multi Command encapsulation as described above.

For clarification it should be emphasized that a part of implementing the Multi Command Command Class as “controller” the device must also be able to decode the encapsulated responses that may come back as a result of sending encapsulated request (e.g. get Command to retrieve a report).

A device that supports the Multi Command Command Class must be able to receive and interpret the encapsulated form of all the command classes that it list in the NIF and supports in the normal non-encapsulated form, except of course the Multi Command Command Class itself.

3.38.1 Multi Command Encapsulated Command

The Multi Command Encapsulated Command is used to contain multiple Commands. The encapsulated Commands must be executed in the order they are received. In case get Commands in a Multi Command Encapsulated Command are received by a device the reports must be replied in a Multi Command Encapsulated Command in the same order as the gets were received. Be aware of the payload limitations with respect to a routed single cast frame [2].

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CMD							
Command = MULTI_CMD_ENCAP							
Number of Commands							
Command Length 1							
Command Class 1							
Command 1							
Data 1,1							
Data 1,2							
…							
Data 1,N							
…							
Command Length X							
Command Class X							
Command X							
Data X,1							
Data X,2							
…							
Data X,N							

Number of Commands (8 bit)

This field indicates the number of encapsulated Commands.

Command Length (8 bit)

The Command Length field indicates the number of bytes used by the following encapsulated Command including Command Class identifiers, Command identifiers and the data.

Command Class (8 bit)

The Command Class field indicates the command class identifier of the encapsulated Command.

Command (8 bit)

The Command field indicates the Command identifier of the encapsulated Command.

Data 1-N (variable)

The Data fields in the encapsulated Command of the respective command class and Command identifiers.

3.38.2 Example

This example shows how a battery operated device can be instructed to go to sleep immediately after an internal parameter have been changed by one frame using the Multi Command Encapsulated Command.

7	6	5	4	3	2	1	0
COMMAND_CLASS_MULTI_CMD							
MULTI_CMD_ENCAP							
Number of Commands = 0x02							
Length = 0x05							
COMMAND_CLASS_CONFIGURATION							
CONFIGURATION_SET							
Parameter Number = 0x01							
Size = 0x01							
Configuration Value = 0x07							
Length = 0x02							
COMMAND_CLASS_WAKE_UP							
WAKE_UP_NO_MORE_INFORMATION							

In this example the Multi Command Encapsulated Command contains two encapsulated Commands. First the internal parameter no. 1 is changed to 0x07 by the Configuration Set Command and afterwards the Wake Up No More Information Command instructs the battery operated device to go to sleep immediately after, thereby minimizing the power consumption.

3.39 Multi Instance Association Command Class, version 1

This Multi Instance Association Command Class is used to enable bindings to nodes comprising of a number of instances. The command class can handle nodes with and without instances. The purpose of this command class is to simplify handling of identical devices in one node using instances and where it is acceptable that a number of instances cannot be activated simultaneously via a Command.

The Association Command Class version 1 is a subset of the Multi Instance Association Command Class and the data intersection of the two command classes must be mirrored on application level allowing access to the same data via both command classes. A device supporting the Multi Instance Association Command Class must therefore support the Association Command Class version 1 to fulfill the above requirement.

NOTE: A device supporting the Multi Instance Association Command Class must support the Association Command Class version 1.

NOTE: The Association Report will only return node IDs for nodes without instances specified. To obtain associations configured for nodes with and without instances request the Multi Instance Association Report Command.

3.39.1 Multi Instance Association Set Command

The Multi Instance Association Set Command is used to add nodes with/without instances to a given grouping identifier. The node receiving the set Command should add the nodes received to the nodes already associated by this grouping until the grouping is full. Remember that routing slaves also must have assigned return routes by a controller using the API call ZW_AssignReturnRoute [2] to all the associated nodes. Delete all return routes by the API call ZW_DeleteReturnRoute before assignment. This is of course necessary for all the associated nodes out of direct range but should always be done default to all the associated nodes to create a reliable and robust network. It's optional whether the return routes are assigned before or after the Multi Instance Association Set Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_INSTANCE_ASSOCIATION							
Command = MULTI_INSTANCE_ASSOCIATION_SET							
Grouping identifier							
Node ID1							
Node ID2							
...							
Node ID n							
Marker = MULTI_INSTANCE_ASSOCIATION_SET_MARKER							
Multi Instance Node ID1							
Instance 1							
Multi Instance Node ID2							
Instance 2							
…							
Multi Instance Node IDn							
Instance n							

Grouping identifier (8 bit)

This grouping identifier is used to instruct how nodes are grouped together. The grouping identifier values must be a sequence starting from 1. This field can be ignored in case the node only supports one grouping.

NodeID1 .. NodeIDn

These fields contain a list of node IDs that should be associated with the grouping.

Marker (8 bit)

This marker identifier is used to separate between nodes without and with instances attached. This field can be omitted in case no multi instance node follows.

Multi Instance NodeID1 .. Multi Instance NodeIDn

These fields contain a list of node IDs with instances attached that should be associated with the grouping.

Instance 1 .. Instance n

These fields specify the instance number for a given multi instance node ID.

3.39.2 Multi Instance Association Get Command

The Multi Instance Association Get Command is used to request the current association configuration of a given grouping identifier. The node receiving this Command should answer with Multi Instance Association Report Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_INSTANCE_ASSOCIATION							
Command = MULTI_INSTANCE_ASSOCIATION_GET							
Grouping Identifier							

Grouping identifier (8 bit)

This grouping identifier is used to identify how nodes are grouped together. The grouping identifier values must be a sequence starting from 1. This field can be ignored in case the node only supports one grouping.

3.39.3 Multi Instance Association Report Command

The Multi Instance Association Report Command should be used to report all nodes associated with the matching grouping identifier. Be aware that it's only possible to get information about how many groupings a given node are associated to, but not the total number of groupings. Therefore the Grouping Identifiers should be allocated with care starting from 0x01 to avoid unnecessary overhead in finding the groupings with associations. A remote with up to 6 different groupings there are controlled by 6 buttons numbered 1...6 could use the same numbers as grouping identifiers. The Multi Instance Association Report Command can be sent unsolicited or requested by the Multi Instance Association Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_INSTANCE_ASSOCIATION							
Command = MULTI_INSTANCE_ASSOCIATION_REPORT							
Grouping Identifier							
Max Nodes Supported							
Reports to Follow							
NodeID1							
Node ID2							
…							
Node IDn							
Marker = MULTI_INSTANCE_ASSOCIATION_REPORT_MARKER							
Multi Instance Node ID1							
Instance 1							
Multi Instance Node ID2							
Instance 2							
…							
Multi Instance Node IDn							
Instance n							

Grouping identifier (8 bit)

Refer to description under the Multi Instance Association Set Command.

Max Nodes Supported (8 bit)

Maximum number of nodes grouping identifier above supports.

Reports to Follow (8 bit)

This value indicates how many report frames there is left before the entire node IDs associated with the given grouping identifier is transferred.

NodeID1 .. NodeIDn

Refer to description under the Multi Instance Association Set Command.

Marker (8 bit)

Refer to description under the Multi Instance Association Set Command.

Multi Instance NodeID1 .. Multi Instance NodeIDn

Refer to description under the Multi Instance Association Set Command.

Instance 1 .. Instance n

Refer to description under the Multi Instance Association Set Command.

3.39.4 Multi Instance Association Remove Command

The Multi Instance Association Remove Command is used to remove nodes from a given grouping.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_INSTANCE_ASSOCIATION							
Command = MULTI_INSTANCE_ASSOCIATION_REMOVE							
Grouping identifier							
Node ID1							
Node ID2							
..							
Node IDn							
Marker = MULTI_INSTANCE_ASSOCIATION_REMOVE_MARKER							
Multi Instance Node ID1							
Instance 1							
Multi Instance Node ID2							
Instance 2							
…							
Multi Instance Node IDn							
Instance n							

Grouping identifier

This grouping identifier is used to determine from which grouping the supplied node ID's should be removed.

NodeID1 .. NodeIDn

These fields contain a list of node ID's that should be removed from the specified grouping identifier. In case no node ID's are supplied the whole grouping should be cleared.

Marker (8 bit)

Refer to description under the Multi Instance Association Set Command.

Multi Instance NodeID1 .. Multi Instance NodeIDn

These fields contain a list of node IDs with instances attached that should be removed from the specified grouping identifier.

Instance 1 .. Instance n

Refer to description under the Multi Instance Association Set Command.

3.39.5 Multi Instance Association Supported Groupings Get Command

The Multi Instance Association Supported Groupings Get Command is used request the number of groupings that this node supports.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_INSTANCE_ASSOCIATION							
Command = MULTI_INSTANCE_ASSOCIATION_GROUPINGS_GET							

3.39.6 Multi Instance Association Supported Groupings Report Command

The Multi Instance Association Supported Groupings Report Command is used to report the maximum number of groupings the given node supports. The Multi Instance Association Supported Groupings Report Command can be sent unsolicited or requested by the Multi Instance Association Supported Groupings Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_INSTANCE_ASSOCIATION							
Command = MULTI_INSTANCE_ASSOCIATION_GROUPINGS_REPORT							
Supported Groupings							

Supported Groupings (8 bit)

The number of groupings this node supports.

3.40 Multi Channel Association Command Class, version 2

This Multi Channel Association Command Class is used to enable bindings to nodes comprising of a number of end points. The command class can handle nodes with and without end points.

The Association command class, version 1 is a subset of the Multi Channel Association command class and the data intersection of the two command classes must be mirrored on application level allowing access to the same data via both command classes. A device supporting the Multi Channel Association command class must therefore support the Association command class, version 2 to fulfill the above requirement.

NOTE: A device supporting the Multi Channel Association command class must support the Association command class, version 2.

NOTE: The Association Report will only return node IDs for nodes without end points specified. To obtain associations configured for nodes with and without end points request the Multi Channel Association Report command.

3.40.1 Multi Channel Association Set Command

The Multi Channel Association Set Command is used to add nodes with/without end points to a given grouping identifier. The node receiving the set command should add the nodes received to the nodes already associated by this grouping until the grouping is full.

Note: Remember that routing slaves also must have assigned return routes by a controller using the API call ZW_AssignReturnRoute [2] to all the associated nodes. Delete all return routes by the API call ZW_DeleteReturnRoute before assignment. This is of course necessary for all the associated nodes out of direct range but should always be done default to all the associated nodes to create a reliable and robust network. It's optional whether the return routes are assigned before or after the Multi Channel Association Set command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL_ASSOCIATION							
Command = MULTI_CHANNEL_ASSOCIATION_SET							
Grouping identifier							
Node ID1							
Node ID2							
...							
Node ID n							
Marker = MULTI_CHANNEL_ASSOCIATION_SET_MARKER							
Multi Channel Node ID1							
Bit address	End Point						
Multi Channel Node ID2							
Bit address	End Point						
…							
Multi Channel Node IDn							
Bit address	End Point						

Grouping identifier (8 bit)

This grouping identifier is used to instruct how nodes are grouped together. The grouping identifier values must be a sequence starting from 1. This field can be ignored in case the node only supports one grouping.

NodeID1 .. NodeIDn

These fields contain a list of node IDs that should be associated with the grouping.

Marker (8 bit)

This marker identifier is used to separate between nodes without and with end points attached. This field can be omitted in case no Multi Channel node follows.

Multi Channel NodeID1 .. Multi Channel NodeIDn

These fields contain a list of node IDs with End Points attached that should be associated with the grouping.

Bit address (1 bit)

This bit is set to 1 if the end point(s) is given in a bit mask which makes it possible to address end points in parallel.

Note: Only the first 7 end points are bit addressable.

This bit is set to 0 if the end point is addressed individually.

End Point (7 bit)

The end point(s) that should receive the command. This field must be interpreted based in the “Bit address” value:

Bit address equals 1: Bit 0 is End Point 1, bit 1 is End Point 2 … bit 6 is End Point 7

Bit address equals 0: End Point addresses an individual End Point of the device. Valid values are 1 to 127.

Note: The same node id can occur several times in the Multi Channel Nodes ID list, if associations to several end points are desired.

3.40.2 Multi Channel Association Get Command

The Multi Channel Association Get Command is used to request the current association configuration of a given grouping identifier. The node receiving this command should answer with Multi Channel Association Report command.

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 bit)

This grouping identifier is used to identify how nodes are grouped together. The grouping identifier values must be a sequence starting from 1. This field can be ignored in case the node only supports one grouping.

3.40.3 Multi Channel Association Report Command

The Multi Channel Association Report Command should be used to report all nodes associated with the matching grouping identifier. Be aware that it's only possible to get information about how many groupings a given node are associated to, but not the total number of groupings. Therefore the Grouping Identifiers should be allocated with care starting from 0x01 to avoid unnecessary overhead in finding the groupings with associations. A remote with up to 6 different groupings there are controlled by 6 buttons numbered 1...6 could use the same numbers as grouping identifiers. The Multi Channel Association Report command can be send unsolicited or as a result of receiving a Multi Channel Association Get command.

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							
NodeID1							
Node ID2							
…							
Node IDn							
Marker = MULTI_CHANNEL_ASSOCIATION_REPORT_MARKER							
Multi Channel Node ID1							
Bit address	End Point						
Multi Channel Node ID2							
Bit address	End Point						
…							
Multi Channel Node IDn							
Bit address	End Point						

Grouping identifier (8 bit)

Refer to description under the Multi Channel Association Set command.

Max Nodes Supported (8 bit)

Maximum number of nodes grouping identifier above supports.

Reports to Follow (8 bit)

This value indicates how many report frames there is left before the entire node IDs associated with the given grouping identifier is transferred.

NodeID1 .. NodeIDn

Refer to description under the Multi Channel Association Set command.

Marker (8 bit)

Refer to description under the Multi Channel Association Set command.

Multi Channel NodeID1 .. Multi Channel NodeIDn

Refer to description under the Multi Channel Association Set command.

Bit address (1 bit)

Refer to description under the Multi Channel Association Set command.

End Point (7 bit)

Refer to description under the Multi Channel Association Set command.

3.40.4 Multi Channel Association Remove Command

The Multi Channel Association Remove Command is used to remove nodes from a given grouping.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL_ASSOCIATION							
Command = MULTI_CHANNEL_ASSOCIATION_REMOVE							
Grouping identifier							
Node ID1							
Node ID2							
..							
Node IDn							
Marker = MULTI_CHANNEL_ASSOCIATION_REMOVE_MARKER							
Multi Channel Node ID1							
Bit address	End Point						
Multi Channel Node ID2							
Bit address	End Point						
…							
Multi Channel Node IDn							
Bit address	End Point						

Grouping identifier

This grouping identifier is used to determine from which grouping the supplied node ID's should be removed. If no group ID or group ID zero is supplied all groups should be cleared.

NodeID1 .. NodeIDn

These fields contain a list of node ID's that should be removed from the specified grouping identifier. In case no node ID's are supplied the whole grouping should be cleared.

Marker (8 bit)

Refer to description under the Multi Channel Association Set command.

Multi Channel NodeID1 .. Multi Channel NodeIDn

Refer to description under the Multi Channel Association Set command.

Bit address (1 bit)

Refer to description under the Multi Channel Association Set command.

End Point (7 bit)

Refer to description under the Multi Channel Association Set command.

3.40.5 Multi Channel Association Supported Groupings Get Command

The Multi Channel Association Supported Groupings Get Command is used request the number of groupings that this node supports.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL_ASSOCIATION							
Command = MULTI_CHANNEL_ASSOCIATION_GROUPINGS_GET							

3.40.6 Multi Channel Association Supported Groupings Report Command

The Multi Channel Association Supported Groupings Report Command is used to report the maximum number of groupings the given node supports. The Multi Channel Association Supported Groupings Report command can be send unsolicited or as a result of receiving a Multi Channel Association Supported Groupings Get command.

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 bit)

The number of groupings this node supports.

3.41 Multi Instance Command Class, version 1

Do not use this command class for new devices.

Use instead Multi Channel Command Class version 2 for such devices.

The Multi Instance Command Class is used to control multiple instances of a given functionality in a device. When the Multi Instance Command Class is advertised in the NIF , the controller needs to ask the slave for the number of instances of each supported Command Class. This implies that a device can have multiple instances of several different Command Classes at the same time.

3.41.1 Multi Instance Get Command

The Multi Instance Get Command will be used to get the number of instances of a specific Command Class.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_INSTANCE							
Command = MULTI_INSTANCE_GET							
Command Class							

Command Class (8 bit)

The Command Class field indicates what Command Class the get Command is referring to.

3.41.2 Multi Instance Report Command

The Multi Instance Report Command reports the number of instances of a given Command Class in a device. The Multi Instance Report Command can be sent unsolicited or requested by the Multi Instance Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_INSTANCE							
Command = MULTI_INSTANCE_REPORT							
Command Class							
Instances							

Command Class (8 bit)

The Command Class field indicates what Command Class the report Command is referring to.

Instances (8 bit)

Number of instances supported by the device.

3.41.3 Multi Instance Command Encapsulation Command

The Multi Instance Command Encapsulation Command is used to encapsulate other Commands that refer to specific instances of a given Command Class.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_INSTANCE							
Command = MULTI_INSTANCE_CMD_ENCAP							
Instance							
Command Class							
Command							
Parameter 1							
...							
Parameter N							

Instance (8 bit)

The instance field points to the instance that the embedded Command should refer to. The instance values must be a sequence starting from 1.

Command Class (8 bit)

Command class identifier of embedded command class.

Command (8 bit)

Command identifier of embedded command class.

Parameter 1 … Parameter N (variable)

Parameters attached to embedded command class.

3.42 Multi Channel Command Class, version 2

This section contains commands that can be used to control a Multi Channel Command Class.

The Multi Channel command class is used to control one or more end points in a given device that supports this command class. It is required that an application using this command class sets the optional functionality bit in the NIF.

Multi Channel devices can have from 1 to 127 end points that can be controlled individually. The first 7 end points can be controlled together in any combination. This command class defines how the capabilities of the end points are communicated in a Z-Wave network and how the end points are addressed when other nodes needs to communicate with them.

It is up to the application developer to define how a Multi Channel device reacts to un-encapsulated commands. The reaction must however be within these guidelines:

- Un-encapsulated commands must trigger an action and/or response that accurately reflect the basic functionality of the Multi Channel device in question[1]. This must be enforced to preserve backwards compatibility with simple controllers not able to address individual end points.

- Un-encapsulated commands must not limit the functionality, or enable non-compliant behavior by any End Point in the device.

- For simplicity un-encapsulated command can be forwarded to End Point 1 in the Multi Channel device. If this is the case the End Point capabilities of End Point 1 must be reported in the standard Z-Wave NIF.
  Note: For this case End point 1 can therefore not be a dynamic end point (more details see 3.42.2.2).

IMPORTANT NOTE:
Please be aware that the identifiers for the new Multi Channel command class is the same as the Multi Instance command class and the new Multi Channel Association command class identifier is the same as the Multi Instance Association command class. For this reason the two new command classes will start with version 2. In this way this new command class will be backward compatible with any existing products implementing the Multi Instance command class.
There are two exceptions to the backward capability:

1. The number of instances are changed from 255 in version 1 to 127 in version 2
2. Multi Instance devices cannot control Multi Channels where the end points are not identical

This structure means any controller that needs to control a device that either implements Multi Channel command class or Multi Channel Association command class MUST interview the device for the version before using these classes. If the Version Command Class is not supported or version 1 is reported the controller MUST use the Multi Instance and Multi Instance Association command classes.

Please note that it is required for all devices supporting the Multi Channel Command Class and / or the Multi Channel Association Command Class to also support the Version Command Class.

Please refer to chapter 3.42.3 for implementation recommendation.

[1] *E.g. for a power strip with 5 outlets designed as a Multi Channel device, an un-encapsulated Basic Off command could turn off all outlets in the power strip, and a un-encapsulated Basic On could turn On all outlets.*

3.42.1 Multi Instance Commands

This first part of the Multi Channel command class defines the new Multi Instance commands that makes the Multi Channel command class backward compatible with the old Multi Instance command class.

3.42.1.1 Multi Instance Get Command

The Multi Instance Get Command will be used to get the number of instances of a specific Command Class.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_INSTANCE_GET							
Command Class							

Command Class (8 bit)

The Command Class field indicates what Command Class the get command is referring to.

3.42.1.2 Multi Instance Report Command

The Multi Instance Report Command reports the number of instances of a given Command Class in a device. The Multi Instance Report command can be send unsolicited or as a result of receiving a Multi Instance Get command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_INSTANCE_REPORT							
Command Class							
Res	Instances						

Command Class (8 bit)

The Command Class field indicates what Command Class the report command is referring to.

Res (1 bit)

This bit is reserved bit. Must be set to 0 by devices sending this frame and must be ignored by devices receiving this frame.

Instances (7 bit)

The instances of the requested command class that are supported by the device.

Valid values are 0 to 127.

Note: If end points in a Multi Channel device are not identical the device can only support the command classes supported by the first endpoint using the multi instance command class.

3.42.1.3 Multi Instance Command Encapsulation Command

The Multi Instance Command Encapsulation Command is used to encapsulate commands send to a Z-Wave node implementing version 1 of the Multi Channel command class.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_INSTANCE_CMD_ENCAP							
Res	Instance						
Command Class							
Command							
Parameter 1							
...							
Parameter N							

Res (1 bit)

This bit is reserved bit. Must be set to 0 by devices sending this frame and must be ignored by devices receiving this frame.

Instance (7 bit)

The interpretation of the instance number is depended on the command encapsulated in the frame. Valid values are 1 to 127.

<u>If the command is a request (e.g. Binary Switch Get):</u>
The number indicates the “destination instance”. This is the instance that should receive the encapsulated command.

<u>If the command is a reply to a Multi Instance encapsulated request (e.g. Binary Switch Report):</u>
The number indicates the “source instance”. This is the instance that did send the encapsulated command.

Note: All replies send based on an encapsulated request MUST be encapsulated in a Multi Instance Command Encapsulation command.

Note: Extreme care should be taken if using the Multi Instance Command Encapsulation for unsolicited replies (e.g. Binary Sensor Report). It is generally not recommended to use Multi Instance Command Encapsulation for unsolicited replies of high importance as there are no guarantee the receiving device will interpret the command. If the device must send unsolicited commands from an endpoint the commands must be sent using the Multi Instance Command Encapsulation as the default method. Multi Channel Command Encapsulation can only be used if the device has extended information about the receiver.

Note: All command classes and commands can be encapsulated; however care should be taken when encapsulating management command such as configuration and association command classes. These will only affect the end point addressed and not the device in whole.

Note: It is allowed to encapsulate this command into another type of encapsulation, and to encapsulate another type of encapsulation into this command. BUT it is not allowed to have multiple encapsulations of the same type e.g. Multi Instance / Channel Command Encapsulations or Multi Command Encapsulations nested into each other.

Command Class (8 bit)

Command class identifier of embedded command class.

Command (8 bit)

Command identifier of embedded command class.

Parameter 1 .. Parameter N (variable)

Parameters attached to embedded command class. The number of fields transmitted can be determined from the length field in the frame.

3.42.2 Multi Channel Commands

This second part of the Multi Channel command class defines all new command that must be implemented and used in all new multi channel devices.

3.42.2.1 Multi Channel End Point Get Command

The Multi Channel End Point Get Command is used to get the number of end points embedded in a single node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_END_POINT_GET							

3.42.2.2 Multi Channel End Point Report Command

The Multi Channel End Point Report Command reports the number of end points embedded in a single node. The Multi Channel End Point Report command can only be send as a result of receiving a Multi Channel End Point Get command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_END_POINT_REPORT							
Dyna-mic	Iden-tical	Res					
Res	End Points						

Res (6 bit) and (1 bit)

These bits are reserved bit. Must be set to 0 by devices sending this frame and must be ignored by devices receiving this frame.

End Points (7 bit)

Number of end points embedded in the node. The maximum number of end points is 127.

Identical (1 bit)

This bit is set to 1 if all the end points in the node has the same generic and specific command class and supports the same optional command classes.

Dynamic (1 bit)

This field is set to 1 if the device has a dynamic number of end points. When the dynamic bit is set the number of end points in the device can change over time.

Warning: Care should be taken when communicating with dynamic end points as the transmitter cannot be entirely sure the specific end point exists.

Note: Implementation of a device supporting dynamic end points must follow the guide lines given in 3.42.3.3. Future definitions may be defined, always refer to the newest version of the command class specification.

3.42.2.3 Multi Channel Capability Get Command

The Multi Channel Capability Get Command is used to get the capabilities of the end points in a node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_CAPABILITY_ GET							
Res	End Point						

Res (1 bit)

This bit is reserved bit. Must be set to 0 by devices sending this frame and must be ignored by devices receiving this frame.

End Point (7 bit)

End Point number to get capabilities from.

3.42.2.4 Multi Channel Capability Report Command

The Multi Channel Capability Report Command reports the generic and specific device class of the end point and the supported command classes of the end point. The Multi Channel Capability Report command can be send unsolicited or as a result of receiving a Multi Channel Command Get command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_CAPABILITY_REPORT							
Dyna-mic	End Point						
Generic Device Class							
Specific Device Class							
Command Class 1							
Command Class 2							
…							
Command Class N							

End Point (7 bit)

The End point field indicates what end point number the report frame is referring to.

Dynamic (1 bit)

This field is set to one if this end point is a dynamic end point. When this bit is set in an end point it can not be assumed that it will reply to commands send to it because it could be gone again when a command is send to it.

Note: End point 1 can not be dynamic

Generic Device class (8 bit)

The generic device class of the specified end point.

Specific Device class (8 bit)

The specific device class of the specified end point.

Command Class 1 .. Command Class N (N*8 bit)

Command classes supported or controlled by the device in question. The number of fields transmitted can be determined from the length field in the frame.

Note: For memory reasons it is not required for a controlling device to save the capabilities of each end point. However controlling devices should be able to control with at least the basic command class for each end point.

3.42.2.5 Multi Channel End Point Find Command

The Multi Channel End Point Find Command is used to find end points in a device with a given set of generic and specific device class.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_END_POINT_FIND							
Generic Device Class							
Specific Device Class							

Generic Device Class (8 bit)

The generic device class that should be found in the Multi Channel device.

Specific Device Class (8 bit)

The specific device class that should be found in the Multi Channel device. If 0xFF is specified in this field then all devices with the specified generic device class will be returned.

3.42.2.6 Multi Channel End Point Find Report Command

The Multi Channel End Point Find Report Command is used to reply to a Multi Channel End Point Find command. This command can only be send as a response to a Multi Channel End Point Find command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_END_POINT_FIND_REPORT							
Reports to Follow							
Generic Device Class							
Specific Device Class							
Res	End Point 1						
…							
Res	End Point n						

Res (1 bit)

This bit is reserved bit. Must be set to 0 by devices sending this frame and must be ignored by devices receiving this frame.

End Point 1 … n (7 bit)

The end point(s) that matches the generic and specific device class send in the get command.

Reports to Follow (8 bit)

This value indicates how many report frames there is left before the end points matching the given generic and specific device class is transferred.

3.42.2.7 Multi Channel Command Encapsulation Command

The Multi Channel Command Encapsulation Command is used to encapsulate commands send to a Multi Channel device so it can address one or several end points in a Z-Wave node. Any command that the end point reports that it supports can be encapsulated using this command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_CMD_ENCAP							
Res	Source End Point						
Bit address	Destination End Point						
Command Class							
Command							
Parameter 1							
...							
Parameter N							

Res (1 bit)

This bit is reserved bit. Must be set to 0 by devices sending this frame and must be ignored by devices receiving this frame.

Source End Point (7 bit)

This field indicates the end point from where the command was send. Valid Source End Points are 1 to 127. The Source End Point must be used as destination in the case the encapsulated frame contains a request to the destination.

If the sending device does not support multiple channels the Source End Point must be set to 0. Please note that in this case a response to a GET command will be send encapsulated back to End Point 0.

Bit address (1 bit)

This bit is set to 1 if the end point(s) is given in a bit mask which makes it possible to address end points in parallel.

This bit is set to 0 if the end point is addressed individually.

Note: Only the first 7 end points are bit addressable.

Note: If the encapsulated command is a request (requiring a reply from the destination) it is not allowed to bit address the frame. This is prohibited to decrease implementation complexity of devices supporting this command class.

Destination End Point (7 bit)

The end point(s) that should receive the command. This field must be interpreted based in the “Bit address” value:

Bit address equals 1: Bit 0 is End Point 1, bit 1 is End Point 2 … bit 6 is End Point 7

Bit address equals 0: End Point addresses an individual End Point of the device. Valid values are 1 to 127.

Command Class (8 bit)

Command class identifier of embedded command class.

Command (8 bit)

Command identifier of embedded command class.

Parameter 1 .. Parameter N (variable)

Parameters attached to embedded command class. The number of fields transmitted can be determined from the length field in the frame.

Note: All command classes and commands can be encapsulated; however care should be taken when encapsulating management command such as configuration and association command classes. These will only affect the end point addressed and not the device in whole.

Note: If the command encapsulated in this frame is a request, the reply must be encapsulated in a Multi Channel Command Encapsulation frame.

Note: It is allowed to encapsulate this command into another type of encapsulation, and to encapsulate another type of encapsulation into this command. BUT it is not allowed to have multiple encapsulations of the same type e.g. Multi Instance / Channel Command Encapsulations or Multi Command Encapsulations nested into each other.

3.42.3 Implementation Recommendation

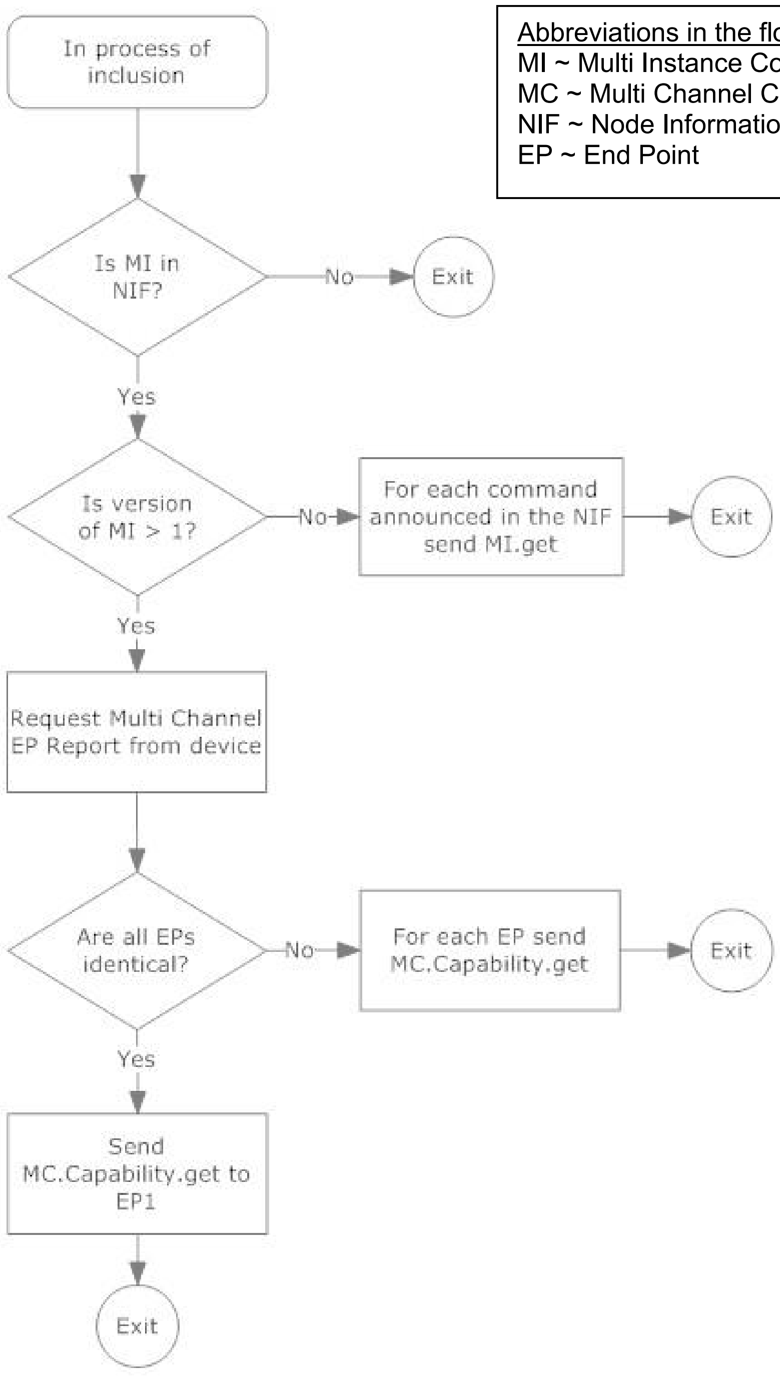
In order to ensure interoperability between devices implementing the Multi Channel command class this section give a recommendation on how to initial interview any Multi Channel devices. Furthermore some examples are shown on "normal operation" between Multi Channel devices.

3.42.3.1 Controlling Devices

Interview

The interview process during inclusion of a Multi Channel devices must be able to handle both version 1 (the old Multi Instance command class) and the new Multi Channel command class version 2. Below is a flowchart showing a diagram of the decision process when including an arbitrary Multi Channel device:

Abbreviations in the flowchart:
MI ~ Multi Instance Command Class
MC ~ Multi Channel Command Class
NIF ~ Node Information Frame
EP ~ End Point


In process of inclusion
Is MI in NIF?
No
Exit
Yes
Is version of MI > 1?
No
For each command announced in the NIF send MI.get
Exit
Yes
Request Multi Channel EP Report from device
Are all EPs identical?
No
For each EP send MC.Capability.get
Exit
Yes
Send MC.Capability.get to EP1
Exit

Note: It is not required for a controlling device to store all information about each available end point.

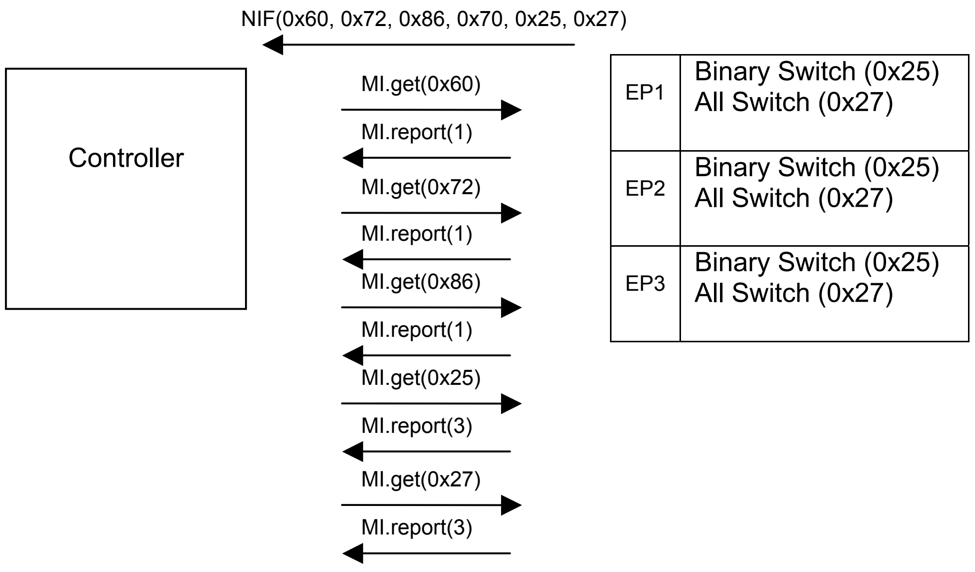
Examples

This section gives 2 examples on how interoperability is maintained between the Multi Instance Command Class and the Multi Channel Command Class.

Device with identical End Points

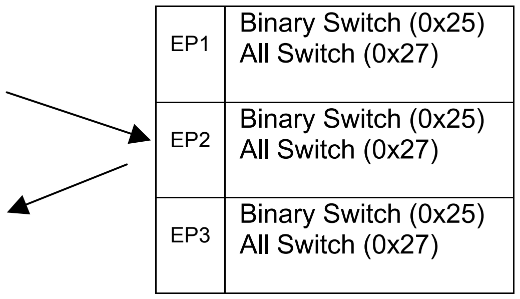
The first example shows how a controller implementing version 1 of Multi Channel (Multi Instance) command class will interview and operate a device implementing Multi Channel command class with identical End Points:

The NIF include the commands: Multi Channel (0x60), Manufacturer Specific (0x72), Version (0x86), Binary Switch (0x25), All Switch (0x27)



The controller can now operate the 3 Binary Switch and All Switch commands in the device sending a Multi Instance Command Encapsulation command addressing the End Points:

7	6	5	4	3	2	1	0
COMMAND_CLASS_MULTI_CHANNEL							
MULTI_INSTANCE_CMD_ENCAP							
0	Instance=2						
Command Class=Binary Switch							
Get							



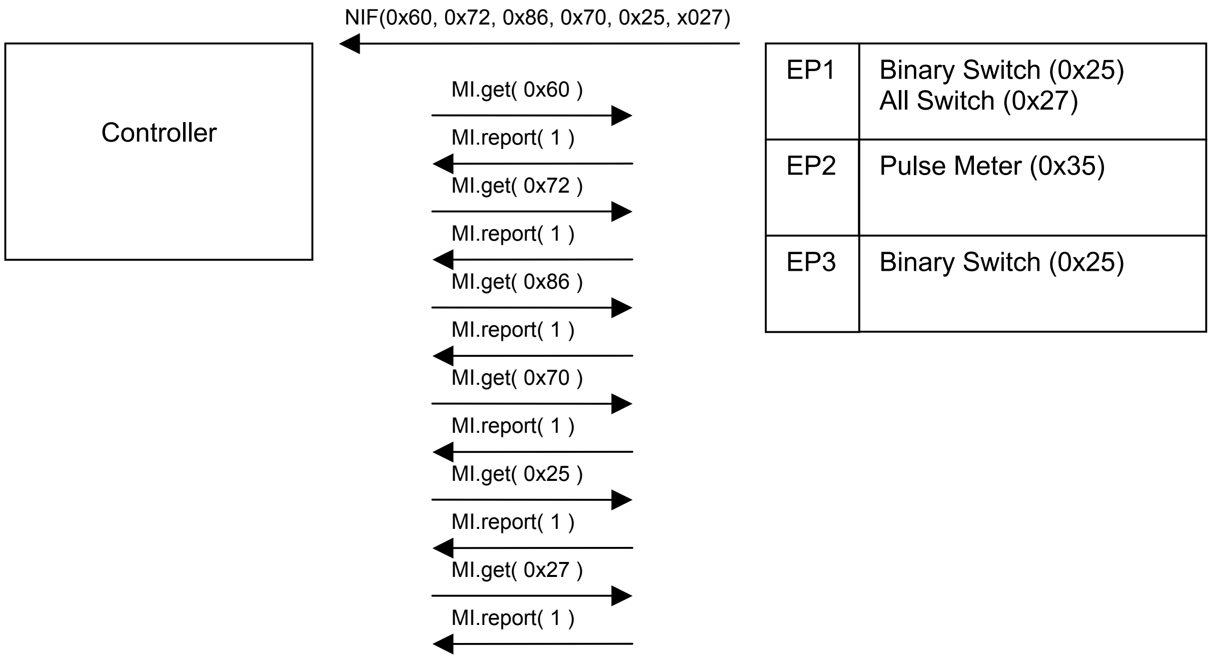
7	6	5	4	3	2	1	0
COMMAND_CLASS_MULTI_CHANNEL							
MULTI_INSTANCE_CMD_ENCAP							
0	Instance=2						
Command Class=Binary Switch							
Command Class = Binary Report							
Value 0x00/0xFF							

The Binary Switch get command above is send to End Point 2 encapsulated in a Multi Instance Command Encapsulation frame. The Instance indicates the destination End Point while the answer (the Binary Switch Report) is encapsulated in a Multi Instance Command Encapsulation frame where the Instance indicates the source End Point.

Device where End Points are not identical

This next example is a Multi Channel device where End Points have different command classes implemented.

The NIF include the commands: Multi Channel (0x60), Manufacturer Specific (0x72), Version (0x86), Configuration (0x70), Binary Switch (0x25), All Switch (0x27),

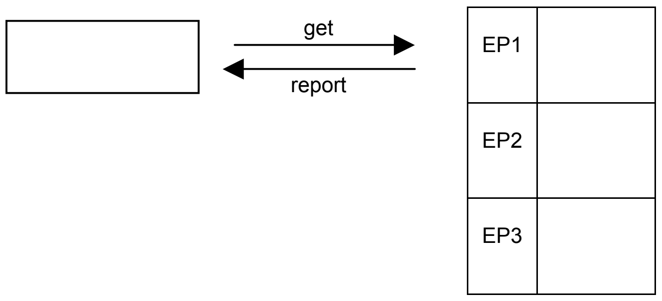


The Multi Channel device must report that the numbers of instances are 1 for each command class in the first end point since the End Points are not identical. In this case it is not possible for the controller to send commands to End Point 2 and 3.

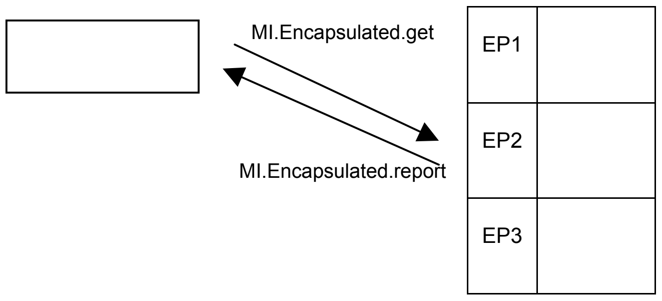
3.42.3.2 Supporting Devices

A device supporting Multi Channel command class must “answer as asked”. There are 3 possible ways of sending a report to a request:

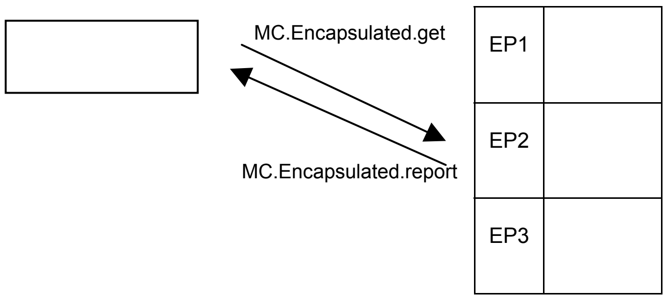
A.) Non-encapsulated frames



B.) Multi Instance Command Encapsulation



C.) Multi Channel Command Encapsulation



3.42.3.3 Supporting Dynamic End Points

When implementing a device supporting dynamic End Points the following guide lines must be followed for adding and removing End Points.

Adding an End Point

When adding a new End Point it must be numbered in succession. If the next consecutive number is 128, the device must search from the start of the list for the first empty entry.

Examples adding end points

The following examples show how End Points are inserted into the list of End Points.

Adding an End Point

End Point list before

1	Occupied
2	Occupied
3	Free
…	
126	Free
127	Free

End Point list after

1	Occupied
2	Occupied
3	Occupied
…	
126	Free
127	Free

Adding an End Point

End Point list before

1	Occupied
2	Free
3	Occupied
…	
126	Occupied
127	Free

End Point list after

1	Occupied
2	Free
3	Occupied
…	
126	Occupied
127	Occupied

Adding an End Point

End Point list before

1	Occupied
2	Free
3	Occupied
…	
126	Occupied
127	Occupied

End Point list after

1	Occupied
2	Occupied
3	Occupied
…	
126	Occupied
127	Occupied

Removing an End Point

When removing an End Point, the End Point must leave an empty spot in the End Point list, thus making sure there is no change in the numbering of the other End Points supported by the device.

Example removing end point

The following example show how End Points are removed from the list of End Points.

Remove End Point #2

End Point list before

1	Occupied
2	Occupied
3	Occupied
…	
126	Free
127	Free

End Point list after

1	Occupied
2	Free
3	Occupied
…	
126	Free
127	Free

3.43 Multilevel Sensor Command Class, version 1-3

This section contains Commands that can be used to control a multilevel sensor.

3.43.1 Multilevel Sensor Get Command

The Multilevel Sensor Get Command is used to request the level of a multilevel sensor. The Multilevel Sensor Get Command versions 1-3 have the same layout.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SENSOR_MULTILEVEL							
Command = SENSOR_MULTILEVEL_GET							

3.43.2 Multilevel Sensor Report Command

This Command can be used by a multilevel sensor to send a report either unsolicited or requested by the Multilevel Sensor Get Command. The Multilevel Sensor Report Command version 2 and 3 are extensions with respect to Sensor Types and associated Scales.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SENSOR_MULTILEVEL							
Command = SENSOR_MULTILEVEL_REPORT							
Sensor Type							
Precision			Scale		Size		
Sensor Value 1							
Sensor Value 2							
..							
Sensor Value n							

Sensor Type (8 bit)

Sensor type specifies what type of sensor this Command originates from. Refer to the table below with respect to defined sensors. New sensor types/values can be requested from Zensys.

Sensor Type	Value
Temperature (version 1)	0x01
General purpose value (version 1)	0x02
Luminance (version 1)	0x03
Power (version 2)	0x04
Relative humidity (version 2)	0x05
Velocity (version 2)	0x06
Direction (version 2)	0x07
Atmospheric pressure (version 2)	0x08
Barometric pressure (version 2)	0x09
Solar radiation (version 2)	0x0A
Dew point (version 2)	0x0B
Rain rate (version 2)	0x0C
Tide level (version 2)	0x0D
Weight (version 3)	0x0E
Voltage (version 3)	0x0F
Current (version 3)	0x10
CO_2-level (version 3)	0x11
Air flow (version 3)	0x12
Tank capacity (version 3)	0x13
Distance (version 3)	0x14

Precision (3 bit)

The precision field describes what the precision of the sensor value is. The number indicates the number of decimals. The decimal value 1025 with precision 2 is therefore equal to 10.25.

Scale (2 bit)

The Scale is used to indicate what unit the sensor uses. Refer to the table below with respect to defined scales for the relevant sensors. New scales/values can be requested from Zensys.

Sensor Type	Scale	Value
Air temperature (version 1)	Celsius (C)	0x00
	Fahrenheit (F)	0x01

General purpose value (version 1)	Percentage value	0x00
	Dimensionless value	0x01
Luminance (version 1)	Percentage value	0x00
	Lux	0x01
Power (version 2)	W	0x00
	Btu/h	0x01
Relative humidity (version 2)	Percentage value	0x00
Velocity (version 2)	m/s	0x00
	mph	0x01
Direction (version 2)	0 to 360 degrees. 0 = no wind, 90 = east, 180 = south, 270 = west, and 360 = north	0x00
Atmospheric pressure (version 2)	kPa	0x00
	inches of Mercury	0x01
Barometric pressure (version 2)	kPa	0x00
	inches of Mercury	0x01
Solar radiation (version 2)	W/m²	0x00
Dew point (version 2)	Celsius (C)	0x00
	Fahrenheit (F)	0x01
Rain rate (version 2)	mm/h	0x00
	in/h	0x01
Tide level (version 2)	m	0x00
	feet	0x01
Weight (version 3)	kg	0x00
	pounds	0x01
Voltage (version 3)	V	0x00
	mV	0x01

Current (version 3)	A	0x00
	mA	0x01
CO2-level (version 3)	ppm	0x00
Air flow (version 3)	m³/h	0x00
	cfm (cubic feet per minute)	0x01
Tank capacity (version 3)	l	0x00
	cbm	0x01
	US gallons	0x02
Distance (version 3)	m	0x00
	cm	0x01
	feet	0x02

Size (3 bit)

The size field indicates the number of bytes that is used for the sensor value. This field can take values from 1 (001b), 2 (010b) or 4 (100b).

Sensor Value (variable)

The sensor value is a signed field. The field can be 1, 2 or 4 bytes in size. The first byte is the most significant byte. The first byte is the most significant byte. The table below shows signed decimal values together with their hexadecimal equivalents.

Signed 1 byte decimal value	Hexadecimal	Signed 2 bytes decimal value	Hexadecimal
127	0x7F	32767	0x7FFF
25	0x19	1025	0x0401
2	0x02	2	0x0002
1	0x01	1	0x0001
0	0x00	0	0x0000
-1	0xFF	-1	0xFFFF
-2	0xFE	-2	0xFFFE
-25	0xE7	-1025	0xFBFF
-128	0x80	-32768	0x8000

Notice: The device receiving the Multilevel Sensor Report must always show the sensor value even though the Sensor Type and/or Scale are not supported.

3.44 Multilevel Switch Command Class, version 1

This section contains Commands that can be used to control a multilevel switch. These Commands allow applications to turn a multilevel switch on/off, start/stop dimming/operation, jumping/dimming to a specified level, and read the current level.

3.44.1 Multilevel Switch Set Command

The Multilevel Switch Set Command version 1 can be used to set the level in a device that supports the multilevel switch functionality. The speed that the switch increases or decreases the level with is implementation specific.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_MULTILEVEL							
Command = SWITCH_MULTILEVEL_SET							
Value							

Value (8 bit)

The value can be either 0x00 (off/disable) or 0xFF (on/enable). Furthermore it can take values from 1 to 99 (0x01 – 0x63). The unit of the value is implementation specific i.e. percentage. Devices shall implement all values.

The value 0xFF (on/enable) will cause the device to change to the level when the device last was turned on. With respect to motor controls both 0x63 and 0xFF may be interpreted as fully open.

The values 100…254 (0x64…0xFE) are reserved and shall be ignored by the receiving device.

Controlling devices may send any of the values 0x00…0x63 and 0xFF to the device. Controlling devices must not send any of the reserved values to the device.

Devices receiving this Command are free in the mapping of the received values 1…99 (0x01…0x63) to an internal representation of their setting, except that for a mapping to percentages the value 0x63 represents 100%. The mapping of the value received should be monotonous i.e. a higher value in a set Command should always result in either a higher or same level.

It is not required that a device implements 100 distinct setting levels. If a device implements less than 100 internal setting levels, then the *active* internal setting levels should be spread equally over the entire range of 1…99 (0x01…0x63).

<u>Example:</u>
Recommended behavior of a device implementing three light levels:

0	OFF	(inactive / OFF setting)
1…33	1/3 On	(3 *active* settings → spread over 99 values)
34…66	2/3 On	
67…99	FULL ON	

3.44.2 Multilevel Switch Get Command

The Multilevel Switch Get Command version 1 can be used to request the status of a multilevel switch.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_MULTILEVEL							
Command = SWITCH_MULTILEVEL_GET							

3.44.3 Multilevel Switch Report Command

The Multilevel Switch Report Command version 1 can be sent unsolicited or requested by the Multilevel Switch Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_MULTILEVEL							
Command = SWITCH_MULTILEVEL_REPORT							
Value							

Value (8 bit)

The value can be either 0x00 (off/disable) or 0xFF (on/enable). Furthermore it can return values from 1 to 99 (0x01 – 0x63). Devices must not respond with a Multilevel Switch Report with any other value.

It is neither specified that a device will respond in a Report with exactly the value sent in a Set Command, nor that a device will ever use all permitted values in Reports.

<u>Note:</u>
If a controlling device is implementing the dimming Commands and want to align all the dimmers when dimming is stopped requires extra precaution: The get Command that is used to enquire the end level from one device must be from a Multilevel Switch to avoid any unintentionally jumps in light level.

3.44.4 Multilevel Switch Start Level Change Command

The Multilevel Switch Start Level Change Command version 1 can be used to inform a multilevel switch, that it should start changing the level. The speed that the switch increases or decreases the level with is implementation specific.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_MULTILEVEL							
Command = SWITCH_MULTILEVEL_START_LEVEL_CHANGE							
Re-ser-ved	Up/ Down	Ignore Start Level	Reserved				
Start Level							

Up/Down (1 bit)

If the Up/Down bit is set to 0 the switch shall increase its level. If field is set to 1 the switch shall decrease its level.

Ignore Start Level (1 bit)

An Ignore Start Level bit set to 0 indicates to use the start level specified in the Command. An Ignore Start Level bit set to 1 indicates to start from the actual level in the device.

A device that supports Multilevel Switch Commands is not required to implement the Ignore Start Level bit. If not implemented, the behavior shall be as if the Ignore Start Level bit is 1.

A controlling device is not required to implement the Ignore Start Level bit. If not implemented, the Ignore Start Level bit shall always be set to 1 by a device sending this Command.

Note:
While many types of devices that can be supported by Multilevel Switch Commands are capable of jumping immediately to a specified start level, some types of devices cannot change their level immediately (e.g. High Intensity Discharge lamps, motor controlled devices). For this reason, it is permitted for devices not to implement the start level as specified above. For dimmers it is recommended to implement both values of the Ignore Start Level bit.

Reserved (1+5 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Start Level (8 bit)

The start level field contains the initial level for the device to assume when starting to change the level.

3.44.5 Multilevel Switch Stop Level Change Command

The Multilevel Switch Stop Level Change Command version 1 can be used to inform a multilevel switch, that it should stop changing the level.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_MULTILEVEL							
Command = SWITCH_MULTILEVEL_STOP_LEVEL_CHANGE							

3.45 Multilevel Switch Command Class, version 2

This section contains Commands to control a multilevel switch. These Commands allow applications to turn a multilevel switch on/off, start/stop dimming/operation at a given rate, jumping/dimming to a specified level for a given duration, and read the current level. Version 2 is extended with respect to a parameter specifying the duration/rate of the wanted operation for the Multilevel Switch Set and Multilevel Switch Start Level Change Commands.

3.45.1 Multilevel Switch Set Command

The Multilevel Switch Set Command version 2 can be used to set the level in a device that supports the multilevel switch functionality. The speed that the switch increases or decreases the level with is implementation specific.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_MULTILEVEL							
Command = SWITCH_MULTILEVEL_SET							
Value							
Dimming Duration							

Value (8 bit)

The value can be either 0x00 (off/disable) or 0xFF (on/enable). Furthermore, it can take values from 1 to 99 (0x01 – 0x63). The unit of the value is implementation specific i.e. percentage. All values must be implemented, but the actual implementation of the values 0x01…0x63 is not defined, except that for a mapping to percentages the value 0x63 represents 100%. The mapping of the value received should be monotonous i.e. a higher value in a set Command will always result in either a higher or the same setting. If a device implements less than distinct 100 values in its internal setting, then the mapping of the active internal setting should be spread equally over the entire range of 0x00…0x63.

The value 0xFF (on/enable) will cause the device to jump to the level when the device last was turned on.

All other values are ignored by the receiving device. These values are reserved for future use.

Controlling devices may send any of the values 0x00…0x63 and 0xFF to the device. Controlling devices must not send any of the reserved values 0x64…0xFE to the device.

Dimming Duration (8 bit)

The dimming duration can either be instantly or it can be a dimming duration, or a factory default dimming duration. The dimming duration parameter is the only addition to version 1 to create a version 2 Command. The table below shows how to obtain the wanted functionality:

Dimming Duration	Description
0x00	Instantly
0x01-0x7F	Obtain dimming durations from 1 second (0x01) to 127 seconds (0x7F) in 1-second resolution. The dimming duration is defined as the interval between start of dimming and until the specified level is reached.
0x80-0xFE	Specify dimming durations from 1 minute (0x80) to 127 minutes (0xFE) in 1-minute resolution. The dimming duration is defined as the interval between start of dimming and until the specified level is reached.
0xFF	Factory default dimming duration or a rate compatible with a version 1 implementation of the Multilevel Command Class.

It is allowed to make a device supporting version 2 of the Multilevel Switch Command Class which at the same time is backward compatible to a device supporting version 1 of the Multilevel Switch Command Class with respect to the dimming profile. This can be done by implementing a version 1 dimming profile when specifying a dimming duration equal to 0xFF for a device supporting version 2 of the Multilevel Switch Command Class. Remember to default initialize the dimming duration to 0xFF for a device supporting version 2 of the Multilevel Switch Command Class to be backward compatible in case it is controlled by a Multilevel Switch Command Class version 1.

3.45.2 Multilevel Switch Get Command

The Multilevel Switch Get Command version 2 is similar to version 1.

3.45.3 Multilevel Switch Report Command

The Multilevel Switch Report Command version 2 is similar to version 1.

3.45.4 Multilevel Switch Start Level Change Command

The Multilevel Switch Start Level Change Command version 2 can be used to inform a multilevel switch, that it should start changing the level. The speed that the switch increases or decreases the level with is implementation specific.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_MULTILEVEL							
Command = SWITCH_MULTILEVEL_START_LEVEL_CHANGE							
Re-ser-ved	Up/ Down	Ignore Start Level	Reserved				
Start Level							
Dimming Duration							

Up/Down (1 bit)

If the Up/Down bit is set to 0 the switch should increase the level. If field is set to 1 the switch should decrease the level.

Ignore Start Level (1 bit)

If the Ignore Start Level bit is set to 0 the switch should use the start level specified in the Command. If field is set to 1 the switch should start from the actual level in the device.

Reserved (1 + 5 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Start Level (8 bit)

The start level field contains the initial level that the switch should assume when it start to change the level. Most devices are capable of jumping immediately to the specified start level. But some devices the Multilevel Switch Command Class also was intended for can not change level immediately such as HID (High Intensity Discharge) lamps, motor controlled devices etc. Therefore are Z-Wave enabled devices that are not capable to change level immediately allowed to use the current level as a starting point even though a start level is specified in the Command.

Dimming Duration (8 bit)

Refer to description under the Multilevel Switch Set Command version 1. The only deviation is how the dimming duration is defined. The Multilevel Switch Start Level Change only starts dimming but does not define when to stop. The Multilevel Switch Stop Level Change Command is required to stop dimming. This requires that the dimming duration must be defined according to a fixed reference. The dimming duration is therefore in this case defined as the interval it takes to dim from level 0 to 99. The Multilevel Switch Start Level Change can now determine the dimming rate to use. The dimming duration parameter is the only addition to version 1 to create a version 2 Command.

3.45.5 Multilevel Switch Stop Level Change Command

The Multilevel Switch Stop Level Change Command version 2 is similar to version 1.

3.46 Multilevel Switch Command Class, version 3

Version 3 of the Multilevel Switch Command Class version 3implements the option to retrieve a switch type from the device for the controller to determine the behavior in terms of movement. In addition a general increment and decrement mechanism has been added to the Start Level Change command.

Commands not described in Version 3 stays unchanged from Version 2.

3.46.1 Multilevel Switch Supported Get Command

The Multilevel Switch Supported Get Command is used to interview the device for the Switch Type.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_MULTILEVEL							
Command = SWITCH_MULTILEVEL_SUPPORTED_GET							

3.46.2 Multilevel Switch Supported Report Command

The Multilevel Switch Supported Report Command is sent as a response to the Multilevel Switch Supported Get command. This report command contains information about the behavior of the device with regards to the Up/Down and Inc/Dec parameters included in the Multilevel Switch Start Level Change command. This report must not be sent unsolicited.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_MULTILEVEL							
Command = SWITCH_MULTILEVEL_SUPPORTED_REPORT							
Reserved			Primary Switch Type				
Reserved			Secondary Switch Type				

Reserved (3 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Primary / Secondary Switch Type (5 bit)

Note! The Primary Switch Type is addressed by Multilevel Switch Set, Get and Report as well as Start/Stop Level Change (Up/Down) commands. In case the Primary Switch Type is set to 0x00 (Undefined / Not supported) the application must respond to Multilevel Switch Get with Multilevel Switch Report (Value = 0xFE) for unknown state/position.

Note! The Secondary Switch Type is addressed only through Multilevel Switch Start/Stop Level Change (Inc/Dec).

For both Switch Types the value describes the movement profile of the device. Refer to the table below with respect to defined Switch Types. New types/values can be requested from Zensys.

Switch Type Value	0x00 (Direction/Endpoint A)	0x63/0xFF (Direction/Endpoint B)
0x00	Undefined / Not supported	
0x01	Off	On
0x02	Down	Up
0x03	Close	Open
0x04	Counter-Clockwise	Clockwise
0x05	Left	Right
0x06	Reverse	Forward
0x07	Pull	Push
0x08-0x1F	Reserved	

3.46.3 Multilevel Switch Start Level Change Command

The Multilevel Switch Start Level Change Command can be used to inform a multilevel switch, that it should start changing the level. The speed that the switch increases or decreases the level with is implementation specific. In Multilevel Switch Command Class (Version 3) a Step Size field has been added to this command to support general increment or decrement function typically available in a motor controlled device featuring two-dimensional movements.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_MULTILEVEL							
Command = SWITCH_MULTILEVEL_START_LEVEL_CHANGE							
Up/Down		Ignore Start Level	Inc/Dec		Reserved		
Start Level							
Dimming Duration							
Step Size							

Up/Down (2 bit)

If the Up/Down field is set to 0 the switch should increase the level. If field is set to 1 the switch should decrease the level. If the field is set to 3 there is no change to the Up/Down level.

Value	Description
0x00	Up
0x01	Down
0x02	Reserved
0x03	No Up/Down motion

Ignore Start Level (1 bit)

This field is unchanged from version 2.

Note! Refer to the device class specification for correct implementation of this field.

Reserved (3 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Start Level (8 bit)

This field is unchanged from version 2.

Note! Refer to the device class specification for correct implementation of this field.

Dimming Duration (8 bit)

This field is unchanged from version 2.

Note! Refer to the device class specification for correct implementation of this field.

Inc/Dec (2 bit)

This field instructs the device to Increment or Decrement the value specified in the Step Size field.

Value	Description
0x00	Increment
0x01	Decrement
0x02	Reserved
0x03	No Inc/Dec

Step Size (8 bit)

The “Step Size” field indicates the percentage of steps an increment or decrement function should execute. “Step Size” can take values from 0 to 99 (0x00 – 0x63) and 255 (0xFF). All values must be implemented, but the actual implementation of the values 0x00…0x63 is a mapping to percentages where the value 0x63 represents 100%. The mapping of the value received should be monotonous i.e. a higher value in a start level change command will always result in either a higher or the same setting. If a device implements less than distinct 100 values in its internal setting, then the mapping of the active internal setting should be spread equally over the entire range of 0x00…0x63. The value 255 (0xFF) indicates a fixed step size defined by the OEM.

If the Increment/Decrement field is set to 3, the Step Size field must be set to 0.

3.47 Multilevel Toggle Switch Command Class, version 1

Do not use this command class for new devices.

Use instead Multilevel Switch Generic Device Class for such devices.

This section contains Commands that can be used to make a multilevel toggle switch. These Commands allow applications to set and get the level of a multilevel toggle switch.

3.47.1 Multilevel Toggle Switch Set Command

The Multilevel Toggle Switch Set Command can be used to set the level in a device that supports the multilevel switch functionality.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_TOGGLE_MULTILEVEL							
Command = SWITCH_TOGGLE_MULTILEVEL_SET							

3.47.2 Multilevel Toggle Switch Get Command

The Multilevel Toggle Switch Get Command can be used to request the state of the load controlled by the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_TOGGLE_MULTILEVEL							
Command = SWITCH_TOGGLE_MULTILEVEL_GET							

3.47.3 Multilevel Toggle Switch Report Command

The Multilevel Toggle Switch Report Command can be sent unsolicited or requested by the Multilevel Toggle Switch Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_TOGGLE_MULTILEVEL							
Command = SWITCH_TOGGLE_MULTILEVEL_REPORT							
Value							

Value (8 bit)

The value can be either 0x00 (off/disable) or 0xFF (on/enable). Furthermore it can take values from 1 to 99 (0x01 – 0x63).

3.47.4 Multilevel Toggle Switch Start Level Change Command

The Multilevel Toggle Switch Start Level Change Command can be used to inform a multilevel toggle switch, that it should start changing the level. The speed that the switch increases or decreases the level with is implementation specific.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_TOGGLE_MULTILEVEL							
Command = SWITCH_TOGGLE_MULTILEVEL_START_LEVEL_CHANGE							
Roll Over	Reser-ved	Ignore Start Level	Reserved				
Start Level							

Roll Over (1 bit)

If the roll over bit is set to 0 then the switch should stop when reaching the max or min level. If the roll over bit is set to 1 then the switch should continually increase and decrease the level until otherwise instructed.

Ignore Start Level (1 bit)

If the Ignore Start Level bit is set to 0 the switch should use the start level specified in the Command. If field is set to 1 the switch should start from the actual level in the device.

Reserved (1 + 5 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Start Level (8 bit)

The start level field contains the initial level that the switch should assume when it start to change the level.

3.47.5 Multilevel Toggle Switch Stop Level Change Command

The Multilevel Toggle Switch Stop Level Change Command can be used to inform a multilevel toggle switch, that it should stop changing the level.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SWITCH_TOGGLE_MULTILEVEL							
Command = SWITCH_TOGGLE_MULTILEVEL_STOP_LEVEL_CHANGE							

3.48 No Operation Command Class, version 1

The No Operation Command Class is used to check if a node is reachable by sending a Command less frame to the specified destination. This feature is used by the Z-Wave protocol in many situations e.g. checking that an excluded node is non-responding. This Command can also be used on application level e.g. checking if a SUC/SIS is reachable from a new node in the network. This command class contains no command identifier and data.

Notice: It is not necessary to announce the No Operation Command Class in the NIF.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_NO_OPERATION							

3.49 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 all nodes in a Z-Wave network. The text strings must be stored in non-volatile memory by the application in the actual devices and can be requested by any other node in the network.

Notice: Please be aware that routing slaves based on future chip series can have limitations on non-volatile memory. It might then only be possible to base a design with the Node Naming and Location Command Class on libraries using an external EEPROM.

3.49.1 Node Name Set Command

The Node Name Set Command is used to set the name of a 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 2							
…							
Node name char x							

Node name char 1-x (variable)

Node name using specified character representation. The Node name can have a maximum of 16 characters. The number of character fields transmitted can be determined from the frame length. If a frame with more than 16 characters is received only the first 16 characters must be accept. The remaining characters must be ignored.

Reserved (5 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Char. Presentation (3 bit)

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

Char. Presentation	Description
0	Using standard ASCII codes, see Appendix B (values 128-255 are ignored)
1	Using standard and OEM Extended ASCII codes, see Appendix B
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. E.g. 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.

This list may evolve in the future. Undefined values of the character presentation identifier must be ignored.

3.49.2 Node Name Get Command

The Node Name Get Command is used to request the stored name from a node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_NODE_NAMING							
Command = NODE_NAMING_NODE_NAME_GET							

3.49.3 Node Name Report Command

The Node Name Report returns the stored node name from a node requested by the Node Name Get Command.

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 2							
…							
Node name char x							

Node name char 1-x (variable)

Node name using specified character representation. The number of characters transmitted can be determined from the length field in the frame.

Reserved (5 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Char. Presentation (3 bit)

Refer to the description under the Node Name Set Command.

3.49.4 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 2							
…							
Node location char x							

Node location char 1-x (variable)

Node location using specified character representation. The Node location can have a maximum of 16 characters. The number of character fields transmitted can be determined from the frame length. If a frame with more than 16 characters is received only the first 16 characters must be accept. The remaining characters must be ignored.

Reserved (5 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Char. Presentation (3 bit)

Refer to the description under the Node Name Set Command.

3.49.5 Node Location Get Command

The Node Location Command is used to request the stored node location from a node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_NODE_NAMING							
Command = NODE_NAMING_NODE_LOCATION_GET							

3.49.6 Node Location Report Command

The Node Location Report Command is used to report the stored node location from a node requested by the Node Location Get Command.

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 2							
…							
Node location char x							

Node location char 1-x (variable)

Node name using specified character representation. The number of characters transmitted can be determined from the length field in the frame.

Reserved (5 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Char. Presentation (3 bit)

Refer to the description under the Node Name Set Command.

3.50 Powerlevel Command Class, version 1

The Powerlevel Command Class defines RF transmit power controlling Commands useful when installing or testing a network. The Commands makes it possible for supporting controllers to set/get the RF transmit power level of a node and test specific links between nodes with a specific RF transmit power level.

NOTE: This Command Class is only used in an installation or test situation.

3.50.1 Powerlevel Set Command

The Powerlevel Set Command is used to set the power level indicator value, which should be used by the node when transmitting RF, and the timeout for this power level indicator value before returning the power level defined by the application.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_POWERLEVEL							
Command = POWERLEVEL_SET							
Power level							
Timeout							

Power level (8 bit)

The power level indicator value to set.

Valid levels are:

- NormalPower
- minus1dBm
- minus2dBm
- minus3dBm
- minus4dBm
- minus5dBm
- minus6dBm
- minus7dBm
- minus8dBm
- minus9dBm

Timeout value is ignored if Power level is set to normalPower. The node must, when receiving this Command, call the ZW_SET_POWERLEVEL API function to effectuate the Command.

Timeout (8 bit)

The time in seconds the node should keep the Power level before resetting to normalPower level. It is fundamental, that the timeout IS implemented and followed by the application, for keeping the network consistent. Valid values are 1-255 resulting in timeouts from 1 second to 255 seconds.

3.50.2 Powerlevel Get Command

The Powerlevel Get Command is used to request the current power level indicator value in use by the node. The node receiving this Command should answer with a Powerlevel Report.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_POWERLEVEL							
Command = POWERLEVEL_GET							

3.50.3 Powerlevel Report Command

The Powerlevel Report Command is used to report the current power level indicator value when transmitting and the timeout for this power level indicator value before returning the power level defined by the application. The Powerlevel Report Command can be sent unsolicited or requested by the Powerlevel Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_POWERLEVEL							
Command = POWERLEVEL_REPORT							
Power level							
Timeout							

Power level (8 bit)

This value is the current power level indicator value in effect on the node.

Valid levels are:

- NormalPower
- minus1dBm
- minus2dBm
- minus3dBm
- minus4dBm
- minus5dBm
- minus6dBm
- minus7dBm
- minus8dBm
- minus9dBm

If the returned value is normalPower, the timeout value is ignored. The node must call the ZW_GET_POWERLEVEL API function to get the current power level indicator value.

Timeout (8 bit)

The time in seconds the node has back at Power level before resetting to normalPower level.

3.50.4 Powerlevel Test Node Set Command

The Powerlevel Test Node Set Command is used to instruct the destination node to transmit a number of test frames to the specified nodeID with the RF power level specified. After the test frame transmissions the RF power level is reset to normal and the result (number of acknowledged test frames) must be saved. The result of the test can be acquired with a Powerlevel Test Node Get Command, which results in a Powerlevel Test Node Report Command being transmitted.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_POWERLEVEL							
Command = POWERLEVEL_TEST_NODE_SET							
Test nodeID							
Power level							
Test frame count (MSB)							
Test frame count (LSB)							

Test nodeID (8 bit)

The test nodeID that should receive the transmitted test frames. The node must, when receiving this Command:

- Call the ZW_RF_POWERLEVEL_SET API function to set Power level.

- Call the ZW_SEND_TEST_FRAME API function to transmit the test frame with the Power level to Test nodeID this it must do the specified number of times.

- Finally the node must call the ZW_RF_POWERLEVEL_SET API function to reset the power level back to normal to effectuate the Command.

Power level (8 bit)

The power level indicator value to use in the test frame transmission.

Valid levels are:

- NormalPower
- minus1dBm
- minus2dBm
- minus3dBm
- minus4dBm
- minus5dBm
- minus6dBm
- minus7dBm
- minus8dBm
- minus9dBm

Test frame count (16 bit)

The Test frame count field contains the number of test frames to transmit to Test nodeID. The first byte is the most significant byte. Valid Test frame count range is 1-65535.

3.50.5 Powerlevel Test Node Get Command

The Powerlevel Test Node Get Command is used to request for the result of the latest Powerlevel Test Node Set Command effectuated. The node receiving this Command should answer with a Powerlevel Test Node Report.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_POWERLEVEL							
Command = POWERLEVEL_TEST_NODE_GET							

3.50.6 Powerlevel Test Node Report Command

The Powerlevel Test Node Report Command is used to report the latest result of a test frame transmission started by the Powerlevel Test Node Set Command. The test report can be send either as a response to a Powerlevel Test Node Get Command or unsolicited, for example when a requested test run is done the test report can be send to the originating node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_POWERLEVEL							
Command = POWERLEVEL_TEST_NODE_REPORT							
Test NodeID							
Status of operation							
Test frame acknowledged count (MSB)							
Test frame acknowledged count (LSB)							

Test NodeID (8bit)

This field contains the NodeID on the node, which is or has been under test. Test NodeID contains the nodeID set in the latest Powerlevel Test Node Set Command. If Test NodeID is ZW_TEST_NOT_A_NODEID then no test has been made and the Status of operation and the Test frame acknowledged count fields can be ignored.

Status of operation (8 bit)

This field contains the result of latest Powerlevel Test Node Set Command. Valid values are:

ZW_TEST_SUCCES

ZW_TEST_FAILED

ZW_TEST_INPROGRESS

ZW_TEST_IN_PROGRESS is returned if a test is still in progress. ZW_TEST_SUCCES is returned if at least 1 test frame transmission has been acknowledged (TRANSMIT_COMPLETE_OK) else ZW_TEST_FAILED (no test frame transmissions has been acknowledged) is returned.

Test frame acknowledged count (16 bit)

Number of test frames transmitted, which the Test NodeID has acknowledged. The first byte is the most significant byte.

3.51 Proprietary Command Class, version 1

The Proprietary Command Class is used to transfer data between devices. The data content must be vendor specific and must be non-value added with respect to the Home Automation application in general.

Note: The Proprietary Command Class must not be used without written approval from Zensys.

3.51.1 Proprietary Set Command

The Proprietary Set Command is used to transfer data to a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROPRIETARY							
Command = PROPRIETARY_SET							
Data 1							
…							
Data N							

Data 1 .. Data N (variable)

The data fields can be used to set various data in the device. Each data field is 8 bit and the maximum number of data fields in one single cast or broadcast frame is 54 bytes for a non-secure Z-Wave solution. The number of data fields transmitted can be determined from the length field in the frame.

3.51.2 Proprietary Get Command

The Proprietary Get Command is used to request data from a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROPRIETARY							
Command = PROPRIETARY_GET							
Data 1							
…							
Data N							

Data 1 ... Data N (variable)

Refer to explanation under the Proprietary Set Command.

3.51.3 Proprietary Report Command

The Proprietary Report is used to retrieve various data from a device. The Proprietary Report Command can be sent unsolicited or requested by the Proprietary Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROPRIETARY							
Command = PROPRIETARY_REPORT							
Data 1							
…							
Data N							

Data 1 ... Data N (variable)

Refer to explanation under the Proprietary Set Command.

3.52 Protection Command Class, version 1

The Protection Command Class version 1 is used to protect a device against unintentionally control by e.g. a child.

3.52.1 Protection Set Command

The Protection Set Command is used to set the protection state in a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROTECTION							
PROTECTION_SET							
Protection State							

Protection State (8 bit)

The protection state field is used to set the protection state of the device.

Protection State	Description
0x00	Unprotected - The device is not protected, and can be operated normally via the user interface.
0x01	Protection by sequence - The device is protected by altering the way the device normally is operated into a more complicated sequence of actions, e.g. if a device normally is controlled by a single press of a button on the device it might be changed to require 3 rapid presses on a button to control it.
0x02	No operation possible - It is not possible at all to control a device directly via the user interface.

Control via Z-Wave is always possible independent of protection state.

3.52.2 Protection Get Command

The Protection Get Command is used to request the protection state from a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROTECTION							
Command = PROTECTION_GET							

3.52.3 Protection Report Command

The Protection Report Command is used to report the protection state of a device. The Protection Report Command can be sent unsolicited or requested by the Protection Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROTECTION							
Command = PROTECTION_REPORT							
Protection State							

Protection State (8 bit)

Refer to explanation under Protection Get Command.

3.53 Protection Command Class, version 2

The Protection Command Class version 2 is extended to specify whether a device can be controlled via RF Commands or not. When a video recorder is powered by an outlet that can be controlled by RF the user would like to prevent the video recorder from being turned of when it is programmed to record her/his favorite show. In this case the Protection Command Class version 2 can be used to protect the outlet from being turned off by setting the outlet in “No RF Control” state.

The following Commands have been added or changed in version 2. The Commands not mentioned here will remain the same**.**

Notice: This command class is suggested for convenience applications. For liability reasons the command class is not recommended for safety applications.

3.53.1 Protection Set Command

The Protection Set Command is used to set the protection state in a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROTECTION							
Command = PROTECTION_SET							
Reserved				Local Protection State			
Reserved				RF Protection State			

Local Protection State (4 bit)

The protection state field is used to set the protection state of the device.

Local Protection State	Description
0	Unprotected - The device is not protected, and can be operated normally via the user interface.
1	Protection by sequence - The device is protected by altering the way the device normally is operated into a more complicated sequence of actions, e.g. if a device normally is controlled by a single press of a button on the device it might be changed to require 3 rapid presses on a button to control it.
2	No operation possible - It is not possible at all to control a device directly via the user interface.

Note! – Local Protection can only protect a device from “normal operation”. This means only the operation that is intended by the application of the device. It is NOT allowed to protect the device from network functionalities. The device cannot be protected from being put into learn mode nor from sending out the NIF.

RF Protection State (4 bit)

The RF protection state field is used to set the RF protection state of the device. In the case where a device set into a RF Protection State which instructs the device not to answer to a “normal operation” Command, the device must return with the Application Rejected Request Command (Status = 0) from the Application Status Command Class. Please refer to 3.5 for more details about the Application Status Command Class.

RF Protection State	Description
0	Unprotected - The device must accept and respond to all RF Commands.
1	No RF control - all runtime Commands are ignored by the device. The device must still respond with status on requests.
2	No RF response at all. The device will not even reply to status requests.

Note! – It is only possible to un-protect the device with the Protection Set Command. It is not allowed ignore Protection Commands. If a device is excluded from the network, the protection states must be reset.

3.53.2 Protection Report Command

The Protection Report Command is used to report the protection state of a device. The Protection Report Command can be sent unsolicited or requested by the Protection Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROTECTION							
Command = PROTECTION_REPORT							
Reserved				Local Protection State			
Reserved				RF Protection State			

3.53.3 Protection Supported Get Command

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROTECTION							
Command = PROTECTION_SUPPORTED_GET							

3.53.4 Protection Supported Report Command

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROTECTION							
Command = PROTECTION_SUPPORTED_REPORT							
Reserved						Exclusive Control	Timeout
Local Protection State Byte 1							
Local Protection State Byte 2							
RF Protection State Byte 1							
RF Protection State Byte 2							

Local Protection State Byte 1 .. 2

The list of all Local Protection States can be found in section 3.53.1. The two bytes must be interpreted as bit masks where byte 1 bit 0 represent Local Protection State 0, byte 1 bit 1 represent Protection State 1, byte 2 bit 0 represent Protection State 8 etc.

RF Protection State Byte 1 .. 2

The list of all RF Protection States can be found in section 3.53.1. The two bytes must be interpreted as bit masks where byte 1 bit 0 represent RF Protection State 0, byte 1 bit 1 represent Protection State 1, byte 2 bit 0 represent Protection State 8 etc.

Exclusive Control

When this bit is set to 1 the device support Exclusive Control. When Exclusive Control is supported the device must support the Commands Protection Exclusive Control Set, Get and Report described below.

Timeout

When this bit is set to 1 the device supports a timeout for RF Protection State. When the timeout is supported the device must support the Commands Protection Timeout Set, Get and Report described below.

3.53.5 Protection Exclusive Control

The Protection Exclusive Control is an optional feature. The Commands in this chapter can only be implemented if the device supporting Protection Command Class version 2 announces support for Exclusive Control in the Protection Supported Report Command.

3.53.5.1 Protection Exclusive Control Set Command

The Protection Exclusive Control Set Command is used to set the node ID of a Z-Wave device that can override the protection state in a device that is protected.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROTECTION							
Command = PROTECTION_EC_SET							
Node ID							

Node ID

All “normal operation” RF Commands received from this node ID are accepted and executed. “normal operation” RF Commands from any other node ID’s in the network are ignored and as reply the Application Rejected Request Command must be send.

Factory default setting of the Node ID for exclusive control must be set to 0. To reset the exclusive control state in a device an Exclusive Control Set Command with Node ID 0 as parameter must be send to the device.

3.53.5.2 Protection Exclusive Control Get Command

The Protection Exclusive Control Get Command is used to request a Protection Exclusive Control Report Command from the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROTECTION							
Command = PROTECTION_EC_GET							

3.53.5.3 Protection Exclusive Control Report Command

The Protection Exclusive Control Report Command is used to return the node ID of a Z-Wave device that has exclusive control over this device in protection mode.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROTECTION							
Command = PROTECTION_EC_REPORT							
Node ID							

Node ID

See description under the Protection Exclusive Control Set Command section 3.53.5.1.

3.53.6 Protection Timeout

The Protection Timeout is an optional feature. The Commands in this chapter can only be implemented if the device supporting Protection Command Class version 2 announces support for Timeout in the Protection Supported Report Command.

3.53.6.1 Protection Timeout Set Command

The Protection Timeout Set Command is used to set the timeout for protection mode in a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROTECTION							
Command = PROTECTION_TIMEOUT_SET							
Timeout							

Timeout

The timeout describe the time a device will remain in RF Protection mode.

Factory default setting for the Timeout parameter must be 0x00.

Timeout	Description
0x00	No timer is set. All “normal operation” Commands must be accepted.
0x01-0x3C	Timeout is set from 1 second (0x01) to 60 seconds (0x3C) in 1-second resolution.
0x41-0xFE	Timeout is set from 2 minutes (0x41) to 191 minutes (0xFE) in 1-minute resolution.
0xFF	No Timeout – The Device will remain in RF Protection mode infinitely.

3.53.6.2 Protection Timeout Get Command

The Protection Timeout Command is used to request a Protection Timeout Report Command from the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROTECTION							
Command = PROTECTION_TIMEOUT_GET							

3.53.6.3 Protection Timeout Report Command

The Protection Timeout Report Command is used to return the remaining time that a device will remain in protection mode.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_PROTECTION							
Command = PROTECTION_TIMEOUT_REPORT							
Timeout							

Timeout

Timeout	Description
0x00	No timer is set. All "normal operation" Commands must be accepted.
0x01-0x3C	If the remaining time for protection mode is 1 minute or less the remaining time will be returned in 1-second resolution from 1 second (0x01) to 60 seconds (0x3C).
0x41-0xFE	If the remaining time for protection mode is more than 1 minute the remaining time will be returned in 1-minute resolution from 2 minutes (0x41) to 191 minutes (0xFE).
0xFF	No Timeout is set – The Device will remain in RF Protection mode infinitely.

3.54 Pulse Meter Command Class, version 1

The Pulse Meter Command Class defines the Commands necessary to implement the pulse meter functionality. The Pulse Meter Command Class is intended for all kinds of meters that generate pulses, such as gas and water meters.

3.54.1 Pulse Meter Get Command

The Pulse Meter Get Command is used to request the number of pulses that has been counted.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_METER_PULSE							
Command = METER_PULSE_GET							

3.54.2 Pulse Meter Report Command

The Pulse Meter Report Command is used to report the number of pulses detected in the device. The Pulse Meter Report Command can be sent unsolicited or requested by the Pulse Meter Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_METER_PULSE							
Command = METER_PULSE_REPORT							
Pulse Count 1							
Pulse Count 2							
Pulse Count 3							
Pulse Count 4							

Pulse Count (32 bit)

The Pulse Count field contains the number of pulses generated by the meter. The first byte is the most significant byte.

3.55 Remote Association Activation Command Class, version 1

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.

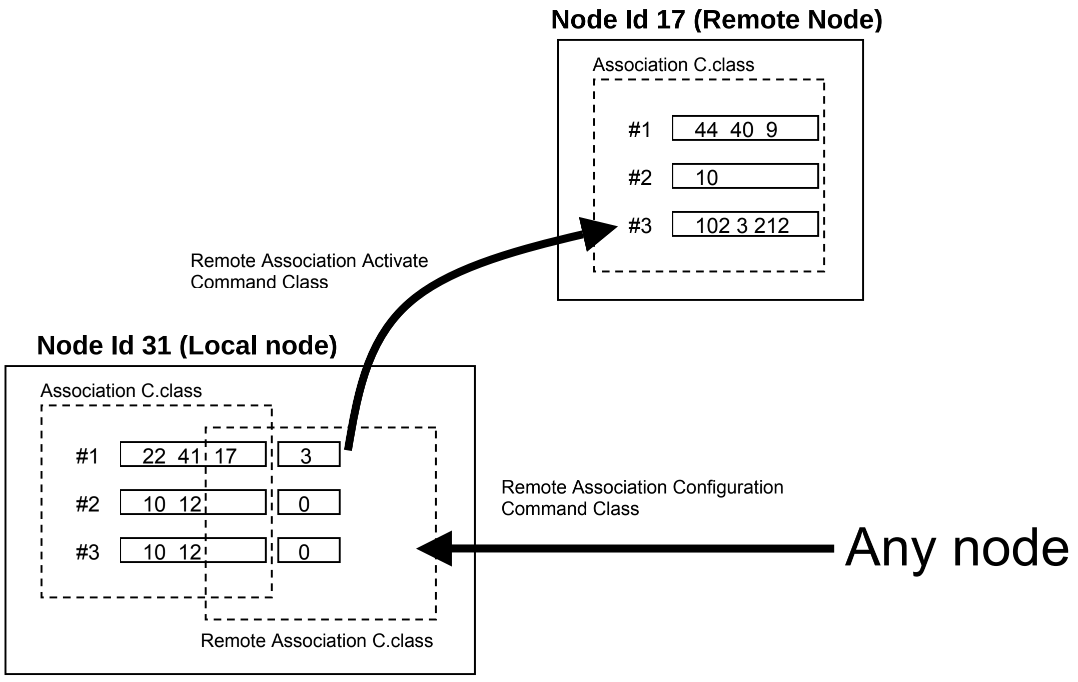


Figure 8, Remote Association Activation Command Class

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 instruct the 3rd node to activate one of its locally stored association group identifiers as defined by the Association Command Class.

3.55.1 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 bit)

This group identifier is used to specify the grouping identifier on the remote node

3.56 Remote Association Configuration Command Class, version 1

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.

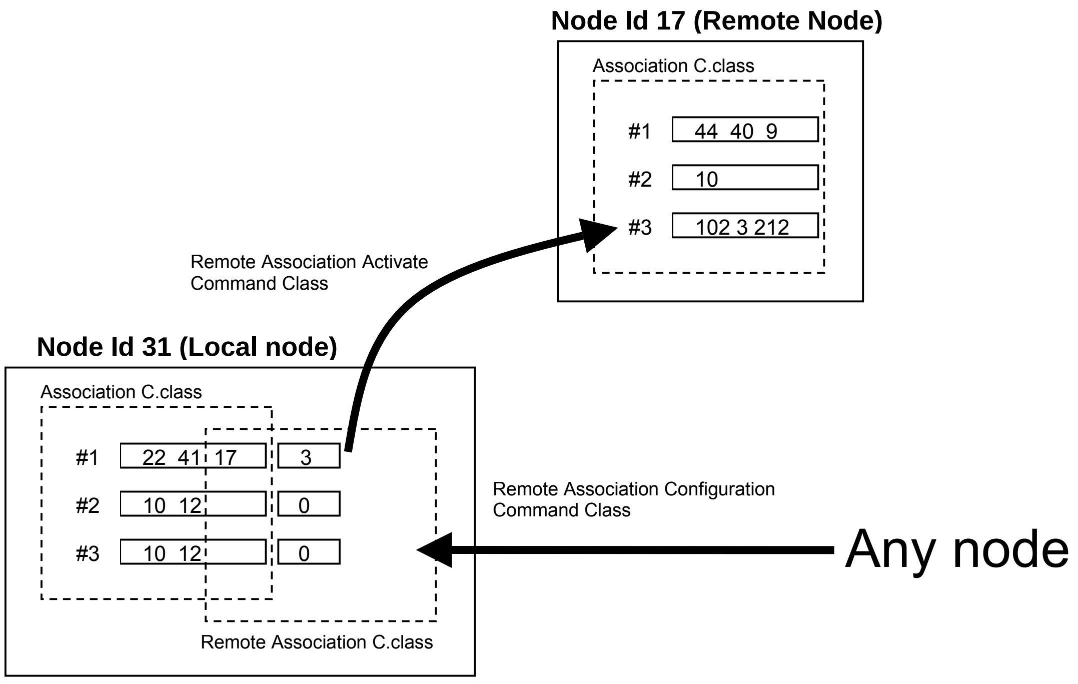


Figure 9, Remote Association Configuration Command Class

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.

3.56.1 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 bit)

This group identifier is as explained in the Association Command Class 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 bit)

This NodeId is 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 bit)

This group identifier is used to specify the grouping identifier on the remote node.

3.56.2 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.

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 bit)

This group identifier is as explained in the Association Command Class used to specify the grouping identifier on the local node.

3.56.3 Remote Association Configuration Report Command

The Remote Association Configuration Report Command returns the remote node ID and the grouping identifier. The Remote Association Configuration Report can be send requested by the Remote Association Configuration Get Command.

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 bit)

This group identifier is used to specify the grouping identifier on the local node

Remote NodeId (8 bit)

This NodeId is 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 bit)

This Remote grouping identifier is used to specify the grouping identifier on the remote node that should be activated.

3.57 Scene Activation Command Class, version 1

The Scene Activation Command Class is used for the actual scene launching in a number of devices e.g. a another scene-controlling unit, in a multilevel switch, in a binary switch etc. This command class requires an initial configuration of the scenes to be launched by the Scene Actuator Configuration Set or Scene Controller Configuration Set Command depending on device used.

The advantage of this approach is that since it is a common identifier that is sent out, the multicast frame type can be used, which may eliminate potential popping effect which could be the result if individual set-level Commands were send out to a large number of nodes distributed over a vast area. The multicast must still be followed by individual singlecasts to ensure all the nodes addressed got the message.

3.57.1 Scene Activation Set Command

The Scene Activation Set Command is used to activate the setting associated to the scene ID. The Scene Activation Set Command is sent as a multicast to assure all nodes within direct range responds immediately. After the multicast follows a sequence of single casts to each device to ensure all devices received the scene ID.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SCENE_ACTIVATION							
Command = SCENE_ACTIVATION_SET							
Scene ID							
Dimming Duration							

Scene ID (8 bit)

Scene ID (1…255) to be activated in the device.

Dimming Duration (8 bit)

The Dimming Duration can either use a pre-configured value, it can be instantly, or it can be a duration that is communicated as part of the Scene Activation Set Command. Only the Multilevel Scene Switch specific device classes interpret this field. The table below shows how to obtain the wanted functionality:

Dimming Duration	Description
0x00	Instantly
0x01-0x7F	Obtain dimming durations from 1 second (0x01) to 127 seconds (0x7F) in 1-second resolution
0x80-0xFE	Specify dimming durations from 1 minute (0x80) to 127 minutes (0xFE) in 1-minute resolution.
0xFF	Specify dimming duration configured by the Scene Actuator Configuration Set and Scene Controller Configuration Set Command depending on device used.

3.58 Scene Actuator Configuration Command Class, version 1

The Scene Actuator Configuration Command Class is used to configure scenes in a scene device e.g. a multilevel scene switch, binary scene switch etc. A scene device must always support 255 scene IDs.

3.58.1 Scene Actuator Configuration Set Command

The Scene Actuator Configuration Set Command is used to associate the specified scene ID to the defined settings.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SCENE_ACTUATOR_CONF							
Command = SCENE_ACTUATOR_CONF_SET							
Scene ID							
Dimming Duration							
Over-ride	Reserved						
Level							

Scene ID (8 bit)

Scene ID (1...255) to be associated with the current settings.

Dimming Duration (8 bit)

Dimming Duration specify how long time it must take to reach the wanted level associated to the Scene ID. Dimming always start from current level. So the dimming duration specified is the same independent of the number of levels to be changed. Only the Multilevel Scene Switch specific device classes interpret this field. The table below shows how to obtain the wanted functionality:

Dimming Duration	Description
0x00	Specify Instantly
0x01-0x7F	Specify dimming durations from 1 second (0x01) to 127 seconds (0x7F) in 1-second resolution
0x80-0xFE	Specify dimming durations from 1 minute (0x80) to 127 minutes (0xFE) in 1-minute resolution.
0xFF	Specify factory default dimming duration.

Override (1 bit)

If the Override bit is set to 0 then the current settings in the device is associated with the Scene ID. If the Override bit is set to 1 then the Level value in the Command is associated to the Scene ID.

Reserved (7 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Level (8 bit)

The value specified must correspond to the format the device uses when receiving Basic Set Commands.

3.58.2 Scene Actuator Configuration Get Command

The Scene Actuator Configuration Get Command is used to request the settings for a given scene identifier or for the currently active scene settings.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SCENE_ACTUATOR_CONF							
Command = SCENE_ACTUATOR_CONF_GET							
Scene ID							

Scene ID (8 bit)

Scene ID (1...255) to request. If scene ID is equal to 0 then current active scene is requested.

3.58.3 Scene Actuator Configuration Report Command

The Scene Actuator Configuration Report Command is used to report the locally stored configuration for a given scene identifier in the device requested by the Scene Actuator Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SCENE_ACTUATOR_CONF							
Command = SCENE_ACTUATOR_CONF_REPORT							
Scene ID							
Level							
Dimming Duration							

Scene ID (8 bit)

The scene ID (1…255) indicates scene settings being returned. If scene ID is equal to 0 it indicate that no scene is currently active in the device.

Level (8 bit)

The value reported by the device must correspond to the format the device uses to respond to Basic Get Commands.

Dimming Duration (8 bit)

Dimming Duration specify how long time it must take to reach the wanted level associated to the Scene ID. Only the Multilevel Scene Switch specific device classes interpret this field. The table below shows how the different dimming durations are reported:

Dimming Duration	Description
0x00	Instantly
0x01-0x7F	Dimming durations from 1 second (0x01) to 127 seconds (0x7F) in 1-second resolution
0x80-0xFE	Specify dimming durations from 1 minute (0x80) to 127 minutes (0xFE) in 1-minute resolution.

3.59 Scene Controller Configuration Command Class, version 1

The Scene Controller Configuration Command Class is used to configure scenes controlled from a scene controlling device by some kind of physical activation. A scene device must always support 255 scene IDs.

3.59.1 Scene Controller Configuration Set Command

The Scene Controller Configuration Set Command is used to configure settings for a given physical item on the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SCENE_CONTROLLER_CONF							
Command = SCENE_CONTROLLER_CONF_SET							
Group ID							
Scene ID							
Dimming Duration							

Group ID (8 bit)

The grouping identifier is mapped into a physical item e.g. a push button on the device in question. The grouping identifier values must be a sequence starting from 1. The Association Supported Groupings Get Command can be used to request the number of groupings that the device supports.

Scene ID (8 bit)

Scene ID (1...255) to be associated with the grouping identifier. To disable an associated scene for the specified group ID set scene ID equal to 0.

Dimming Duration (8 bit)

Dimming Duration specify how long time it must take to reach the wanted level associated to the Scene ID. Dimming always start from current level. So the dimming duration specified is the same independent of the number of levels to be changed. Only the Multilevel Scene Switch specific device classes interpret this field. The table below shows how to obtain the wanted functionality:

Dimming Duration	Description
0x00	Specify Instantly
0x01-0x7F	Specify dimming durations from 1 second (0x01) to 127 seconds (0x7F) in 1-second resolution
0x80-0xFE	Specify dimming durations from 1 minute (0x80) to 127 minutes (0xFE) in 1-minute resolution.
0xFF	Specify factory default dimming duration.

3.59.2 Scene Controller Configuration Get Command

The Scene Controller Configuration Get Command is used to request the settings for a given grouping identifier or the active settings.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SCENE_CONTROLLER_CONF							
Command = SCENE_CONTROLLER_CONF_GET							
Group ID							

Group ID (8 bit)

Group ID field indicates what grouping identifier the get Command is referring to. The grouping identifier values must be a sequence starting from 1. A grouping identifier equal to 0 requests the currently active group and scene ID. The grouping identifier is mapped into a physical item e.g. a push button on the device in question.

3.59.3 Scene Controller Configuration Report Command

The Scene Controller Configuration Report Command is used to report the current settings in the device requested by the Scene Controller Configuration Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SCENE_CONTROLLER_CONF							
Command = SCENE_CONTROLLER_CONF_REPORT							
Group ID							
Scene ID							
Dimming Duration							

Group ID (8 bit)

The requested or active grouping identifier.

Scene ID (8 bit)

Scene ID (1...255) setting for the specified grouping identifier. In case the scene ID is disabled then 0 is returned.

Dimming Duration (8 bit)

The Dimming Duration to be used when the specified Scene ID is launched with the Scene Activation Command Class. Dimming Duration specify how long time it must take to reach the wanted level associated to the Scene ID. Only the Multilevel Scene Switch specific device classes interpret this field. The table below shows how the different dimming durations are reported:

Dimming Duration	Description
0x00	Instantly
0x01-0x7F	Dimming durations from 1 second (0x01) to 127 seconds (0x7F) in 1-second resolution
0x80-0xFE	Specify dimming durations from 1 minute (0x80) to 127 minutes (0xFE) in 1-minute resolution.

3.60 Screen Attributes Command Class, version 1

This Screen Attribute Command Class is used to retrieve screen attributes from the device hosting the screen. This allows another device to send data formatted according to the screen attributes to the device hosting the screen. The screen can be located on any device in the Z-Wave network.

3.60.1 Screen Attributes Get Command

The Screen Attributes Get Command is used to request the screen attributes.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SCREEN_ATTRIBUTES							
Command = SCREEN_ATTRIBUTES_GET							

3.60.2 Screen Attributes Report Command

The Screen Attributes Report Command can be sent unsolicited or requested by the Screen Attributes Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SCREEN_ATTRIBUTES							
Command = SCREEN_ATTRIBUTES_REPORT							
Reserved			Number of Lines				
Number of Characters per Line							
Size of Line Buffer							
Numerical Presentation of a Character							

Reserved (3 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Number of Lines (5 bit)

Number of lines the screen supports (1..16).

Number of Characters per Lines (8 bit)

Number of characters the screen supports on each line (1..255).

Size of Line Buffer (8 bit)

Number of characters the line buffer supports for each line (1..255). Size of line buffer will always be equal or larger than the number of visual characters per line. The text will typically scroll in case it is larger than the number of visual characters.

Numerical Presentation of a Character (8 bit)

The screen supports the following numerical presentations of a character:

Bit Map	Description
Bit 0	Supports ASCII codes if the bit is 1 and the opposite if 0. See Appendix B (values 128-255 are ignored)
Bit 1	Supports ASCII codes and Extended ASCII codes if the bit is 1 and the opposite if 0. See Appendix B
Bit 2	Supports Unicode UTF-16 if the bit is 1 and the opposite if 0.
Bit 3	Supports ASCII codes and Player codes, see Appendix B (undefined values are ignored)

This list may evolve in the future.

3.61 Screen Attributes Command Class, version 2

This Screen Attribute Command Class is enhanced with a parameter specifying Screen Timeout.

The Commands not mentioned here will remain the same as in version 1**.**

3.61.1 Screen Attributes Report Command

The Screen Attributes Report Command can be sent unsolicited or requested by the Screen Attributes Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SCREEN_ATTRIBUTES							
Command = SCREEN_ATTRIBUTES_REPORT							
Reserved		Escape Sequence	Number of Lines				
Number of Characters per Line							
Size of Line Buffer							
Numerical Presentation of a Character							
Screen Timeout							

Reserved (2 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Escape Sequence(1 bit)

If set to 0 escape sequences are not supported by the device. If set to true then escape sequences are supported by the device.

Number of Lines (5 bit)

Number of lines the screen supports (1..16).

Number of Characters per Lines (8 bit)

Number of characters the screen supports on each line (1..255).

Size of Line Buffer (8 bit)

Number of characters the line buffer supports for each line (1..255). Size of line buffer will always be equal or larger than the number of visual characters per line. The text will typically scroll in case it is larger than the number of visual characters.

Numerical Presentation of a Character (8 bit)

The screen supports the following numerical presentations of a character:

Bit Map	Description
Bit 0	Supports ASCII codes if the bit is 1 and the opposite if 0. See Appendix B (values 128-255 are ignored)
Bit 1	Supports ASCII codes and Extended ASCII codes if the bit is 1 and the opposite if 0. See Appendix B
Bit 2	Supports Unicode UTF-16 if the bit is 1 and the opposite if 0.
Bit 3	Supports ASCII codes and Player codes, see Appendix B (undefined values are ignored)

This list may evolve in the future.

Screen Timeout (8 bit)

If Screen Timeout is set to 0 display is always on. If set to larger than 0, defines the display timeout in seconds.

3.62 Screen Meta Data Command Class, version 1

The Screen Meta Data Command Class used to streaming data containing user related information to a screen located on a device in a Z-Wave network. The screen can request single or multiple data packets. The device having the data containing user related information to the screen can also initiate the data streaming.

The API call ZW_SendDataMeta must be used when streaming data to ensure that this traffic don't prevent control data from getting through in the network, especially important for 9.6kbps nodes because they can't detect 40kbps RF communication. Refer to [2] regarding a detailed description of the API call ZW_SendDataMeta.

3.62.1 Screen Meta Data Get Command

The Screen Meta Data Get Command is used to request the Screen Meta Data Report Command. The Screen Meta Data Get Command is used as handshake to avoid buffer overflow in the receiving device. The Screen Meta Data Get Command will optionally be able to request multiple Screen Meta Data Report Commands to improve the effective bandwidth.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SCREEN_MD							
Command = SCREEN_MD_GET							
Number of Reports							
Node ID							

Number of Reports (8 bit)

Number of Screen Meta Data Report Commands to be received without requesting each Screen Meta Data Report Command (1..255). Be aware of overflow when requesting multiple reports.

Node ID (8 bit)

The Node ID (1..232) specifies the device to receive the requested reports. In case node ID is equal to 0x00 then the information is requested by the source node ID of the Screen Meta Data Get Command.

3.62.2 Screen Meta Data Report Command

The Screen Meta Data Report Command is used to send data to the device hosting the screen. The Screen Meta Data Report Command can be sent unsolicited or requested by the Screen Meta Data Get Command. The size of the payload should not be bigger than 48 bytes because routing over 4 hops can be necessary to reach the destination. It's possible to write characters to multiple lines in the same frame.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SCREEN_MD							
Command = SCREEN_MD_REPORT							
More Data	Reser-ved	Screen Settings			Char. Presentation		
Line Settings A			Clear A	Line Number A			
Character Position A							
Number of Characters A							
Character 1,A							
…							
Character N,A							
…							
…							
Line Settings B			Clear B	Line Number B			
Character Position B							
Number of Characters B							
Character 1,B							
…							
Character N,B							

More Data (1 bit)

The more data bit indicates if additional reports are expected before the whole data streaming is completed. If the more data bit is set to 1 then additional reports are expected and the opposite if 0.

Reserved (1 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Screen Settings (3 bit)

The screen settings identifier can be set to the following values:

Screen Settings	Description
0	Whole screen is cleared before lines are written
1	Current content on screen is scrolled one line down
2	Current content on screen is scrolled one line up
7	Do not change the current content on the screen

This list may evolve in the future. Undefined values of the screen settings identifier must be ignored.

Char. Presentation (3 bit)

The character presentation identifier can be set to the following values:

Char. Presentation	Description
0	Using standard ASCII codes, see Appendix B (values 128-255 are ignored)
1	Using standard ASCII codes and OEM Extended ASCII codes, see Appendix B
2	Unicode UTF-16
3	Using standard ASCII codes and Player codes, see Appendix B (undefined values are ignored)

Note: Devices supporting Unicode UTF-16 characters are described by a 2 byte long decimal representation. The first byte is the most significant byte. E.g. 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.

This list may evolve in the future. Undefined values of the character presentation identifier must be ignored.

Line Settings (3 bit)

The line settings identifier can be set to the following values:

Line Settings	Description
0	Characters are written in selected font
1	Characters are written as highlighted
2	Characters are written using a larger font compared to line settings equal to 0
3	
4	
5	
6	
7	

Clear (1 bit)

Determine if the characters are written directly or line is cleared first.

Clear	Description
0	Characters are written directly
1	Line is cleared before characters are written

Line Number (4 bit)

The line number field indicates the line to write the characters to counting from zero (0..15).

Character Position (8 bit)

The character position field indicates where on the line to write the characters counting from zero (0..255). The character position can be larger than the display size in case the line buffer is bigger (See the Screen Attributes Report Command).

Number of Characters (8 bit)

The number of characters field indicates how many characters to be written on the screen for the specified line number, counting from 1.

Character 1 .. Character N (variable)

The character fields hold the string to output in specified character representation. Characters will be ignored in case there is no room left in the line buffer.

3.63 Screen Meta Data Command Class, version 2

The Screen Meta Data Command Class is enhanced with an extended settings bit that enabled an extra byte for settings. This version includes a setting for Screen Timeout which can be specified by the Screen Attribute Command Class version 2.

The Commands not mentioned here will remain the same as in version 1**.**

3.63.1 Screen Meta Data Report Command

The Screen Meta Data Report Command is used to transfer data to the device hosting the screen. The Screen Meta Data Report Command can be sent unsolicited or requested by the Screen Meta Data Get Command. The size of the payload should not be bigger than 48 bytes because routing over 4 hops can be necessary to reach the destination. It's possible to write characters to multiple lines in the same frame.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SCREEN_MD							
Command = SCREEN_MD_REPORT							
More Data	Ex-tended Setup	Screen Settings			Char. Presentation		
Line Settings A			Clear A	Line Number A			
Character Position A							
Number of Characters A							
Character 1,A							
…							
Character N,A							
…							
…							
Line Settings B			Clear B	Line Number B			
Character Position B							
Number of Characters B							
Character 1,B							
…							
Character N,B							
Reserved							Screen Timeout

More Data (1 bit)

The more data bit indicates if additional reports are expected before the whole data streaming is completed. If the more data bit is set to 1 then additional reports are expected and the opposite if 0.

Extended Setup (1 bit)

If set to true, the last byte of the payload defines an extended setup.

Screen Settings (3 bit)

The screen settings identifier can be set to the following values:

Screen Settings	Description
0	Whole screen is cleared before lines are written
1	Current content on screen is scrolled one line down
2	Current content on screen is scrolled one line up
7	Do not change the current content on the screen

This list may evolve in the future. Undefined values of the screen settings identifier must be ignored.

Char. Presentation (3 bit)

The character presentation identifier can be set to the following values:

Char. Presentation	Description
0	Using standard ASCII codes, see Appendix B (values 128-255 are ignored)
1	Using standard ASCII codes and OEM Extended ASCII codes, see Appendix B
2	Unicode UTF-16
3	Using standard ASCII codes and Player codes, see Appendix B (undefined values are ignored)

Note: Devices supporting Unicode UTF-16 characters are described by a 2 byte long decimal representation. The first byte is the most significant byte. E.g. 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.

This list may evolve in the future. Undefined values of the character presentation identifier must be ignored.

Line Settings (3 bit)

The line settings identifier can be set to the following values:

Line Settings	Description
0	Characters are written in selected font
1	Characters are written as highlighted
2	Characters are written using a larger font compared to line settings equal to 0
3	Characters are written using a larger font (font B) & highlighted
4	Characters are written in selected font (font A), no scroll
5	Characters are written in selected font (font A) & highlighted, no scroll
6	Characters are written using a larger font (font B), no scroll
7	Characters are written using a larger font (font B) & highlighted, no scroll

For values 0-3, text will be scrolled. For values 4-7 the text will not be scrolled, and will be truncated if it is longer than the width of the display.

Clear (1 bit)

Determine if the characters are written directly or line is cleared first.

Clear	Description
0	Characters are written directly
1	Line is cleared before characters are written

Line Number (4 bit)

The line number field indicates the line to write the characters to counting from zero (0..15).

Character Position (8 bit)

The character position field indicates where on the line to write the characters counting from zero (0..255). The character position can be larger than the display size in case the line buffer is bigger (See the Screen Attributes Report Command).

Number of Characters (8 bit)

The number of characters field indicates how many characters to be written on the screen for the specified line number, counting from 1.

Character 1 .. Character N (variable)

The character fields hold the string to output in specified character representation. Characters will be ignored in case there is no room left in the line buffer. If the Escape Sequence Bit is true in the SCREEN_ATTRIBUTES_REPORT Command, the device supports advanced display features by making escape sequences in the form of an Escape char followed by a char value 0-255.

Screen Timeout (1 bit)

If the screen timeout is set to 0 the devices preset timeout should be used.

If set to 1 the device should keep the display powered. This does not affect on the RF.

Reserved (7 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

3.64 Security Command Class, version 1

The Security Command Class and an Application Security layer specification [4] create the foundation for secure application communication between nodes in a Z-Wave network. The security layer provides confidentiality, authentication and replay attack robustness through AES-128.

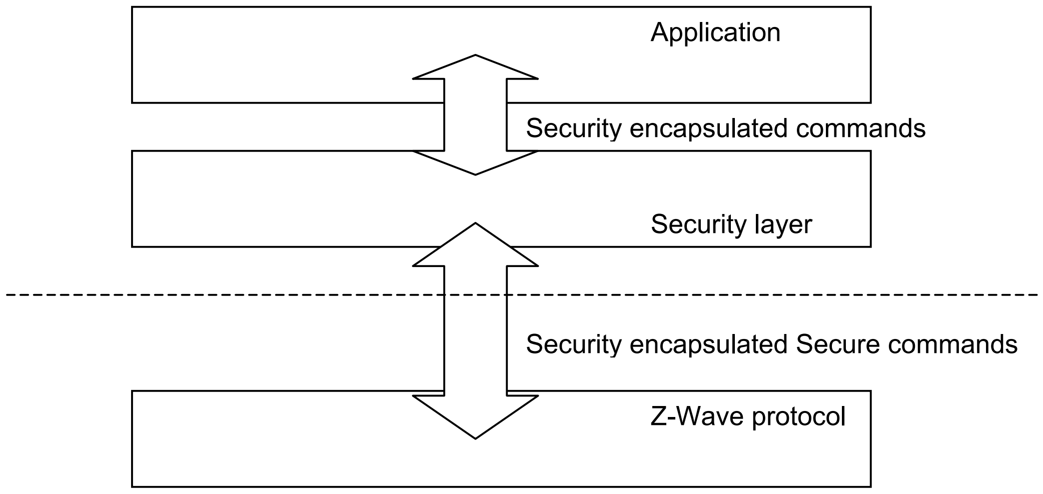

Application
Security encapsulated commands
Security layer
Security encapsulated Secure commands
Z-Wave protocol

Figure 10, Protocol layers extended with security solution

The Security Command Class defines a number of commands used to facilitate handling of encrypted frames in a Z-Wave Network. The commands deal with three main areas:

- Message Encapsulation. The task of taking a plain text frame and encapsulating the frame into an encrypted Security Message.
- Command Class Handling. The task of handling what command classes are supported when communicating with a Security enabled device
- Network Key Management. The task of initial key distribution.

3.64.1 Message Encapsulation and Command Class Handling

For encapsulating messages, Z-Wave requires four commands. Before sending an encrypted frame, the sender must acquire a nonce (number used once) from the recipient. The sender then uses this number along with the locally generated nonce along with the network key to generate the Security Message Encapsulation Command as illustrated below.

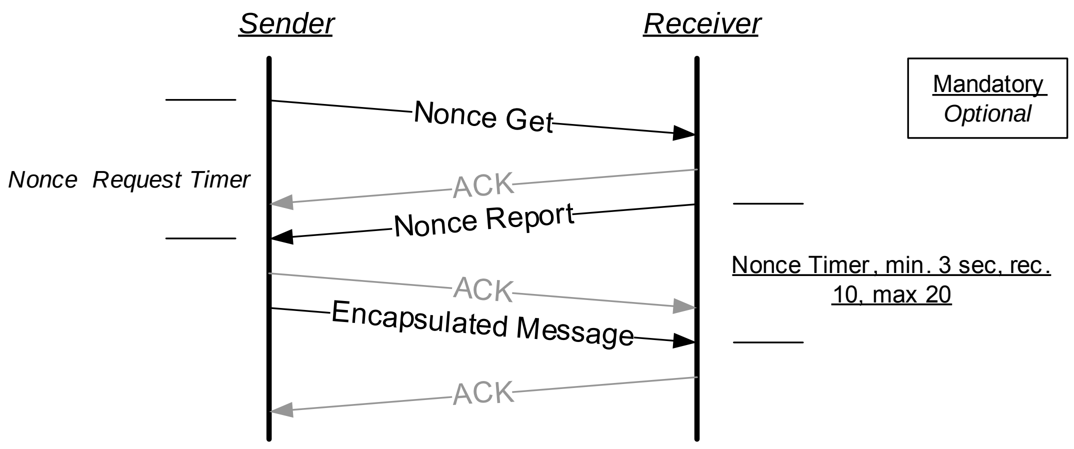


Figure 11, Sending secure messages

This mechanism generates an overhead of three commands for each single frame that is sent encrypted (plus an acknowledge frame).

A number of timers must be implemented to defend against attacks.

First, an optional but recommended timer should be started when Nonce Get has been sent, the Nonce Report must then be received before this timer runs out. The length of this timer will depend on the application it is trying to protect.

The second timer is mandatory and must be activated after the Nonce Report has been sent. The Encapsulated Message must be received within the specified timeout in order to be accepted.

Note that all timers must be started when the frame has been sent, not when Acknowledge has been received, since an attacker could just delay the Acknowledge frame.

The Nonce Timers must be used in all communication that uses the mentioned commands.

In order to optimize the performance the device must use streaming when transmitting multiple frames. The overhead using this option will then converge towards two (instead of three) as the number of frames increases.

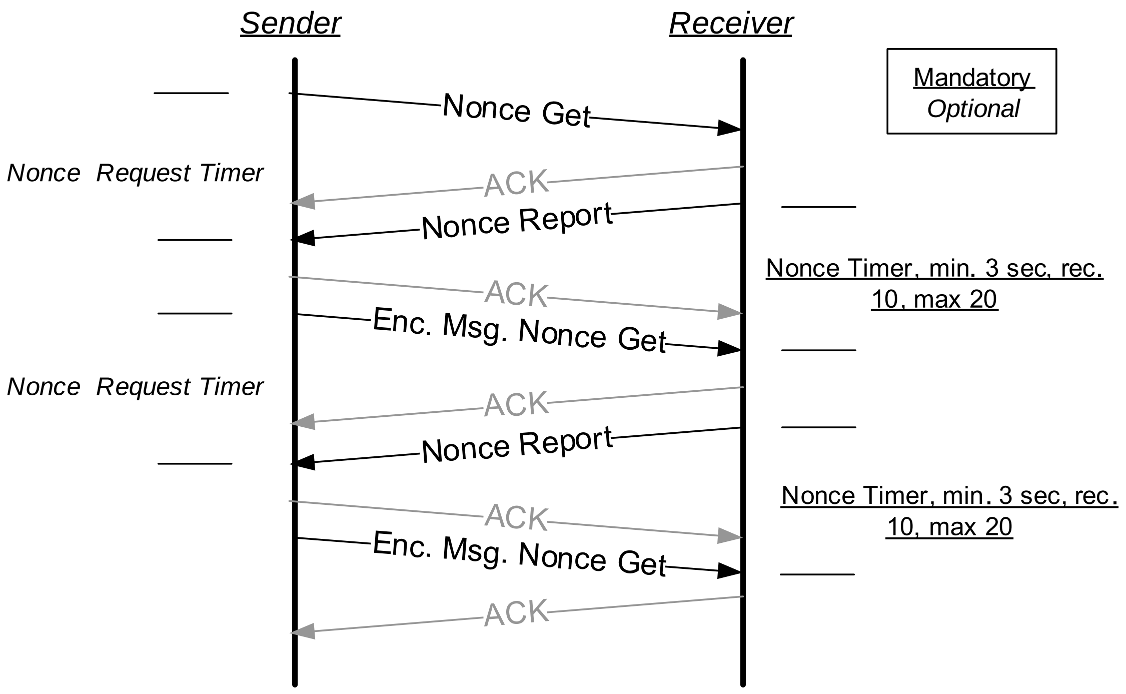


Figure 12, Streaming secure messages

Notice: Due to the security overhead, the maximum command size becomes 28 bytes instead of the usual 48 bytes. Larger commands can use sequencing as described in 3.64.1.3.

3.64.1.1 Nonce Challenge Request Command

The Device uses Security Nonce Get Command to request an external nonce from the receiving node. For a description of the algorithm for generating a Z-Wave Security Nonce, see [4]. Note that a nonce will only be valid for one attempt. Nonce is thrown away when the receiver has used it for decrypting the message and a new nonce must be exchanged.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Security Header = SECURITY_NONCE_GET							

3.64.1.2 Nonce Challenge Response Command

The device uses the Security Nonce Report Command to return the next nonce to the requesting node at the receipt of a Security Nonce Get command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Security Header = SECURITY_NONCE_REPORT							
Nonce byte 1							
Nonce byte 2							
Nonce byte 3							
Nonce byte 4							
Nonce byte 5							
Nonce byte 6							
Nonce byte 7							
Nonce byte 8							

Nonce byte 1..8 (8 Bytes)

This field contains the 8 bytes external nonce used for encryption.

3.64.1.3 Security Message Encapsulation Command

The device uses the Security Message Encapsulation command to encapsulate Z-Wave commands using AES-128.

The device will also request a new external nonce from the receiver when transmitting the message Security Message Encapsulation Nonce Get. The device uses the external nonce when streaming multiple secure messages without having to call Nonce Get after receiving each message as shown in This mechanism generates an overhead of three commands for each single frame that is sent encrypted (plus an acknowledge frame).

A number of timers must be implemented to defend against attacks.

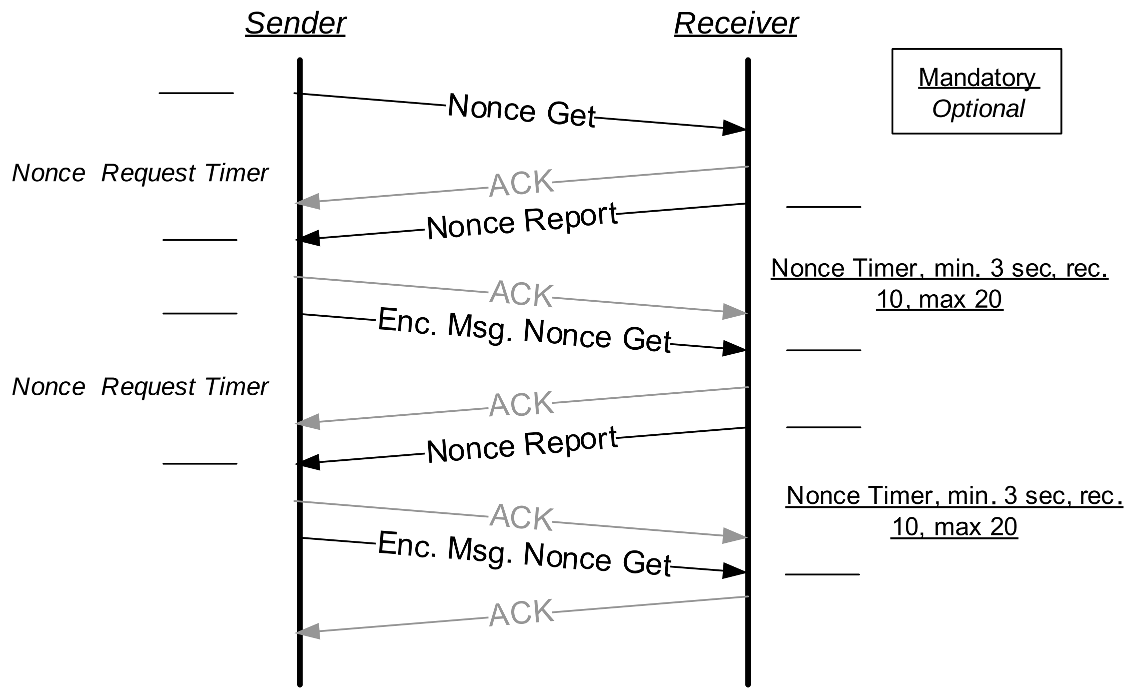
First, an optional but recommended timer should be started when Nonce Get has been sent, the Nonce Report must then be received before this timer runs out. The length of this timer will depend on the application it is trying to protect.

The second timer is mandatory and must be activated after the Nonce Report has been sent. The Encapsulated Message must be received within the specified timeout in order to be accepted.

Note that all timers must be started when the frame has been sent, not when Acknowledge has been received, since an attacker could just delay the Acknowledge frame.

The Nonce Timers must be used in all communication that uses the mentioned commands.

In order to optimize the performance the device must use streaming when transmitting multiple frames. The overhead using this option will then converge towards two (instead of three) as the number of frames increases.



.

As illustrated in For encapsulating messages, Z-Wave requires four commands. Before sending an encrypted frame, the sender must acquire a nonce (number used once) from the recipient. The sender then uses this number along with the locally generated nonce along with the network key to generate the Security Message Encapsulation Command as illustrated below.

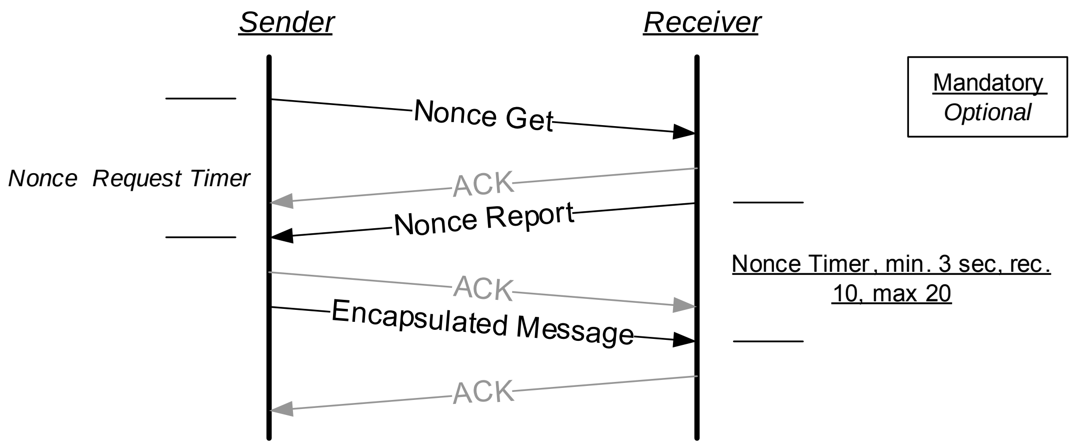


Figure 11, Sending secure messages and This mechanism generates an overhead of three commands for each single frame that is sent encrypted (plus an acknowledge frame).

A number of timers must be implemented to defend against attacks.

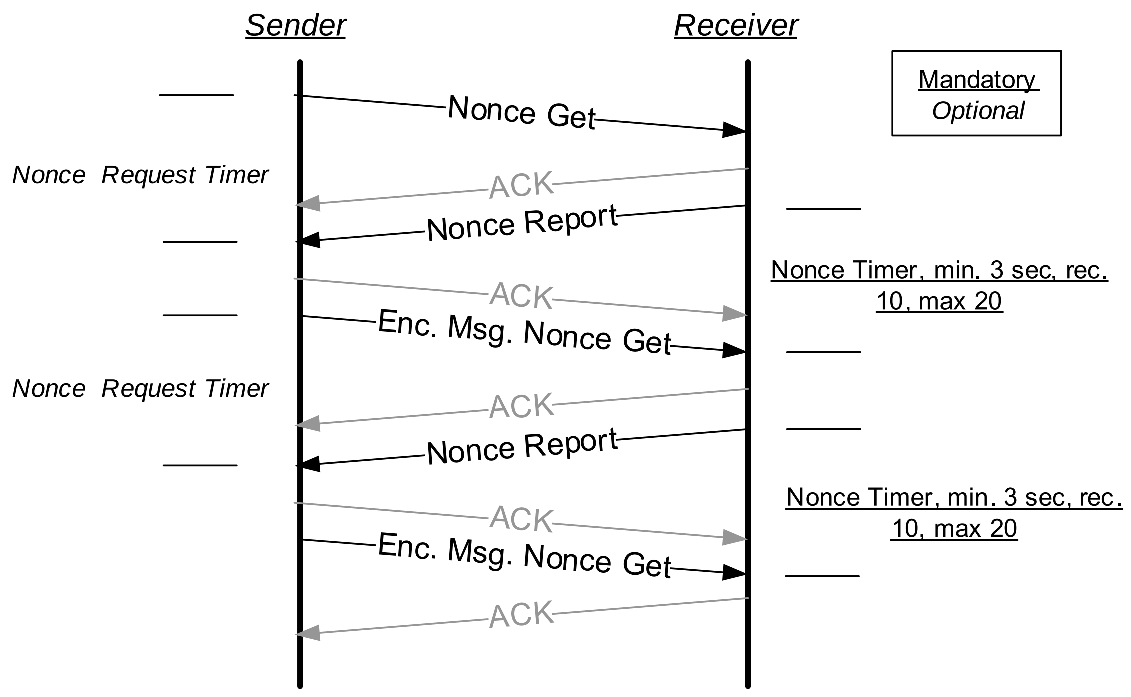
First, an optional but recommended timer should be started when Nonce Get has been sent, the Nonce Report must then be received before this timer runs out. The length of this timer will depend on the application it is trying to protect.

The second timer is mandatory and must be activated after the Nonce Report has been sent. The Encapsulated Message must be received within the specified timeout in order to be accepted.

Note that all timers must be started when the frame has been sent, not when Acknowledge has been received, since an attacker could just delay the Acknowledge frame.

The Nonce Timers must be used in all communication that uses the mentioned commands.

In order to optimize the performance the device must use streaming when transmitting multiple frames. The overhead using this option will then converge towards two (instead of three) as the number of frames increases.



, the device must receive the Security Message Encapsulation command within 3 seconds after the creation of the nonce. The device must start a 3 seconds timer at creation of nonce to keep track of the validity of the nonce.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Security Header = SECURITY_MESSAGE_ENCAPSULATION (_NONCE_GET)							
Initialization Vector byte 1							
Initialization Vector byte 2							
Initialization Vector byte 3							
Initialization Vector byte 4							
Initialization Vector byte 5							
Initialization Vector byte 6							
Initialization Vector byte 7							
Initialization Vector byte 8							
Reserved		Second Frame	Sequenced	Sequence Counter			
(Command Class identifier)							
(Command identifier)							
Command byte 1							
..							
Command byte n							
Receiver's nonce Identifier							
Message Authentication Code byte 1							
Message Authentication Code byte 2							
Message Authentication Code byte 3							
Message Authentication Code byte 4							
Message Authentication Code byte 5							
Message Authentication Code byte 6							
Message Authentication Code byte 7							
Message Authentication Code byte 8							

Initialization Vector byte 1..8 (8 byte)

The initialization vector is the internal nonce generated by the sender. The payload is encrypted with the external and internal nonce concatenated together. See [4].

Reserved (2 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of the fields nor perform processing based on their content.

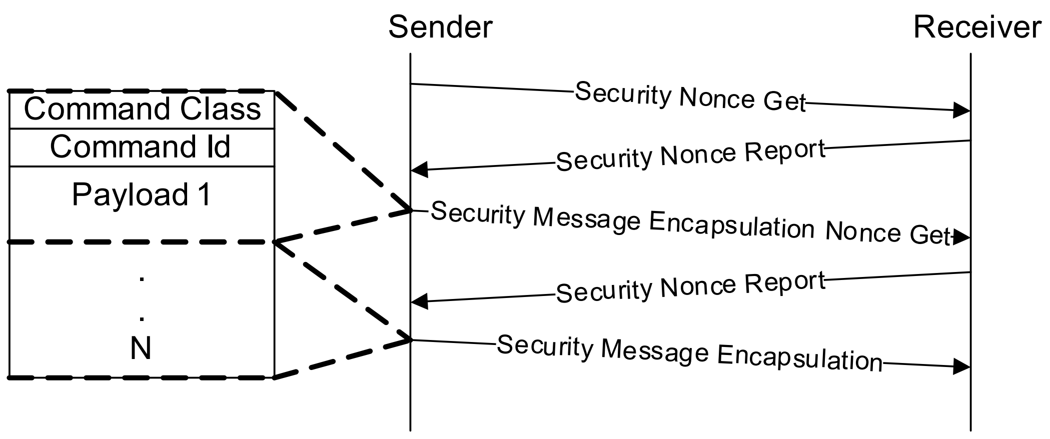


Figure 13, Frame flow for sequenced frames

Sequenced(1 bit)

If this flag is set, the device transmits the command using multiple frames due to payload space shortage. After the receiver has received all the frames, it will reassemble the command. If this flag is not set, the command is contained entirely in this frame. As shown in **Error! Reference source not found.**, the first frame in a sequence must be sent using Security Message Encapsulation Nonce Get to minimize overhead the second can be sent using Security Message Encapsulation if there are no following frames to send.

Notice that device only lists Command class identifier and command identifier in the first frame.

Second Frame (1 bit)

If this flag and the Sequenced flag are set, the frame is the second out of two. If the flag is not set, and Sequenced flag is set, it is the first frame out of two. Valid combinations are:

	Sequenced 1	**Sequenced 0**
Second Frame 1	Second frame of two	-
Second Frame 0	First frame of two	Single Frame

Sequence Counter (4 bit)

If Sequenced flag is set, the frame is one out of two. In order to tell multiple sequences apart, they must be uniquely identified based on the sender node ID and the Sequence Counter. For each sequenced set of frames a node sends it must increment the Sequence Counter by one.

Command Class Identifier (8 bit) (Part of Encrypted Payload)

This field contains the identifier of the Command class, which the device sends to the NodeID.

Command identifier (8 bit) (Part of Encrypted Payload)

This field contains the identifier of the Command, which the device sends to the NodeID.

Command byte1 . Command byte n (Part of Encrypted Payload)

These fields contain the parameters, which the device sends to the NodeID.

Receiver's nonce Identifier (8 bit)

Identifies nonce being used. See [4].

Message Authentication Code byte 1..8 (8 byte)

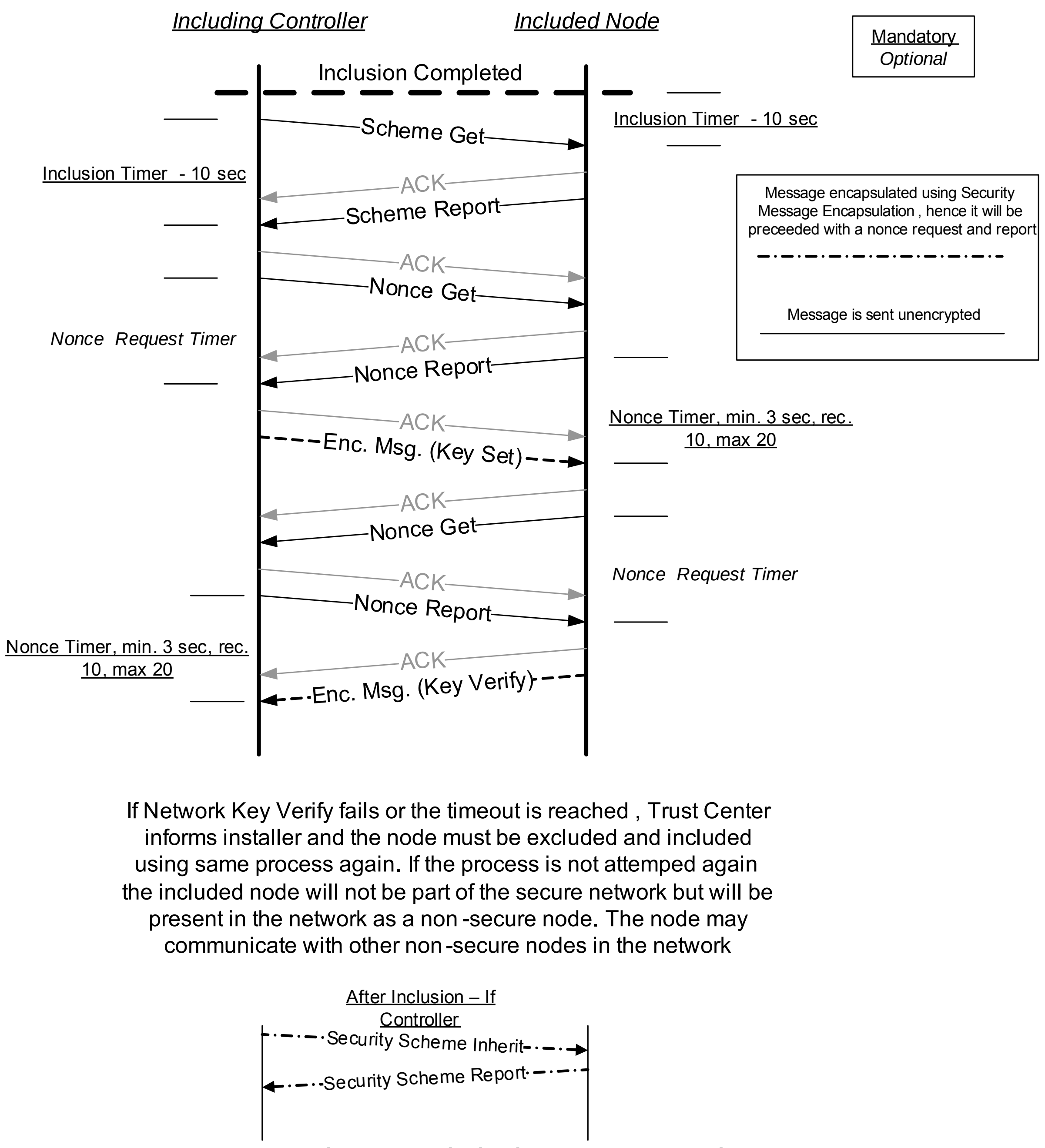
Data used for authenticating the received message to prevent tampering. See [4].

3.64.2 Network Key Management

Distribution of network keys uses a temporary key to protect the key exchange. Exchange of network key happens immediately after successful inclusion of the node. It requires a secure primary/inclusion controller to include a secure node into the secure network as secure.

3.64.2.1 Network Inclusion

The first step of including a node to a secure network is using the standard Z-Wave inclusion process. If both the new node and the inclusion controller support Security command class, the controller will subsequently send the network key to the newly included node.



Including Controller
Included Node
Mandatory
Optional
Inclusion Completed
Inclusion Timer - 10 sec
Scheme Get
Inclusion Timer - 10 sec
ACK
Scheme Report
Message encapsulated using Security Message Encapsulation , hence it will be preceeded with a nonce request and report
Message is sent unencrypted
ACK
Nonce Get
Nonce Request Timer
ACK
Nonce Report
ACK
Nonce Timer, min. 3 sec, rec. 10, max 20
Enc. Msg. (Key Set)
ACK
Nonce Get
ACK
Nonce Request Timer
Nonce Report
Nonce Timer, min. 3 sec, rec. 10, max 20
ACK
Enc. Msg. (Key Verify)
If Network Key Verify fails or the timeout is reached , Trust Center informs installer and the node must be excluded and included using same process again. If the process is not attemped again the included node will not be part of the secure network but will be present in the network as a non -secure node. The node may communicate with other non-secure nodes in the network
After Inclusion – If Controller
Security Scheme Inherit
Security Scheme Report

Figure 14, Inclusion into a secure network

To protect the security of a secure network it is recommended that all controllers should require a PIN to unlock the security inclusion process and slaves should require a PIN to accept being included and excluded.

Following the inclusion of the node into the network, the controller will request the security scheme supported by the included node. Battery operated devices should stay awake for the duration of the setup of the Security Command class.

Currently one security scheme exist which is extendable at a later stage:

1. **Security 0/N:** 0x00 repeated 16 times as temporary key for encrypting the network key when it is transferred using normal power.

The validity of the key is verified in both the added node and the including controller. The node verifies the key based on the Message Authentication Code and then transmits an encrypted Network Key Verify command as response to the controller. When a device supporting the Security Command class does not manage to enter the secure network, it will function as a non-secure device. The node requires exclusion from the network before another attempt comprising of inclusion and network key exchange is possible.

For the including controller to allow inclusion of a Secure device into the secure network, a common security scheme must be supported by both devices. When supporting multiple common schemes the highest possible scheme is used. If no common schemes are supported the device cannot be included into the network.

When nodes in the secure network wish to establish a connection to a device that supports the Security Command class, they must send the Security Command Supported Get command to the device. Receiving no Security Command Supported Report (since the recipient does not have the key to decrypt the request), it will not be able to talk to the device securely. The same applies for the situation where a secure device does not become part of the secure network because it was included by a non-secure controller.

3.64.2.1.1 Inclusion Timers

As shown in Figure 14, a number of timeout must be complied with. For the including controller see Figure 15.

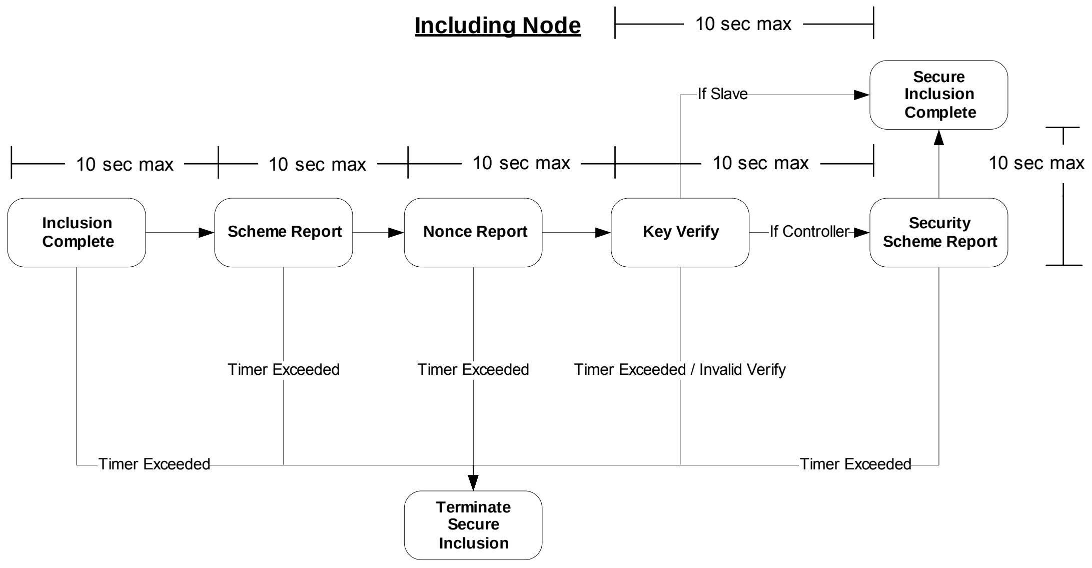


Figure 15, Timers on Including Controller

For the new included node, the timers in Figure 16 must be complied with.

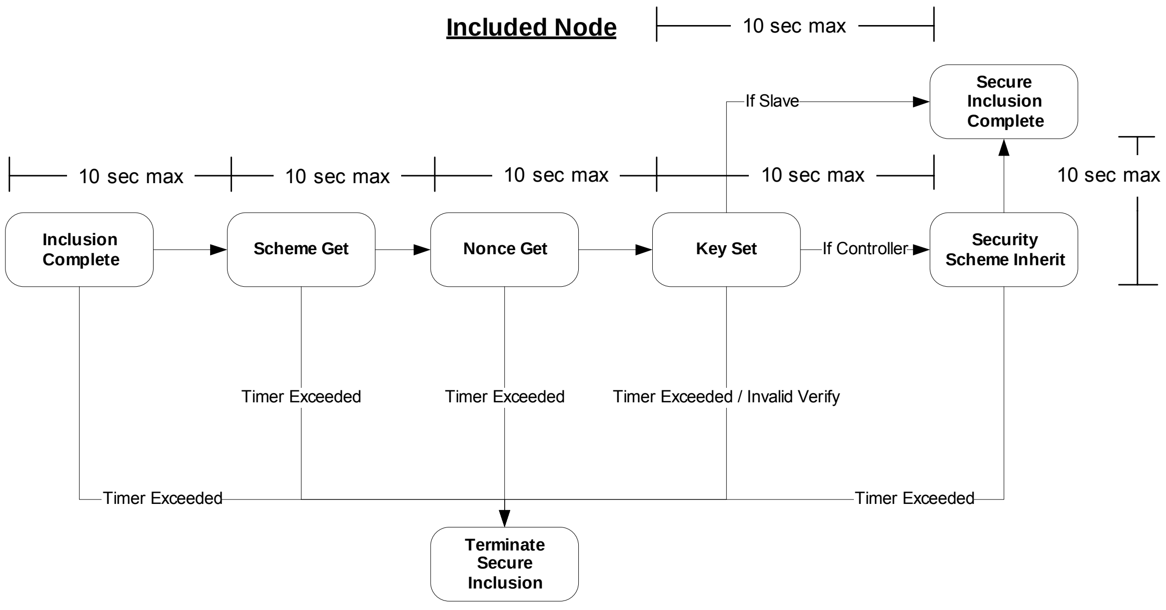


Figure 16, Timers on newly Included Node

The Network Key must only be sent to the new node if a Security Scheme Report command has been received by the including controller within 10 seconds after successful inclusion of the node. The controller should notify the user of an error condition in case of timeout because the device functions only as non-secure. In addition, the included node must only accept and respond to a Scheme Get it is received within 10 seconds of inclusion. When the required frame is received within the timeout, the timeout is extended to allow the next part of the inclusion process, if that part is not reached within 10 seconds inclusion process must terminate.

3.64.2.2 Security Scheme Get Command

The device must send Security Scheme Get Command immediately after the successful inclusion of a node with support for the Security Command class. The Security Scheme Get will request a report specifying the security scheme supported by the new node, and report the controllers own supported security scheme to the new node. The new node must then select the highest common security scheme for entering the secure network as described in the previous section.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Command = SECURITY_SCHEME_GET							
Supported Security Schemes							

Supported Security Schemes (8bit)

The Security Schemes which are supported by the primary/inclusion controller. At least one security scheme must be supported. Possible values are:

Bit	Supports
0	Security 0 using normal power = 0
1	Reserved = 0
2	Reserved = 0
3	Reserved = 0
4	Reserved = 0
5	Reserved = 0
6	Reserved = 0
7	Reserved = 0

The reserved bit fields are for future use. The implementation shall zero these bit fields and shall make no assumptions on the values of the fields nor perform processing based on their content.

3.64.2.3 Security Scheme Report Command

The device must send Security Scheme Report Command as response to a Security Scheme Get command. The Security Scheme Report will inform the controller of the security scheme the node supports. If the only common scheme is Security 0, the device must send network key immediately without waiting for input, by using 16 times 0x00 as the temporary key. If no common scheme exists, the controller must notify the installer that the node cannot be included to the secure network, but will continue as a non-secure node in the network.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Command = SECURITY_SCHEME_REPORT							
Supported Security Schemes							

Supported Security Schemes (8bit)

See Security Scheme Get for a definition.

3.64.2.4 Network Key Set Command

The Device can use the Network Key Set Command to set the network key in a Z-Wave node. Transmission of the Network Key Set command requires existence of a common agreed security scheme. The device uses the agreed temporary key to encapsulate the Network Key Set command. The included node may only accept the Network Key Set command under the guidelines describes in section 3.64.2.

This command must only be send encapsulated by the Security Message Encapsulation command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Command = NETWORK_KEY_SET							
Network Key byte 1							
..							
Network Key byte N							

Network Key byte1 .. Network Key byte N

The Network key to exchange application data secure in the network.

3.64.2.5 Network Key Verify Command

When the included node has received a Network Key Set that is has successfully decrypted, verified by the MAC, it must send a Network Key Verify Command to the including controller. If the controller is capable of decrypting the Network Key Verify command it would indicate that the included node has successfully entered the secure network. Since there is no timeout for the Network Key Verify, the controller can send a Security Commands Supported Get command, and if no response is received, it can be concluded that it has not been included properly.

This command must only be send encapsulated by the Security Message Encapsulation command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Command = NETWORK_KEY_VERIFY							

3.64.2.6 Security Scheme Inherit Command

When a controller is included to the network, it must inherit the same security scheme as the including controller allowing it to become an inclusion controller. This is achieved through the Security Scheme Inherit command, which is sent when the network key has successfully been setup, as shown in Figure 14.

When including a controller into the secure network, the new controller must inherit any common supported security schemes. For example, if the new controller supports security scheme bit 1 and bit 4 but the including controller only supports security scheme bit 1, the new controller must after inclusion also only support security scheme bit 1.

This command must only be send encapsulated by the Security Message Encapsulation command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Command = SECURITY_SCHEME_INHERIT							
Supported Security Schemes							

Supported Security Schemes (8bit)

See Security Scheme Get command, for a definition.

To ensure that the included controller has inherited the correct security scheme, it must respond with a Security Scheme Report command as illustrated in Figure 14. If the reported security scheme does not match, the installer must be notified that the included controller is violating the security scheme, and the node should be excluded again as an error situation has occurred.

3.64.3 Encapsulated Command Class Handling

Since the Node Info Frame must only be used to communicate all the command classes that are supported non-secure, command classes supported security encapsulated must be reported by using the Security Commands Supported Get/Report.

To make a security enabled device compatible with non-secure applications a secure node may choose to report support for some command classes non-secure in the Node Info Frame, as well as in the Security Command Supported Report. Initially the node info frame must contain all command classes that can be supported/controlled non-secure.
If the node is included into a secure network, it may then choose to remove all or some command classes from the node info frame, and thus only support them securely – removing support for the command classes for all non-secure nodes. It should be noted that the rules apply for Supported and Controlled command classes.
If the node is included into a non-secure network, it may choose to support command classes it would not support if it had been included into a secure network.

An example of this could be a relay / switch.

	Before Inclusion	**Included Secure**	**Included Non-Secure**
Security Command Supported Report Frame	-	Binary Switch	-
Node Info Frame	Security Binary Switch Version	Security Version	Security Binary Switch Version

It is up to the implementation of each application to decide which commands should be supported using security encapsulation and non-secure.

If a command class is only supported using security encapsulation it must not be listed in the node info frame, but must instead be listed in the security commands supported report frame. Additionally, the node information frame must contain the security command class.

An exception to this rule is the Basic Command Class, which must be listed in the Node Info Frame if it is supported unsecure, and if not listed implied Support Secure without being listed in the secure list.

Precautions should be taken when setting up the network, since the order of inclusion will be able to change the supported functionality of the devices. For example if the above relay was included by a non-secure controller it would be able to switch it on / off, but if included using a secure controller version request would be possible non-securely.

In a secure network, initially only the including controller will have any knowledge about what nodes in the network have been setup securely. If a node wishes to talk to another node it may send a Security Command Supported Get command encapsulated to the other node. If a Security Commands Supported Report is returned the node is in possession of a valid network key, and is part of the secure network. This mechanism may also be used by the including controller to ensure that the node has been included properly.

3.64.3.1 Security Commands Supported Get Command

The device uses Security Commands Supported Get Command to request which commands the device supports using Security Encapsulation. A node may choose only to support a command as ‘supported’ and/or ‘controlled’, when it is security encapsulated, in which case it must not be shown in the NIF, but it will be shown in the Security Commands Supported Report Command.

This command must only be send encapsulated by the Security Message Encapsulation command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Command = SECURITY_COMMANDS_SUPPORTED_GET							

3.64.3.2 Security Commands Supported Report Command

The device uses Security Commands Supported Report Command as a response to a Security Commands Supported Get command. The report informs the requesting node of which command classes is supported using security encapsulation. It is mandatory to report all command classes that the device supports and controls using the Security command class.

This command must only be send encapsulated by the Security Message Encapsulation command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Command = SECURITY_COMMANDS_SUPPORTED_REPORT							
Reports to follow							
Command Class (0x20 – 0xEE) 1 (support)							
…							
Command Class (0x20 – 0xEE) N (support)							
COMMAND_CLASS_MARK							
Command Class (0x20 – 0xEE) 1 (control)							
…							
Command Class (0x20 – 0xEE) K (control)							

To support extended command classes use the following format. Note that these can be mixed. Please refer to section 6 in [4].

This command must only be send encapsulated by the Security Message Encapsulation command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Command = SECURITY_COMMANDS_SUPPORTED_REPORT							
Reports to follow							
Command Class MSB (0xF1 – 0xFF) 1							
Command Class LSB (0x00 – 0xFF) 1							
…							
Command Class MSB (0xF1 – 0xFF) N							
Command Class LSB (0x00 – 0xFF) N							
COMMAND_CLASS_MARK							
Command Class MSB (0xF1 – 0xFF) 1							
Command Class LSB (0x00 – 0xFF) 1							
…							
Command Class MSB (0xF1 – 0xFF) K							
Command Class LSB (0x00 – 0xFF) K							

Reports to follow (8 bit)

This value indicates how many report frames there is left before all command classes have been transmitted.

Command Class 1 … N (8 bit / N * 8 bit)

The command class identifier as described in section 6 in [4]. 0x20 – 0xEE Application Command Classes, 0xF1 – 0xFF Extended Application Command Classes.

Command Class Mark (8 bit)

The COMMAND_CLASS_MARK is used to indicate that all preceding command classes are supported, and all following command classes are controlled.

3.65 Sensor Configuration Command Class, version 1

This Sensor Configuration Command Class adds the possibility for sensors to act on either a measured value or on a preconfigured value. With this command class an application can act on a specific event. It is up to the application to implement the actual event. This could e.g. be implementation of the Association Command Class where the application would activate a group based on a trigger from the sensor.

The trigger types that can be configured are the same types as the values specified in the Multilevel Sensor Command Class

Most movement sensors can be configured to "ignore" movement if it is not dark. Typically this is done mechanically. With the Sensor Configuration Command Class this can be configured via Z-Wave in an open command class (not the Configuration Command Class since it would then be different for each manufacturer).

A device supporting the Sensor Configuration Command Class can be configured via the trigger level, but the decision on what the level change should trigger is up to the application. For the movement sensor this trigger level could be an input parameter to the logic that controls the light.

3.65.1 Sensor Trigger Level Set Command

The Sensor Trigger Level Set Command can be used to set different triggers to either a specified value or to the current measured value. The Command also supports to restore a factory default value.

All configurable trigger type and values must be mapped direct from the Multilevel Sensor Command Class.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SENSOR_CONFIGURATION							
Command = SENSOR_TRIGGER_LEVEL_SET							
Default	Current	Reserved					
Sensor Type							
Precision			Scale		Size		
Trigger Value							
…							

Default (1 bit)

Reset level of trigger type to factory default when this bit is set to 1. If any value is set in this frame when the Default bit is 1 this value will be ignored.

Current (1 bit)

The current measured value will be stored as trigger value when this bit is set to 1. The trigger value in this frame will be ignored when the Current bit is set to 1.

Reserved (6 bit)

This field is reserved for future use. Controlling devices must set this field to 0 (zero), while supporting devices must ignore this field.

Precision (3 bit)

The precision field describes what the precision of the trigger value is. The number indicates the number of decimals. The decimal value 1025 with precision 2 is therefore equal to 10.25.

Scale (2 bit)

The Scale is used to indicate what unit the trigger uses. Refer to the table in the Multilevel Sensor Command Class with respect to defined scales for the relevant triggers. New scales/values can be requested from Zensys.

Size (3 bit)

The size field indicates the number of bytes that is used for the trigger value. This field can take values from 1 (001b), 2 (010b) or 4 (100b).

Sensor Type (8 bit)

Type specifies what type of trigger this Command will set. Refer to the Multilevel Sensor Command Class specification, where Sensor Type is defined in the Multilevel Sensor Report Command.

Trigger Value

Refer to the Multilevel Sensor Report Command for information on what trigger values to set.

3.65.2 Sensor Trigger Level Get Command

The Sensor Trigger Level Get Command can request the stored trigger level.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SENSOR_CONFIGURATION							
Command = SENSOR_TRIGGER_LEVEL_GET							

3.65.3 Sensor Trigger Level Report Command

The Sensor Trigger Level Report Command returns the stored trigger value.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SENSOR_CONFIGURATION							
Command = SENSOR_TRIGGER_LEVEL_REPORT							
Sensor Type							
Precision			Scale		Size		
Trigger Value							
…							

Sensor Type (8 bit)

Refer to the Sensor Trigger Level Set Command Class.

Precision (3 bit)

Refer to the Sensor Trigger Level Set Command Class.

Scale (2 bit)

Refer to the Sensor Trigger Level Set Command Class.

Size (3 bit)

Refer to the Sensor Trigger Level Set Command Class.

Trigger Value

Refer to the Sensor Trigger Level Set Command Class.

3.65.4 Mapping example

The report structure from the Multilevel Sensor Command Class can be mapped direct into the Sensor Configuration Set Command Class. This example frame below will set the trigger level in the receiving device to 10.25 degree Celsius.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SENSOR_CONFIGURATION							
Command = SENSOR_TRIGGER_LEVEL_SET							
Default(0)	Current(0)	Reserved					
Temperature (0x01)							
Precision (010b)			Celsius (00b)		Size (010b)		
0x04							
0x01							

3.66 Simple AV Control Command Class, version 1

This Simple AV Control Command Class is used to control an AV device in a Z-Wave network. The Simple AV Control Command Class is suited for IR remote replacement. Furthermore this command class supports Windows Vista Media Center and Media Center 2005 remote controls.

3.66.1 Simple AV Control Set Command

The Simple AV Control Set Command is used to control an AV device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SIMPLE_AV_CONTROL							
Command = SIMPLE_AV_CONTROL_SET							
Sequence Number							
Reserved					Key Attributes		
Item ID MSB							
Item ID LSB							
Command MSB,1							
Command LSB,1							
…							
Command MSB,N							
Command LSB,N							

Sequence Number (8 bit)

The sequence number is incremented each time a Simple AV Control Set Command is issued. The receiving device uses the sequence number to ignore duplicates.

Key Attributes (3 bit)

The key attributes specifies the state of the key. Currently the following key attributes are defined:

Key Attribute	Description
0x00	Key Down – Sent when a new key is pressed. It is mandatory to send a Simple AV Control Set Command when this event occurs.
0x01	Key Up – Sent when the key is released. It is optional to send a Simple AV Control Set Command when this event occurs. Only the sequence number and key attribute parameter is changed in the Command.
0x02	Keep Alive – Sent every 500ms while the key is still held down. This event is used as a failsafe feature for the ramping function, e.g. avoid volume jumps to maximum in case a key up event is not received. The keep alive event can also be used to control the speed of the ramping function, e.g. the first few seconds of the key held down is the speed slow and afterwards will it gradually accelerate. It is optional to send a Simple AV Control Set Command when this event occurs. Only the sequence number and key attribute parameter is changed in the Command.

This list may evolve in the future. In case a key attribute is not supported then it should be ignored.

Reserved (5 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Item ID (16 bit)

The ID of the media item the Command should relate to. No media item is selected in case ID is equal to 0. This field is only used if the AV Content Directory Meta Data Command Class is used.

Command MSB, Command LSB (N x 16 bit)

A 2 byte AV control Command according to the table below. It is possible to send a sequence of Commands in one frame. In case a Command number is not supported then it should be ignored. Be aware of that device related labels are not sent but only used internal in the remote to set up the appropriate address. The address can be a node ID. Command number 1 through 40 is the most popular Commands used in remotes. Command number 41 through 363 is less popular and is sorted in alphanumerical order. Finally is support for Windows Vista Media Center and Media Center 2005 remote controls added from 364 to 377 including 16, 200 and 231. This list may evolve in the future.

Command # [Decimal]	Command # [Hexadecimal]	Universal Label	Description
1	0x0001	Mute	
2	0x0002	Volume Down	Level Down
3	0x0003	Volume Up	Level Up
4	0x0004	Channel Up	Program Up
5	0x0005	Channel Down	Program Down
6	0x0006	0	Preset 10
7	0x0007	1	Preset 1
8	0x0008	2	Preset 2
9	0x0009	3	Preset 3
10	0x000A	4	Preset 4
11	0x000B	5	Preset 5
12	0x000C	6	Preset 6
13	0x000D	7	Preset 7
14	0x000E	8	Preset 8
15	0x000F	9	Preset 9
16	0x0010	Last Channel	Recall, Previous Channel (WMC)
17	0x0011	Display	Info
18	0x0012	Favorite Channel	Favorite
19	0x0013	Play	
20	0x0014	Stop	
21	0x0015	Pause	Still
22	0x0016	Fast Forward	Search Forward
23	0x0017	Rewind	Search Reverse
24	0x0018	Instant Replay	Replay
25	0x0019	Record	
26	0x001A	AC3	Dolby Digital
27	0x001B	PVR Menu	Tivo
28	0x001C	Guide	EPG
29	0x001D	Menu	Settings
30	0x001E	Menu Up	Adjust Up
31	0x001F	Menu Down	Adjust Down
32	0x0020	Menu Left	Cursor Left
33	0x0021	Menu Right	Cursor Right
34	0x0022	Page Up	
35	0x0023	Page Down	
36	0x0024	Select	OK
37	0x0025	Exit	
38	0x0026	Input	Input Select
39	0x0027	Power	Standby
40	0x0028	Enter Channel	Channel Enter
41	0x0029	10	
42	0x002A	11	
43	0x002B	12	
44	0x002C	13	
45	0x002D	14	

Command # [Decimal]	Command # [Hexadecimal]	Universal Label	Description
46	0x002E	15	
47	0x002F	16	
48	0x0030	+10	10+
49	0x0031	+20	20+
50	0x0032	+100	
51	0x0033	-/--	
52	0x0034	3-CH	
53	0x0035	3D	Simulated Stereo
54	0x0036	6-CH Input	6 Channel
55	0x0037	A	
56	0x0038	Add	Write
57	0x0039	Alarm	
58	0x003A	AM	
59	0x003B	Analog	
60	0x003C	Angle	
61	0x003D	Antenna	External
62	0x003E	Antenna East	
63	0x003F	Antenna West	
64	0x0040	Aspect	Size
65	0x0041	Audio 1	Audio
66	0x0042	Audio 2	
67	0x0043	Audio 3	
68	0x0044	Audio Dubbing	
69	0x0045	Audio Level Down	
70	0x0046	Audio Level Up	
71	0x0047	Auto/Manual	
72	0x0048	Aux 1	Aux
73	0x0049	Aux 2	
74	0x004A	B	
75	0x004B	Back	Previous Screen
76	0x004C	Background	Backlight
77	0x004D	Balance	
78	0x004E	Balance Left	
79	0x004F	Balance Right	
80	0x0050	Band	FM/AM
81	0x0051	Bandwidth	Wide/Narrow
82	0x0052	Bass	
83	0x0053	Bass Down	
84	0x0054	Bass Up	
85	0x0055	Blank	
86	0x0056	Breeze Mode	
87	0x0057	Bright	Brighten
88	0x0058	Brightness	
89	0x0059	Brightness Down	
90	0x005A	Brightness Up	

Command # [Decimal]	Command # [Hexadecimal]	Universal Label	Description
91	0x005B	Buy	
92	0x005C	C	
93	0x005D	Camera	
94	0x005E	Category Down	
95	0x005F	Category Up	
96	0x0060	Center	
97	0x0061	Center Down	Center Volume Down
98	0x0062	Center Mode	
99	0x0063	Center Up	Center Volume Up
100	0x0064	Channel/Program	C/P
101	0x0065	Clear	Cancel
102	0x0066	Close	
103	0x0067	Closed Caption	CC
104	0x0068	Cold	A/C
105	0x0069	Color	
106	0x006A	Color Down	
107	0x006B	Color Up	
108	0x006C	Component 1	RGB 1
109	0x006D	Component 2	RGB 2
110	0x006E	Component 3	
111	0x006F	Concert	
112	0x0070	Confirm	Check
113	0x0071	Continue	Continuous
114	0x0072	Contrast	
115	0x0073	Contrast Down	
116	0x0074	Contrast Up	
117	0x0075	Counter	
118	0x0076	Counter Reset	
119	0x0077	D	
120	0x0078	Day Down	
121	0x0079	Day Up	
122	0x007A	Delay	
123	0x007B	Delay Down	
124	0x007C	Delay Up	
125	0x007D	Delete	Erase
126	0x007E	Delimiter	Sub-Channel
127	0x007F	Digest	
128	0x0080	Digital	
129	0x0081	Dim	Dimmer
130	0x0082	Direct	
131	0x0083	Disarm	
132	0x0084	Disc	
133	0x0085	Disc 1	
134	0x0086	Disc 2	
135	0x0087	Disc 3	

Command # [Decimal]	Command # [Hexadecimal]	Universal Label	Description
136	0x0088	Disc 4	
137	0x0089	Disc 5	
138	0x008A	Disc 6	
139	0x008B	Disc Down	
140	0x008C	Disc Up	
141	0x008D	Disco	
142	0x008E	Edit	
143	0x008F	Effect Down	
144	0x0090	Effect Up	
145	0x0091	Eject	Open/Close
146	0x0092	End	
147	0x0093	EQ	Equalizer
148	0x0094	Fader	
149	0x0095	Fan	
150	0x0096	Fan High	
151	0x0097	Fan Low	
152	0x0098	Fan Medium	
153	0x0099	Fan Speed	
154	0x009A	Fastext Blue	
155	0x009B	Fastext Green	
156	0x009C	Fastext Purple	
157	0x009D	Fastext Red	
158	0x009E	Fastext White	
159	0x009F	Fastext Yellow	
160	0x00A0	Favorite Channel Down	
161	0x00A1	Favorite Channel Up	
162	0x00A2	Finalize	
163	0x00A3	Fine Tune	
164	0x00A4	Flat	
165	0x00A5	FM	
166	0x00A6	Focus Down	
167	0x00A7	Focus Up	
168	0x00A8	Freeze	
169	0x00A9	Front	
170	0x00AA	Game	
171	0x00AB	GoTo	Index Search
172	0x00AC	Hall	
173	0x00AD	Heat	
174	0x00AE	Help	
175	0x00AF	Home	
176	0x00B0	Index	VISS
177	0x00B1	Index Forward	
178	0x00B2	Index Reverse	
179	0x00B3	Interactive	Planner
180	0x00B4	Intro Scan	

Command # [Decimal]	Command # [Hexadecimal]	Universal Label	Description
181	0x00B5	Jazz	
182	0x00B6	Karaoke	
183	0x00B7	Keystone	
184	0x00B8	Keystone Down	
185	0x00B9	Keystone Up	
186	0x00BA	Language	SAP
187	0x00BB	Left Click	
188	0x00BC	Level	Volume
189	0x00BD	Light	Lamp
190	0x00BE	List	My Shows
191	0x00BF	Live TV	Return to Live
192	0x00C0	Local/Dx	
193	0x00C1	Loudness	
194	0x00C2	Mail	Email
195	0x00C3	Mark	Bookmark
196	0x00C4	Memory Recall	
197	0x00C5	Monitor	Tape Monitor
198	0x00C6	Movie	
199	0x00C7	Multi Room	
200	0x00C8	Music	TV/Radio, My Music (WMC)
201	0x00C9	Music Scan	Memory Scan
202	0x00CA	Natural	
203	0x00CB	Night	
204	0x00CC	Noise Reduction	Dolby NR
205	0x00CD	Normalize	Personal Preference
206	0x00CE	Discrete input Cable	CATV
207	0x00CF	Discrete input CD 1	CD
208	0x00D0	Discrete input CD 2	CDR
209	0x00D1	Discrete input CDR	Compact Disc Recorder
210	0x00D2	Discrete input DAT	Digital Audio Tape
211	0x00D3	Discrete input DVD	Digital Video Disk
212	0x00D4	Discrete input DVI	Digital Video Interface
213	0x00D5	Discrete input HDTV	
214	0x00D6	Discrete input LD	Laser Disc
215	0x00D7	Discrete input MD	Mini Disc
216	0x00D8	Discrete input PC	Personal Computer
217	0x00D9	Discrete input PVR	Personal Video Recorder
218	0x00DA	Discrete input TV	
219	0x00DB	Discrete input TV/VCR	TV/DVD
220	0x00DC	Discrete input VCR	
221	0x00DD	One Touch Playback	OTPB
222	0x00DE	One Touch Record	OTR
223	0x00DF	Open	
224	0x00E0	Optical	
225	0x00E1	Options	

Command # [Decimal]	Command # [Hexadecimal]	Universal Label	Description
226	0x00E2	Orchestra	
227	0x00E3	PAL/NTSC	System Select
228	0x00E4	Parental Lock	Parental Control
229	0x00E5	PBC	Playback Control
230	0x00E6	Phono	
231	0x00E7	Photos	Pictures, My Pictures (WMC)
232	0x00E8	Picture Menu	Picture Adjust
233	0x00E9	Picture Mode	Smart Picture
234	0x00EA	Picture Mute	
235	0x00EB	PIP Channel Down	
236	0x00EC	PIP Channel Up	
237	0x00ED	PIP Freeze	
238	0x00EE	PIP Input	PIP Mode
239	0x00EF	PIP Move	PIP Position
240	0x00F0	PIP Off	
241	0x00F1	PIP On	PIP
242	0x00F2	PIP Size	
243	0x00F3	PIP Split	Multi Screen
244	0x00F4	PIP Swap	PIP Exchange
245	0x00F5	Play Mode	
246	0x00F6	Play Reverse	
247	0x00F7	Power Off	
248	0x00F8	Power On	
249	0x00F9	PPV	Pay Per View
250	0x00FA	Preset	
251	0x00FB	Program	Program Memory
252	0x00FC	Progressive Scan	Progressive
253	0x00FD	ProLogic	Dolby Prologic
254	0x00FE	PTY	Audio Program Type
255	0x00FF	Quick Skip	Commercial Skip
256	0x0100	Random	Shuffle
257	0x0101	RDS	Radio Data System
258	0x0102	Rear	
259	0x0103	Rear Volume Down	
260	0x0104	Rear Volume Up	
261	0x0105	Record Mute	
262	0x0106	Record Pause	
263	0x0107	Repeat	
264	0x0108	Repeat A-B	
265	0x0109	Resume	
266	0x010A	RGB	Red Green Blue Component Video
267	0x010B	Right Click	
268	0x010C	Rock	
269	0x010D	Rotate Left	
270	0x010E	Rotate Right	

Command # [Decimal]	Command # [Hexadecimal]	Universal Label	Description
271	0x010F	SAT	Sky
272	0x0110	Scan	Channel Scan
273	0x0111	Scart	
274	0x0112	Scene	
275	0x0113	Scroll	
276	0x0114	Services	
277	0x0115	Setup Menu	Setup
278	0x0116	Sharp	
279	0x0117	Sharpness	
280	0x0118	Sharpness Down	
281	0x0119	Sharpness Up	
282	0x011A	Side A/B	
283	0x011B	Skip Forward	Next
284	0x011C	Skip Reverse	Previous
285	0x011D	Sleep	Off Timer
286	0x011E	Slow	
287	0x011F	Slow Forward	
288	0x0120	Slow Reverse	
289	0x0121	Sound Menu	Audio Menu
290	0x0122	Sound Mode	Smart Sound
291	0x0123	Speed	Record Speed
292	0x0124	Speed Down	
293	0x0125	Speed Up	
294	0x0126	Sports	Digital Surround Processing
295	0x0127	Stadium	
296	0x0128	Start	
297	0x0129	Start ID Erase	Erase
298	0x012A	Start ID Renumber	Renumber
299	0x012B	Start ID Write	Write
300	0x012C	Step	
301	0x012D	Stereo/Mono	L/R
302	0x012E	Still Forward	Frame Advance
303	0x012F	Still Reverse	Frame Reverse
304	0x0130	Subtitle	Subtitle On-Off
305	0x0131	Subwoofer Down	
306	0x0132	Subwoofer Up	
307	0x0133	Super Bass	Bass Boost
308	0x0134	Surround	
309	0x0135	Surround Mode	Sound Field
310	0x0136	S-Video	
311	0x0137	Sweep	Oscillate
312	0x0138	Synchro Record	CD Synchro
313	0x0139	Tape 1	Deck 1
314	0x013A	Tape 1-2	Deck 1-2
315	0x013B	Tape 2	Deck 2

Command # [Decimal]	Command # [Hexadecimal]	Universal Label	Description
316	0x013C	Temperature Down	
317	0x013D	Temperature Up	
318	0x013E	Test Tone	
319	0x013F	Text	Teletext
320	0x0140	Text Expand	
321	0x0141	Text Hold	
322	0x0142	Text Index	
323	0x0143	Text Mix	
324	0x0144	Text Off	
325	0x0145	Text Reveal	
326	0x0146	Text Subpage	
327	0x0147	Text Timed Page	
328	0x0148	Text Update	Text Cancel
329	0x0149	Theater	Cinema EQ
330	0x014A	Theme	Category Select
331	0x014B	Thumbs Down	
332	0x014C	Thumbs Up	
333	0x014D	Tilt Down	
334	0x014E	Tilt Up	
335	0x014F	Time	Clock
336	0x0150	Timer	
337	0x0151	Timer Down	
338	0x0152	Timer Up	
339	0x0153	Tint	
340	0x0154	Tint Down	
341	0x0155	Tint Up	
342	0x0156	Title	Top Menu
343	0x0157	Track	Chapter
344	0x0158	Tracking	
345	0x0159	Tracking Down	
346	0x015A	Tracking Up	
347	0x015B	Treble	
348	0x015C	Treble Down	
349	0x015D	Treble Up	
350	0x015E	Tune Down	Audio Tune Down
351	0x015F	Tune Up	Audio Tune Up
352	0x0160	Tuner	
353	0x0161	VCR Plus+	Showview
354	0x0162	Video 1	A/V 1
355	0x0163	Video 2	A/V 2
356	0x0164	Video 3	A/V 3
357	0x0165	Video 4	A/V 4
358	0x0166	Video 5	A/V 5
359	0x0167	View	
360	0x0168	Voice	Vocals

Command # [Decimal]	Command # [Hexadecimal]	Universal Label	Description
361	0x0169	Zoom	Magnify
362	0x016A	Zoom In	Zoom Up
363	0x016B	Zoom Out	Zoom Down
364	0x016C	eHome	(WMC)
365	0x016D	Details	(WMC)
366	0x016E	DVD Menu	(WMC)
367	0x016F	My TV	(WMC)
368	0x0170	Recorded TV	(WMC)
369	0x0171	My Videos	(WMC)
370	0x0172	DVD Angle	(WMC)
371	0x0173	DVD Audio	(WMC)
372	0x0174	DVD Subtitle	(WMC)
373	0x0175	Radio	(WMC)
374	0x0176	#	(WMC)
375	0x0177	*	(WMC)
376	0x0178	OEM 1	(WMC)
377	0x0179	OEM 2	(WMC)

Table 4, AV Control codes and associated label

3.66.2 Simple AV Control Get Command

The Simple AV Control Get Command is used request the number of reports necessary to report the supported AC Commands from the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SIMPLE_AV_CONTROL							
Command = SIMPLE_AV_CONTROL_GET							

3.66.3 Simple AV Control Report Command

The Simple AV Control Report Command is used to report the necessary number of reports to report the supported AC Commands from the device. The Simple AV Control Report Command can be send as a result of receiving an Simple AV Control Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SIMPLE_AV_CONTROL							
Command = SIMPLE_AV_CONTROL_REPORT							
Number of reports							

Number of reports (8 bit)

The number of reports necessary to report the supported AC Commands.

3.66.4 Simple AV Control Supported Get Command

The Simple AV Control Supported Get Command is used to request the AV Commands supported by the AV device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SIMPLE_AV_CONTROL							
Command = SIMPLE_AV_CONTROL_SUPPORTED_GET							
Report No							

Report No (8 bit)

Report no. field specify the report number to be requested. The report no. values must be a sequence starting from 1.

3.66.5 Simple AV Control Supported Report Command

The Simple AV Control Supported Report Command is used to report the supported AC Commands from the device. The Simple AV Control Report Command can be send requested by the Simple AV Control Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SIMPLE_AV_CONTROL							
Command = SIMPLE_AV_CONTROL_SUPPORTED_REPORT							
Report No							
Bit Mask 1							
…							
Bit Mask N							

Report No (8 bit)

Report no. field specify the request report number.

Bit Mask 1 .. Bit Mask N (N * Bytes)

The Bit Mask fields describe the supported AV Control Commands by the device. The bit 0 in Bit Mask 1 field is used to indicate whether Command #1 is supported or not. The Command #1 is supported if the bit is 1 and the opposite if 0. The bit 1 in Bit Mask 1 field is used by Command #2 and so forth. It is only necessary to send the Bit Mask fields from 1 and up to the one indicating the last supported Command #. It is not allowed to send more than 45 Bit Mask fields in one frame. The number of Bit Mask fields transmitted can be determined from the length field in the frame.

3.67 Basic Tariff Information Command Class, version 1

This Basic Tariff Information Command Class for use with a single element or dual element meter, and for use with import (electricity received from grid) rates only. The command class is kept as simple as possible without any pricing information.

This command class supports a GET and REPORT.

No SET command is supported, as it is not appropriate to set any of the parameters through Z-Wave.

3.67.1 Basic Tariff Information Get Command

The Basic Tariff Information Get command is used to request current tariff information from the meter.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_BASIC_TARIFF_INFO							
Command = BASIC_TARIFF_INFO_GET							

The response is the Basic Tariff Information Report.

3.67.2 Basic Tariff Information Report Command

The Basic Tariff Information Report command returns information on the number of import rates supported, and current import rate information. The Tariff Report command can be sent unsolicited or as a result of receiving the Basic Tariff Get Information command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_BASIC_TARIFF_INFO							
Command = BASIC_TARIFF_INFO_REPORT							
Dual	Reserved			Total No. Import Rates			
Reserved				E1 – Current Rate in Use			
E1 - Rate Consumption Register – MSB							
E1 - Rate Consumption Register							
E1 - Rate Consumption Register							
E1 - Rate Consumption Register – LSB							
E1 – Time for Next Rate – Hours							
E1 – Time for Next Rate – Minutes							
E1 – Time for Next Rate – Seconds							
Reserved				E2 – Current Rate in Use			
E2 - Rate Consumption Register – MSB							
E2 - Rate Consumption Register							
E2 - Rate Consumption Register							
E2 - Rate Consumption Register – LSB							

Dual (1 bit)

Single Element = 0, Two Elements = 1.

If single element the E2 fields are skipped in the frame. E1 – Time for Next Rate – Seconds will be the last byte of the message and the number of data bytes will be 9.

If two elements the E2 fields are present and the number of data bytes will be 14.

Total Number of Import Rates Supported (7 bit)

Field specifies the number of import rates E1 (and E2) supported by the meter. Range of legal decimal values are 1…8. No units used. The decimal values 0 and 9…15 are reserved and shall be ignored by receiving devices.

Reserved

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

E1 – Current Rate in Use (8 bit)

Field specifies the current rate in use. Range of legal decimal values are 1…8. No units used. The decimal values 0 and 9…15 are reserved and shall be ignored by receiving devices.

E1 – Rate Consumption Register (32 bit)

The meter has a 32-bit consumption register, for the energy used in each rate. This register is the rate consumption register for the current rate in use now on element 1. Units are in Wh.

E1 – Time for Next Rate – Hours (8 bit)

Field specifies the hour value of the time that the rate is due to change on element 1. Range of legal decimal values are 0…23, or 255. The values 24…254 are reserved and shall be ignored by receiving devices.

E1 – Time for Next Rate – Minutes (8 bit)

Field specifies the minute value of the time that the rate is due to change on element 1. Range of legal decimal values are 0…59, or 255. The decimal values 60…254 are reserved and shall be ignored by receiving devices.

E1 – Time for Next Rate – Seconds (8 bit)

Field specifies the second value of the time that the rate is due to change on element 1. Range of legal decimal values are 0…59, or 255. The decimal values 60…254 are reserved and shall be ignored by receiving devices.

NOTE: 255 in each field of the Time to Next Rate specifies no switching time is in use, which is appropriate for single rate meters. 255 is only a legal value if used in all three Time to Next Rate fields.

E2 – Current Rate in Use (8 bit)

Field specifies the current rate in use on element 2. Range of legal decimal values are 1…8. No units used. The decimal values 0 and 9…15 are reserved and shall be ignored by receiving devices.

E2 – Rate Consumption Register (32 bit)

The meter has a 32-bit consumption register, for the energy used in each rate. This register is the rate consumption register for the current rate in use now on element 2. Units are in Wh.

3.68 Thermostat Fan Mode Command Class, version 1

The Thermostat Fan Mode Command Class is used for the HVAC's systems manual fan.

3.68.1 Thermostat Fan Mode Set Command

The Thermostat Fan Mode Set Command is used to set the fan mode in the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_FAN_MODE							
Command = THERMOSTAT_FAN_MODE_SET							
Reserved				Fan Mode			

Fan Mode (8 bit)

Fan Mode	Description
0	Auto / Auto Low – Will turn the manual fan operation off unless turned on by the furnace or AC. Lower speed is selected in case it is a two-speed fan.
1	On / On Low – Will turn the manual fan operation on. Lower speed is selected in case it is a two-speed fan.
2	Auto High – Will turn the manual fan operation off unless turned on by the furnace or AC. High speed is selected in case it is a two-speed fan.
3	On High – Will turn the manual fan operation on. High speed is selected in case it is a two-speed fan.

Reserved (4 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

3.68.2 Thermostat Fan Mode Get Command

The Thermostat Fan Mode Get Command is used to request the fan mode in the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_FAN_MODE							
Command = THERMOSTAT_FAN_MODE_GET							

3.68.3 Thermostat Fan Mode Report Command

The Thermostat Fan Mode Report is used to report the fan mode in a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_FAN_MODE							
Command = THERMOSTAT_FAN_MODE_REPORT							
Reserved				Fan Mode			

Fan Mode (8 bit)

Refer to description under the Thermostat Fan Mode Set Command.

Reserved (4 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

3.68.4 Thermostat Fan Mode Supported Get Command

The Thermostat Fan Mode Supported Get Command is used to request the supported fan modes from the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_FAN_MODE							
Command = THERMOSTAT_FAN_MODE_SUPPORTED_GET							

3.68.5 Thermostat Fan Mode Supported Report Command

The Thermostat Fan Mode Supported Report is used to report the supported fan modes from the device. It can be sent either unsolicited or requested by the Fan Supported Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_FAN_MODE							
Command = THERMOSTAT_FAN_MODE_SUPPORTED_REPORT							
Bit Mask 1							
…							
Bit Mask N							

Bit Mask 1 .. Bit Mask N (N * Byte)

The Bit Mask fields describe the supported fan modes by the thermostat. The bit 0 in Bit Mask 1 field is used to indicate whether Fan Mode = 0 (Auto / Auto Low) is supported or not. The fan mode is supported if the bit is 1 and the opposite if 0. The bit 1 in Bit Mask 1 field is used by Fan Mode = 1 (On / On Low) and so forth. It is only necessary to send the Bit Mask fields from 1 and up to the one indicating the last supported fan mode. The number of Bit Mask fields transmitted can be determined from the length field in the frame.

3.69 Thermostat Fan State Command Class, version 1

The Thermostat Fan State Command Class is used to obtain the fan operating state of the thermostat.

3.69.1 Thermostat Fan State Get Command

The Thermostat Fan State Get Command is used to request the fan operating state from the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_FAN_STATE							
Command = THERMOSTAT_FAN_STATE_GET							

3.69.2 Thermostat Fan State Report Command

The Thermostat Fan State Report is used to report the fan operating state of the device. It can be sent either unsolicited or requested by the Thermostat Fan State Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_FAN_STATE							
Command = THERMOSTAT_FAN_STATE_REPORT							
Reserved				Fan Operating State			

Reserved (4 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Fan Operating State (8 bit)

The fan operating state identifier can be set to the following values:

Fan Operating State	Description
0	Idle
1	Running / Running Low - Lower speed is selected in case it is a two-speed fan.
2	Running High - High speed is selected in case it is a two-speed fan.

This list may evolve in the future. If a fan operating state is not supported then it should be ignored.

3.70 Thermostat Mode Command Class, version 1-2

The Thermostat Mode Command Class is used to control a thermostat. These Commands allow applications to set and get the thermostat parameters. Version 2 extends the available number of modes.

NOTE: A device supporting the Thermostat Mode Command Class cannot support Auto and Auto Changeover mode simultaneously. Devices controlling a device supporting the Thermostat Mode Command Class must be able to control both modes to ensure interoperability.

3.70.1 Thermostat Mode Set Command

The Thermostat Mode Set Command is used to set the wanted mode in the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_MODE							
Command = THERMOSTAT_MODE_SET							
Reserved			Mode				

Mode (5 bit)

The thermostat mode identifier can be set to the following values:

Mode	Description	Version
0	Off – System is off. No heating and cooling will come on. Version 1.	1
1	Heat – Only heating will occur.	1
2	Cool – Only cooling will occur.	1
3	Auto – Heating or cooling will come on according to the heating and cooling setpoints. The system will automatically switch between heating and cooling when the temperature exceeds the setpoints.	1
4	Auxiliary/Emergency Heat – A heat pump (especially air exchange types) are not efficient when the outside temperature is below 35 degrees Fahrenheit (approaching 0 degrees centigrade). Thus, the thermostat may be put into Aux heat mode simply to use a more efficient secondary heat source when there are no failures of the compressor or heat pump unit itself.	1
5	Resume – The system will resume from last active mode.	1
6	Fan Only – Only cycle fan to circulate air.	1
7	Furnace – Only cycle fan to circulate air.	1
8	Dry Air – The system will cycle cooling in relation to the room and set point temperatures in order to remove moisture from ambient (Dehumidification).	1
9	Moist Air – (Humidification).	1
10	Auto Changeover – Heating or cooling will come on according to the auto changeover setpoint.	1
11	Energy Save Heat – Energy Save Mode Heating will occur (usually lower than normal setpoint).	2
12	Energy Save Cool – Energy Save Mode Cooling will occur (usually higher than normal setpoint).	2
13	AWAY – special Heating Mode, i.e. preventing water from freezing in forced water systems.	2

This list may evolve in the future. If a mode is not supported then it should be ignored.

Reserved (3 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

3.70.2 Thermostat Mode Get Command

The Thermostat Mode Get Command is used to request the current mode from the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_MODE							
Command = THERMOSTAT_MODE_GET							

3.70.3 Thermostat Mode Report Command

The Thermostat Mode Report Command is used to report the mode from the device. It can be sent either unsolicited or requested by the Mode Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_MODE							
Command = THERMOSTAT_MODE_REPORT							
Reserved			Mode				

Mode (5 bit)

Refer to description under the Thermostat Mode Set Command.

Reserved (3 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

3.70.4 Thermostat Mode Supported Get Command

The Thermostat Mode Supported Get Command is used to request the supported modes from the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_MODE							
Command = THERMOSTAT_MODE_SUPPORTED_GET							

3.70.5 Thermostat Mode Supported Report Command

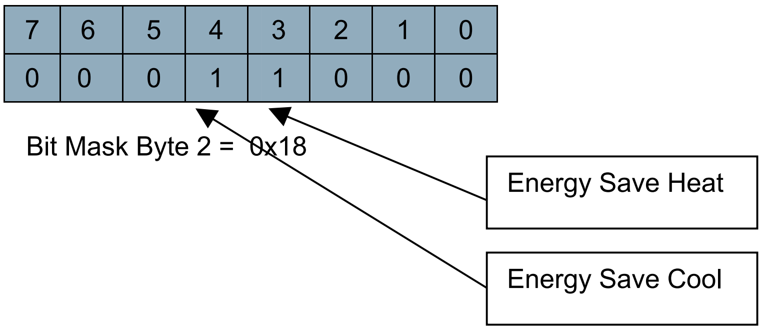
The Thermostat Mode Supported Report Command is used to report the supported modes from the device. It can be sent either unsolicited or requested by the Mode Supported Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_MODE							
Command = THERMOSTAT_MODE_SUPPORTED_REPORT							
Bit Mask 1							
…							
Bit Mask N							

Bit Mask 1 .. Bit Mask N (N * Bytes)

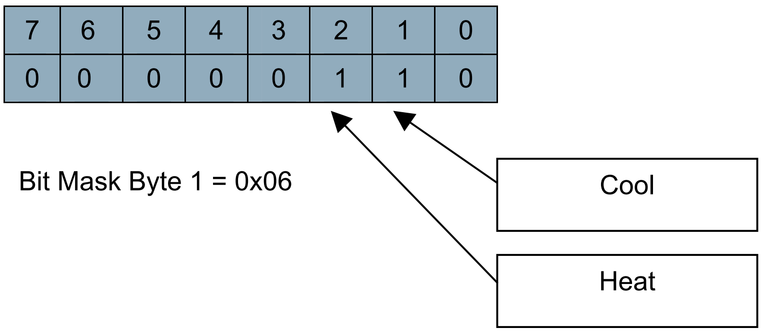
The Bit Mask fields describe the supported modes by the device. The bit 0 in Bit Mask 1 field is used to indicate whether Mode = 0 (Off) is supported or not. The mode is supported if the bit is 1 and the opposite if 0. The bit 1 in Bit Mask 1 field is used by Mode = 1 (Heat) and so forth. It is only necessary to send the Bit Mask fields from 1 and up to the one indicating the last supported mode. The number of Bit Mask fields transmitted can be determined from the length field in the frame.

For example if thermostat supports Heat, Cool, Energy Save Heat and Energy Save Cool bit mask would be 0x18 and 0x06:



7	6	5	4	3	2	1	0
0	0	0	1	1	0	0	0

Bit Mask Byte 2 = 0x18



7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0

Bit Mask Byte 1 = 0x06

3.71 Thermostat Operating State Command Class, version 1

The Thermostat Operating State Command Class is used to obtain the operating state of the thermostat.

3.71.1 Thermostat Operating State Get Command

The Thermostat Operating State Get Command is used to request the operating state from the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_OPERATING_STATE							
Command = THERMOSTAT_OPERATING_STATE_GET							

3.71.2 Thermostat Operating State Report Command

The Thermostat Operating State Report is used to report the operating state of the device. It can be sent either unsolicited or requested by the Operating State Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_OPERATING_STATE							
Command = THERMOSTAT_OPERATING_STATE_REPORT							
Reserved				Operating State			

Reserved (4 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Operating State (8 bit)

The thermostat operating state identifier can be set to the following values:

Operating State	Description
0	Idle
1	Heating
2	Cooling
3	Fan Only
4	Pending Heat. Short cycle prevention feature used in heat pump applications to protect the compressor.
5	Pending Cool. Short cycle prevention feature used in heat pump applications to protect the compressor.
6	Vent/Economizer.

This list may evolve in the future.

3.72 Thermostat Setback Command Class, version 1

The Thermostat Setback Command Class is used to change the current state of a non-schedule setback thermostat.

3.72.1 Thermostat Setback Set Command

The Thermostat Setback Set Command is used to set the state of the thermostat.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_SETBACK							
Command = THERMOSTAT_SETBACK_SET							
Reserved						Setback Type	
Setback State							

Setback Type (2bit)

The setback type field can assume the following values:

Binary	Decimal	Description
0b00	0	No override
0b01	1	Temporary override
0b10	2	Permanent override
0b11	3	Reserved

Note: The temporary override provides an opportunity to implement a timer or equivalent in the device. A temporary override will, if a timer is implemented, be terminated by the timer, if no timer is implemented the temporary override must act as permanent override. If the temporary override is implemented it must be documented in the user's manual.

Setback State (8 bit)

The Setback State can assume the following values:

Setback State		Description
Hexadecimal	Decimal	
0x80 … 0xFF 0x00 0x01 … 0x78	-128 … -1 0 1 … 120	The setback in 1/10 degrees (Kelvin) Example: 0 = 0 degrees setback 1 = 0.1 degrees is added to the setpoint 2 = 0.2 degrees is added to the setpoint -1 = 0.1 degrees is subtracted from the setpoint -2 = 0.2 degrees is subtracted from the setpoint
0x79	121	Frost Protection
0x7A	122	Energy Saving Mode
0x7B – 0x7E	123 – 126	Reserved
0x7F	127	Unused State

When converting between Celsius and Fahrenheit proper rounding must be applied with at least two decimals in the internal calculations of a device to avoid rounding errors.

Note: The implementation of Energy Saving Mode is manufacturer specific, and must be documented in the User's Manual.
If the device is set to an unreachable state, the device must set itself to the state which is closest possible to the requested.

3.72.2 Thermostat Setback Get Command

The Thermostat Setback Get Command is used to request the current state of the thermostat.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_SETBACK							
Command = THERMOSTAT_SETBACK_GET							

3.72.3 Thermostat Setback Report Command

The Thermostat Setback Report Command is used to report the current state of the thermostat.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_SETBACK							
Command = THERMOSTAT_SETBACK_REPORT							
Reserved						Setback Type	
Setback State							

Note: If the device is set to an unreachable state, the device must report the state which is closest possible to the requested.

Setback Type (2bit)

Refer to description under the Thermostat Setback Set Command

Setback State (8 bit)

Refer to description under the Thermostat Setback Set Command.

3.73 Thermostat Setpoint Command Class, version 1-2

The Thermostat Setpoint Command Class is used for setpoint handling. Version 2 extends the available number of setpoint types.

3.73.1 Thermostat Setpoint Set Command

The Thermostat Setpoint Set Command is used to set the setpoint in the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_SETPOINT							
Command = THERMOSTAT_SETPOINT_SET							
Reserved				Setpoint Type			
Precision			Scale		Size		
Value 1							
Value 2							
..							
Value n							

Reserved (4 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Setpoint Type (4 bit)

The setpoint type number specifies the setpoint to be set in the thermostat. Currently the following setpoint types are defined:

Setpoint Type	Description	Version
1	Heating #1	1
2	Cooling #1	1
7	Furnace	1
8	Dry Air	1
9	Moist Air	1
10	Auto changeover	1
11	Energy Save Heating	2
12	Energy Save Cooling	2
13	Away Heating	2

This list may evolve in the future. In case a setpoint type is not supported then it should be ignored.

Precision (3 bit)

The precision field describes what the precision of the setpoint value is. The number indicates the number of decimals. The decimal value 1025 with precision 2 is therefore equal to 10.25.

Scale (2 bit)

The scale field indicates the temperature scale used, 0 indicate the use of the Celsius temperature scale and 1 indicates use of the Fahrenheit scale.

Size (3 bit)

The size field indicates the number of bytes that is used for the setpoint value. This field can take values from 1 (001b), 2 (010b) or 4 (100b).

Value (variable)

The value is a signed field. The field can be 1, 2 or 4 bytes in size. The first byte is the most significant byte. The table below shows signed decimal values together with their hexadecimal equivalents.

Signed 1 byte decimal value	Hexadecimal		Signed 2 bytes decimal value	Hexadecimal
127	0x7F		32767	0x7FFF
25	0x19		1025	0x0401
2	0x02		2	0x0002
1	0x01		1	0x0001
0	0x00		0	0x0000
-1	0xFF		-1	0xFFFF
-2	0xFE		-2	0xFFFE
-25	0xE7		-1025	0xFBFF
-128	0x80		-32768	0x8000

3.73.2 Thermostat Setpoint Get Command

The Thermostat Setpoint Get Command is used to request a given setpoint type in a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_SETPOINT							
Command = THERMOSTAT_SETPOINT_GET							
Reserved				Setpoint Type			

Reserved (4 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Setpoint Type (4 bit)

Refer to description under the Thermostat Setpoint Set Command.

3.73.3 Thermostat Setpoint Report Command

The Thermostat Setpoint Report Command is used to report the value of the setpoint type in a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_SETPOINT							
Command = THERMOSTAT_SETPOINT_REPORT							
Reserved				Setpoint Type			
Precision			Scale		Size		
Value 1							
Value 2							
..							
Value n							

Reserved (4 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Setpoint Type (4 bit)

Refer to description under the Thermostat Setpoint Set Command.

Precision (3 bit)

Refer to description under the Thermostat Setpoint Set Command.

Scale (2 bit)

Refer to description under the Thermostat Setpoint Set Command.

Size (3 bit)

Refer to description under the Thermostat Setpoint Set Command.

Value (variable)

Refer to description under the Thermostat Setpoint Set Command.

3.73.4 Thermostat Setpoint Supported Get Command

The Thermostat Setpoint Supported Get Command is used to request the setpoint types supported by the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_SETPOINT							
Command = THERMOSTAT_SETPOINT_SUPPORTED_GET							

3.73.5 Thermostat Setpoint Supported Report Command

The Thermostat Setpoint Supported Report Command is used to report the setpoint types supported by the device. It can be sent either unsolicited or requested by the Setpoint Supported Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_THERMOSTAT_SETPOINT							
Command = THERMOSTAT_SETPOINT_SUPPORTED_REPORT							
Bit Mask 1							
…							
Bit Mask N							

Bit Mask 1 .. Bit Mask N (N * Bytes)

The Bit Mask fields describe the supported setpoint types by the device. The bit 1 in Bit Mask 1 field is used to indicate whether Setpoint Type = 1 (Heating #1) is supported or not. The setpoint type is supported if the bit is 1 and the opposite if 0. The bit 2 in Bit Mask 1 field is used by Setpoint Type = 2 (Cooling #1) and so forth. It is only necessary to send the Bit Mask fields from 1 and up to the one indicating the last supported setpoint type. The number of Bit Mask fields transmitted can be determined from the length field in the frame.

3.74 Time Command Class, version 1

This Time Command Class version1 is used to read date and time from a device in a Z-Wave network.

Notice that the former Time Command Class version 1 (Revision 4 of this document) is discontinued and replaced by a new one.

3.74.1 Time Get Command

The Time Get Command is used to request current time. Be aware that the communication overhead can be significant in case routing is necessary.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_TIME							
Command = TIME_GET							

3.74.2 Time Report Command

The Time Report Command returns current time. The Time Report command can be send unsolicited or as a result of receiving a Time Get command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_TIME							
Command = TIME_REPORT							
RTC failure	Reserved		Hour Local Time				
Minute Local Time							
Second Local Time							

RTC failure (1 bit)

Many RTC chips have a stop bit indicating if the oscillator has been stopped. The RTC failure bit is used to indicate to the receiving unit that the RTC in the device has been stopped and that the time might be inaccurate.
If the sending/receiving device does not support this feature it must ignore this bit. As a result of this, the bit can only be used to indicate that the time might be inaccurate and not to inform a device that it must ignore the time.

Reserved (2 bit)

The reserved field is for future use. The implementation shall zero these fields and shall make no assumptions on the values of these fields nor perform processing based on their content.

Hour Local Time (8 bit)

Specify the number of complete hours that have passed since midnight (00-23) in local time.

Minute Local Time (8 bit)

Specify the number of complete minutes that have passed since the start of the hour (00-59) in local time.

Second Local Time (8 bit)

Specify the number of complete seconds since the start of the minute (00-59) in local time. The value 60 used to keep UTC from wandering away is not supported.

3.74.3 Date Get Command

The Date Get Command is used to request current date adjusted according to the local time zone and daylight savings time. Be aware that the communication overhead can be significant in case routing is necessary.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_TIME							
Command = DATE_GET							

3.74.4 Date Report Command

The Date Report Command returns current date adjusted according to the local time zone and daylight savings time. The Date Report command can be send unsolicited or as a result of receiving a Date Get command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_TIME							
Command = DATE_REPORT							
Year 1							
Year 2							
Month							
Day							

Year 1..2 (16 bit)

Specify the year in the usual Gregorian calendar. The first byte (Year 1) is the most significant byte.

Example 2007: Year1= 00000111, Year2=11010111

Month (8 bit)

Specify the month of the year between 01 (January) and 12 (December).

Day (8 bit)

Specify the day of the month between 01 and 31.

3.75 Time Command Class, version 2

The Time Command Class version 2 enables setting time zone offset and daylight savings parameters. The data formats are based on the International Standard ISO 8601.

The Commands not mentioned here will remain the same as in version 1**.**

3.75.1 Time Offset Get Command

The Time Offset Get Command is used to request time zone offset and daylight savings parameters.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_TIME							
Command = TIME_OFFSET_GET							

3.75.2 Time Offset Set Command

The Time Offset Set Command is used to set time zone offset and daylight savings parameters to achieve local time depending on the clock source.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_TIME							
Command = TIME_OFFSET_SET							
Sign TZO	Hour TZO						
Minute TZO							
Sign Offset DST	Minute Offset DST						
Month Start DST							
Day Start DST							
Hour Start DST							
Month End DST							
Day End DST							
Hour End DST							

Sign TZO (1 bit)

Plus (0) or minus (1) sign to indicate a positive or negative offset from UTC.

Hour TZO (7 bit)

Specify the number of hours that the originating time zone deviates from UTC. Refer to the DST field regarding daylight savings handling.

Minute TZO (7 bit)

Specify the number of minutes that the originating time zone deviates UTC. Refer to the DST field regarding daylight savings handling.

Sign Offset DST (1 bit)

Plus (0) or minus (1) sign to indicate a positive or negative offset from UTC.

Minute Offset DST (7 bit)

Specify the number of complete minutes the time must be adjusted when daylight savings mode is enabled.

Month Start DST (8 bit)

Specify the month of the year between 01 (January) and 12 (December) when daylight savings mode is enabled.

Day Start DST (8 bit)

Specify the day of the month between 01 and 31 when daylight savings mode is enabled.

Hour Start DST (8 bit)

Specify the number of complete hours that have passed since midnight (00-23) in local time when daylight savings mode is enabled.

Month End DST (8 bit)

Specify the month of the year between 01 (January) and 12 (December) when daylight savings mode is disabled.

Day End DST (8 bit)

Specify the day of the month between 01 and 31 when daylight savings mode is disabled.

Hour End DST (8 bit)

Specify the number of complete hours that have passed since midnight (00-23) in local time when daylight savings mode is disabled.

3.75.3 Time Offset Report Command

The Time Offset Report Command returns time zone offset and daylight savings parameters and can be requested by the Time Offset Get command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_TIME							
Command = TIME_OFFSET_REPORT							
Sign TZO	Hour TZO						
Minute TZO							
Sign Offset DST	Minute Offset DST						
Month Start DST							
Day Start DST							
Hour Start DST							
Month End DST							
Day End DST							
Hour End DST							

Refer to description under the Time Offset Set command.

3.76 Time Parameters Command Class, version 1

This Time Parameters Command Class is used to set date and time in a device hosting this facility. In case the clock is updated via an external source such as SAT, internet, Rugby/Frankfurt source then omit this command class. Time zone offset and daylight savings can be set in the Time Command Class if necessary. The data formats are based on the International Standard ISO 8601.

3.76.1 Time Parameters Set Command

The Time Parameters Set Command is used to set current date and time in Universal Time (UTC). Be aware that the communication overhead can be significant in case routing is necessary. For two nodes within direct range is the communication overhead less than 100 msec.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_TIME_PARAMETERS							
Command = TIME_PARAMETERS_SET							
Year 1							
Year 2							
Month							
Day							
Hour UTC							
Minute UTC							
Second UTC							

Year 1..2 (16 bit)

Specify the year in the usual Gregorian calendar. The first byte (Year 1) is the most significant byte.

Month (8 bit)

Specify the month of the year between 01 (January) and 12 (December).

Day (8 bit)

Specify the day of the month between 01 and 31.

Hour UTC (8 bit)

Specify the number of complete hours that have passed since midnight (00-23) in UTC.

Minute UTC (8 bit)

Specify the number of complete minutes that have passed since the start of the hour (00-59) in UTC. Minutes are measured in Universal Time (UTC).

Second UTC (8 bit)

Specify the number of complete seconds since the start of the minute (00-59) in UTC. Seconds are measured in Universal Time (UTC).

3.76.2 Time Parameters Get Command

The Time Parameters Get Command is used to request date and time parameters.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_TIME_PARAMETERS							
Command = TIME_PARAMETERS_GET							

3.76.3 Time Parameters Report Command

The Time Parameters Report Command can be requested by the Time Parameters Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_TIME_PARAMETERS							
Command = TIME_PARAMETERS_REPORT							
Year 1							
Year 2							
Month							
Day							
Hour UTC							
Minute UTC							
Second UTC							

Refer to description under the Time Parameters Set Command.

3.77 User Code Command Class, version 1

The purpose of the User Code Command Class is to supply a Z-Wave™ enabled Door Lock Device with a command class to manage user codes.

3.77.1 User Code Set Command

The User Code Set Command is used to set a User Code in the device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_USER_CODE							
Command = USER_CODE_SET							
User Identifier							
User ID Status							
USER_CODE1							
..							
USER_CODEn							

User Identifier (8 bits)

The User Identifier is used to recognise the user identity. The User Identifier values must be a sequence starting from 1. This field can be ignored in case the node only supports one User Code.
Setting the User Identifier to 0 will address all User Identifiers available in the device.

User ID Status

The User ID Status field indicates the state of the User Identifier. All other values not mentioned in below list are reserved for future implementation.

Hexadecimal	Description
0x00	Available (not set)
0x01	Occupied
0x02	Reserved by administrator
0xFE	Status not available

USER_CODE1…USER_CODEn

These fields contain a list of node User Code. Minimum code length is 4 and maximum 10 ASCII digits. If set to zero's then the User Identifier has not been set.

3.77.2 User Code Get Command

The User Code Get Command is used to request the user code of a specific user identifier.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_USER_CODE							
Command = USER_CODE_GET							
User Identifier							

User Identifier (8 bits)

See description for USER_CODE_SET, 0 is not allowed from USER_CODE_GET.

3.77.3 User Code Report Command

The User Code Report Command can be used by e.g. a door lock device to send a report either unsolicited or requested by the User Code Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_USER_CODE							
Command = USER_CODE_REPORT							
User Identifier							
User ID Status							
USER_CODE1							
..							
USER_CODEn							

See parameter description for USER_CODE_SET.

3.77.4 Users Number Get Command

The User Number Get Command is used to request the number of USER CODES that this node supports.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_USER_CODE							
Command = USERS_NUMBER_GET							

3.77.5 Users Number Report Command

The Users Number Report Command is used to report the maximum number of USER CODES the given node supports. The Users Number Report Command can be send requested by the Users Number Get Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_USER_CODE							
Command = USERS_NUMBER_REPORT							
Supported Users							

Supported Users (8 bits)

The number of User Codes this node supports. ‘0’ indicates User Code is not supported by the device.

3.78 Version Command Class, version 1

This Version Command Class is 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.

NOTE: A device supporting a Command Class having a version higher than 1 must support the Version Command Class to be able to identify the supported version. In case the device doesn't support the Version Command Class then it can be assumed that all command classes are equal to version 1.

3.78.1 Version Get Command

The Version Get Command can be used to get the library type, protocol version and application version from a device that supports the Version Command Class. How to obtain the type and protocol version of the library used by the application is described in [2].

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_VERSION							
Command = VERSION_GET							

3.78.2 Version Report Command

The Version Report Command can be used to report the library type, protocol version and application version from a device. The Version Report Command can be sent unsolicited or requested by the Version Get Command.

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 bit)

Returns the Z-Wave Library Type i.e. it are a controller library, slave library, installer library etc. Refer to ZW_basis_api.h source code file for a list of the possible library types and the assigned values.

Z-Wave Protocol Version (8 bit)

Returns the Z-Wave Protocol Version of the used library and can have values in the range 0 to 255. In the table below are the Z-Wave Protocol version listed for a given Developer's Kit version :

Developer's Kit Version	Z-Wave Protocol Version	Z-Wave Protocol Sub Version
5.02 Patch 2	2	64
5.02 Patch 1	2	51
5.02	2	48
5.01	2	36
5.00 Patch 1	2	22
5.00 Beta 1	2	16
4.50 Beta 1	2	74
4.30 Beta 1	2	30
4.28	2	67
4.27	2	40
4.26	2	32
4.25	2	31
4.24 Patch 1	2	28
4.24	2	24
4.23	2	17
4.22	2	09
4.21	2	06
4.20	1	97
4.11	1	91
4.10	1	78
4.07	2	27
4.06	2	23
4.05	2	07
4.04	1	99
4.03	1	81
4.02	1	69
4.01	1	68
4.00	1	59
3.40	1	53
3.31	1	44
3.30	1	37
3.22	1	39
3.21	1	25
3.20	1	21

Table 5, Z-Wave Protocol version for a given Developer's Kit version

Warning: Products can only be Z-Wave certified based on matured versions of the Z-Wave Protocol, i.e. Developer's Kit having versions different from x.y0.

Z-Wave Protocol Sub Version (8 bit)

Returns the Z-Wave Protocol Sub Version of the used library and can have values in the range 0 to 255.

Application Version (8 bit)

Returns the Application Version and can have values in the range 0 to 255. The manufacturer assigns the Application Version.

Application Sub Version (8 bit)

Returns the Application Sub Version and can have values in the range 0 to 255. The manufacturer assigns the Application Sub Version.

3.78.3 Version Command Class Get Command

The Version Command Class Get Command can be used to request the individual command class versions from a device. Only versions from the command classes shown in the NIF can be requested. It's not possible to get a version number for the Generic and Specific Device Classes.

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 bit)

The Request Command Class field specifies which command class identifier is being requested.

3.78.4 Version Command Class Report Command

The Version Command Class Report Command can be 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 bit)

The Requested Command Class field specifies what command class the returned version belongs to.

Command Class Version (8 bit)

Returns the Command Class Version and can have values in the range 1 to 255. It starts with 1 and is incremented every time a new version of the Command Class is released. In case the requested command class is not present in the NIF then Command Class Version is set to zero. Refer to ZW_classcmd.h source code file for actual version number.

3.79 Wake Up Command Class, version 1

The Wake Up Command Class version 1 allows battery-operated devices to wake up occasionally and notify another device (always listening), that the device is ready to receive any queued commands.

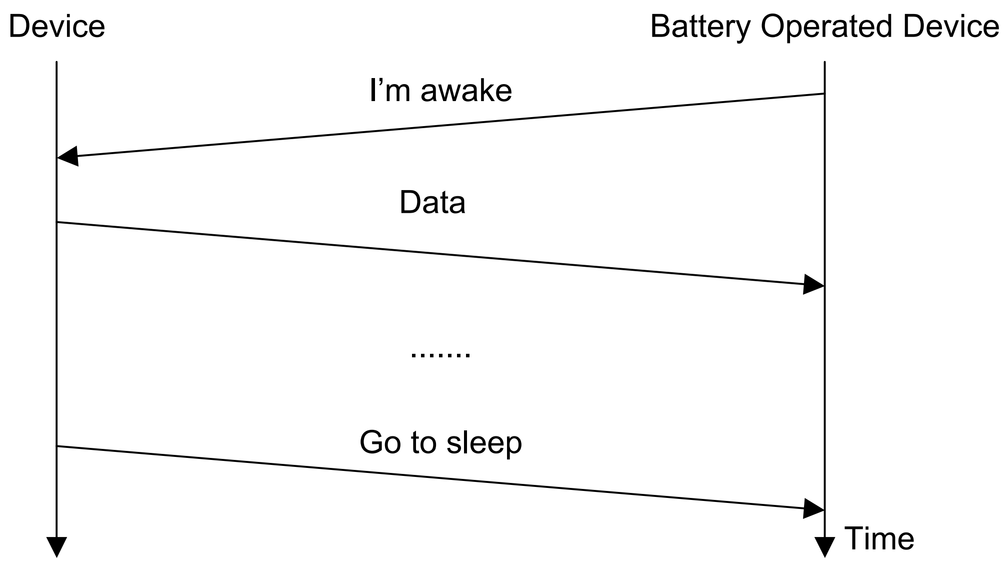


Figure 17, Wake Up sequence

Since this Command Class is normally used by battery operated devices it is required that Wake Up Notification commands are handled immediately and response (or data) is sent as soon as possible, in order to minimize battery consumption.

3.79.1 Wake Up Interval Set Command

The Wake Up Interval Set Command is used to configure the wake up interval of a device and the node ID 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							

Seconds 1 … 3 (24 bit)

The Seconds field contains the number of seconds between wake up of a battery-operated device. The first byte is the most significant byte.

In case number of seconds is set to 0 then wake up is initiated by an event determined by the application e.g. a pushbutton activation.

NodeID (8 bit)

The NodeID field contains the node ID of the device receiving the Wake Up Notification Command.

3.79.2 Wake Up Interval Get Command

The Wake Up Interval Get Command is used to request the wake up interval of a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_WAKE_UP							
Command = WAKE_UP_INTERVAL_GET							

3.79.3 Wake Up Interval Report Command

The Wake Up Interval Report Command is used to report the wake up interval of a device and the node ID of the device receiving the Wake Up Notification Command. The Wake Up Interval Report Command can be sent unsolicited or requested by the Wake Up Interval Get Command.

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 1 … 3 (24 bit)

The Seconds field contains the number of seconds between wake up of a battery-operated device. The first byte is the most significant byte.

NodeID (8 bit)

The NodeID field contains the ID on the node that should receive the Wake Up Notification Command.

3.79.4 Wake Up Notification Command

Devices will use the Wake Up Notification Command to notify other devices, that it is awake. Devices will normally start a timer that will force the device to power down after some time in case the Wake Up No More Information Command fails. This time shall be sufficient for the receiver to receive the command, check if any information is pending and send response.
A device is allowed to send the Wake Up Notification Command as broadcast as long as a node ID is not configured by Wake Up Interval Set Command. If a Wake Up Notification Command broadcast is received then it is not allowed to respond with a Wake Up No More Information Command before the node ID is configured by Wake Up Interval Set Command.

Please note that if the battery operated device wishes to send an unsolicited report, then it should be done before sending the Wake Up Notification Command. Otherwise, the Wake Up No More Information reply is likely to collide with the unsolicited report.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_WAKE_UP							
Command = WAKE_UP_NOTIFICATION							

3.79.5 Wake Up No More Information Command

The Wake Up No More Information Command is used by devices to notify the sender of the Wake Up Notification Command that it should not expect any more information, and consequently it can go back to sleep to minimize power consumption.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_WAKE_UP							
Command = WAKE_UP_NO_MORE_INFORMATION							

3.80 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. Version 2 comprises of all the version 1 commands and two new commands to enable this feature.

3.80.1 Wake Up Interval Capabilities Get Command

The Wake Up Interval Capabilities Get Command is used to request the wake up interval capabilities of a device.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_WAKE_UP							
Command = WAKE_UP_INTERVAL_CAPABILITIES_GET							

3.80.2 Wake Up Interval Capabilities Report Command

The Wake Up Interval Capabilities Report Command is used to report the wake up interval capabilities of a device. The Wake Up Interval Capabilities Report Command can be send requested by the Wake Up Interval Capabilities Get Command.

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 Byte 1							
Minimum Wake Up Interval Seconds Byte 2							
Minimum Wake Up Interval Seconds Byte 3							
Maximum Wake Up Interval Seconds Byte 1							
Maximum Wake Up Interval Seconds Byte 2							
Maximum Wake Up Interval Seconds Byte 3							
Default Wake Up Interval Seconds Byte 1							
Default Wake Up Interval Seconds Byte 2							
Default Wake Up Interval Seconds Byte 3							
Wake Up Interval Step Seconds Byte 1							
Wake Up Interval Step Seconds Byte 2							
Wake Up Interval Step Seconds Byte 3							

Byte 1 is the most significant byte for all of the following fields.

Minimum Wake Up Interval Seconds 1 … 3 (24 bit)

This field specifies the minimum wake up interval a battery-operated device supports.

The following values are possible:

Decimal	Description
0	No minimum / maximum / default wake up interval, the battery-operated device is activated by e.g. user interaction in form of a button press on the battery-operated device. If this field is 0, then all the other fields must also be 0. **Note**: This is identical to the specified behavior in Version 1 of the command class.
1 … 16777215	The minimum wake up interval in seconds supported by the battery-operated device.

Maximum Wake Up Interval Seconds 1 … 3 (24 bit)

This field defines the maximum wake up interval a battery-operated device supports.

The following values are possible:

Decimal	Description
0	No minimum / maximum / default wake up interval, the battery-operated device is activated by e.g. user interaction in form of a button press on the battery-operated device. If this field is 0, then all the other fields must also be 0. **Note**: This is identical to the specified behavior in Version 1 of the command class.
1 … 16777215	The maximum wake up interval in seconds supported by the battery-operated device. This interval must never be lower than the minimum wake up interval, but it can be equal, which means the device only supports one interval.

Default Wake Up Interval Seconds 1 … 3 (24 bit)

This field defines the default wake up interval a battery-operated device supports.

The following values are possible:

Decimal	Description
0	No minimum / maximum / default wake up interval, the battery-operated device is activated by e.g. user interaction in form of a button press on the battery-operated device. If this field is 0, then all the other fields must also be 0. **Note**: This is identical to the specified behavior in Version 1 of the command class.
1 … 16777215	The default wake up interval in seconds supported by the battery-operated device. This interval must never be lower than the minimum wake up interval or higher than the maximum wake up interval.

Wake Up Interval Step Seconds 1 … 3 (24 bit)

This field defines the resolution of possible wake up intervals, which a battery-operated device supports.

The following values are possible:

Decimal	Description
0	No interval steps are possible. The battery-operated device only supports the minimum and maximum wake up interval. The interval step must be set to 0 if both the maximum and the minimum interval are equal.
1 … 16777215	The wake up interval step in seconds supported by the battery-operated device. This interval must not exceed the difference between the minimum and maximum wake up interval. The interval steps should have a length so the difference between the maximum and minimum Wake Up Interval is a multiple of the interval steps. *Examples:* If a device has minimum wake up interval of 5 minutes (300 seconds) and a maximum wake up interval of 10 minutes (600 seconds), then 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. If a device has minimum wake up interval of 5 minutes (300 seconds) and a maximum wake up interval of 10 minutes (600 seconds) and wake up interval step of 100 seconds you may only set the following intervals on the device: 300 seconds 400 seconds 500 seconds 600 seconds

3.81 Z/IP Tunneling Client Command Class, version 1

The Z/IP Tunneling Client Command Class covers commands that a Z/IP node must be able to receive and respond to.

3.81.1 Network Management Commands

3.81.1.1 Z/IP Gateway Set Command

Z/IP Gateway → Z/IP Node

The Z/IP Gateway Set command can be used to configure the default gateway entry in a Z/IP node.

The Z/IP node must direct all IP packets outside the local subnet to the default gateway unless additional IP route entries explicitly point at other subnets.
The creation of IP route entries in a Z/IP node is outside the scope of this specification.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_TUN_CLIENT							
Command = COMMAND_ZIP_GATEWAY_SET							
Gateway Node ID							

Gateway node ID (8 bit)

Table 6, Gateway Node ID values

Gateway Node ID	Value
Valid values	1-232
Invalidate entry	0

Gateway node ID specifies the node ID of the Z/IP gateway. In a multi-controller environment, the primary controller including a Z/IP node may inform the node being included that some other node provides the Z/IP gateway services.
Thus, the node sending the `Z/IP Gateway Set` command is not necessarily the Z/IP gateway.
A Z/IP node receiving a zero Node ID must clear the default gateway entry and at the same time clear the current IP subnet information.
A Z/IP Gateway must send the `Z/IP Gateway Set` command to all nodes being included. Older nodes may ignore this message. Z/IP nodes supporting the tunneling or IP transport must accept this message and store the information internally.

A Z/IP node included by a classic controller not supporting Z/IP may issue a `Z/IP Gateway Find` command for locating the Z/IP gateway. In advanced systems, multiple Z/IP gateways may respond by returning a `Z/IP Gateway Set` command. In case of multiple responding Z/IP gateways, the receiving node must determine which gateway provides the best fit.

3.81.1.2 Z/IP Gateway Get Command

Z/IP Gateway → Z/IP Node

The Z/IP Gateway Get Command can be used to read back the default gateway from a Z/IP node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_TUN_CLIENT							
Command = COMMAND_ZIP_GATEWAY_GET							

(no payload is carried in this command)

3.81.2 Tunnel Commands

A Z/IP tunnel constitutes a permanent link between the Z/IP gateway and a Z/IP node and may be used for exchanging application data between an application running in the Z/IP node with very limited memory resources and an Internet server.

3.81.2.1 Z/IP Tunnel Status Command

The Z/IP Tunnel Status Command is used for maintenance functions related to Z/IP tunnels.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_TUN_CLIENT							
Command = COMMAND_ZIP_TUNNEL_STATUS							
Handle							
Status							

A handle is returned to use for the rest of the tunnel lifetime.
If a node sends data into a tunnel that does not exists or other problems arise, the Z/IP gateway may return a `Z/IP Tunnel Status` message indicating the type of error.

The `Z/IP Tunnel Status` message may also indicate that the tunnel is being created but that some additional information is needed, e.g. a URL.

Handle (8 bits)

The `Handle` identifies the tunnel during the entire life time of the tunnel.

Status (8 bits)

The `Status` may be used to specify the status of the tunnel.

Parameter	Size	Usage	Comment
Handle	1 byte	Unique handle	Assigned by Z/IP Gateway
Status	1 byte	Enumerated: 0: NewTunnelCreated 1: ERROR 2: TunnelAlreadyExists 3: NoTunnelExists 4: TunnelWasClosed 5: NoLanConnection 6: UrlCouldNotBeResolved 7: IpAddressNoResponse 16: NewTunnelWaitingForUrl	All non-zero values are codes indicating that the tunnel is not fully operational.

3.82 Z/IP Tunneling Server Command Class, version 1

The Z/IP Tunneling Server Command Class covers commands that a Z/IP node must be able to receive and respond to.

3.82.1 Network Management Commands

3.82.1.1 Z/IP Gateway Report Command

Z/IP Gateway ← Z/IP Node

The Z/IP Gateway Report Command can be sent as a result of receiving a Z/IP Gateway Get command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_TUN_SERVER							
Command = COMMAND_ZIP_GATEWAY_REPORT							
Gateway Node ID							

Gateway Node ID (8 bit)

Specify the Z/IP gateway node ID currently held in the Z/IP node.

3.82.1.2 Z/IP Gateway Find Command

Z/IP Gateway ← Z/IP Node

The Z/IP Gateway Find Command can be used by a Z/IP node to identify Z/IP gateways available in the Z/IP network. IP packets destined for IP addresses outside the local subnet must be forwarded to a Z/IP gateway. The `Z/IP Gateway Find` command should be sent to the Z-Wave multicast address.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_TUN_SERVER							
Command = COMMAND_ZIP_GATEWAY_FIND							

A Z/IP node included by a classic controller not supporting Z/IP may issue a `Z/IP Gateway Find` command for locating the Z/IP gateway.

In advanced systems, multiple Z/IP gateways may respond by returning a `Z/IP Gateway Set` command. In case of multiple responding Z/IP gateways, the receiving node must determine which gateway to use.

3.82.2 Tunnel Commands

A Z/IP tunnel constitutes a permanent link between the Z/IP gateway and a Z/IP node and may be used for exchanging application data between an application running in the Z/IP node with very limited memory resources and an Internet server.

3.82.2.1 Z/IP Tunnel Create Command

The Z/IP Tunnel Create Command can be used to create a tunnel.

Tunnel type == 'HTTP' / 'HTTPS':

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_TUN_SERVER							
Command = COMMAND_ZIP_TUNNEL_CREATE							
Tunnel type = 'HTTP' or “HTTPS”							
Reserved	Wait for response	IPv6	Destination URL	*Reserved*			
IP Destination address 1							
...							
IP Destination address 16							

Refer to sections 3.83.2 for datagram formatting.

`Z/IP Tunnel Create (HTTP or HTTPS)` is used by Z/IP nodes to create a point-to-point connection through the Z/IP infrastructure. The HTTP tunnel type extends into the IP domain. Carrying data in HTTP traffic is a proven method for bypassing firewalls.

When a Z/IP node delivers a tunnel datagram to the Z/IP gateway, the gateway initiates an HTTP POST message carrying the datagram as an embedded binary object in an XML structure.

Tunnel type (8 bits)

The `Tunnel type` signals the header format used.

Tunnel type	Name	Parameters and their use		
80 (0x50)	HTTP	TunnelType = 80 Flags DestinationAddress	- byte - byte - 16 bytes	- Tunnel type - Signaling modes, etc. - Where to send data
187 (0xBB)[2]	HTTPS	TunnelType = 187 Flags DestinationAddress	- byte - byte - 16 bytes	- Tunnel type - Signaling modes, etc. - Where to send data

[2] The HTTPS port number is 443 (0x1BB). Typecasted to one byte, this becomes 187 (0xBB).

Wait for response (1 bit)

The `Wait for response` flag may be used to signal that the Z/IP gateway should keep the TCP connection open. This will allow response data to find its way back to the node originating the tunneled data.

Wait for response	Value
Wait for HTTP response	'1'
Close after sending HTTP request	'0'

IPv6 (1 bit)

The `IPv6` flag may be used to signal that the Z/IP gateway should use IPv6 for the TCP connection.

IPv6	Value
Use IPv6 packet format	'1'
Use IPv4 packet format	'0'

Destination URL (1 bit)

The `Destination URL` flag may be used to signal that a URL is to be specified as part of the tunnel creation. The Z/IP gateway must request the URL by signaling the status value `NewTunnelWaitingForUrl` in the `Z/IP Tunnel Status` command returned in response to the `Z/IP Tunnel Create` command.

Destination URL	Value
A URL will be specified afterwards	'1'
Use the provided IP address	'0'

IP Destination Address (16 bytes)

The IP `Destination Address` may be used to specify the target host of the tunneled data.

If IPv6 is not specified, i.e. IPv4 is selected, only use the first 4 bytes of the IP Destination address.

Reserved

Reserved bits must be set to zero.

3.82.2.2 Z/IP Tunnel Close Command

The Z/IP Tunnel Close Command is used for closing Z/IP tunnels.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_TUN_SERVER							
Command = COMMAND_ZIP_TUNNEL_CLOSE							
Handle							

A `Z/IP Tunnel Status` message is used to signal `TunnelWasClosed`.

Handle (8 bits)

The `Handle` identifies the tunnel during the entire life time of the tunnel.

3.83 Z/IP Tunneling Services Command Class, version 1

The Z/IP Tunneling Services Command Class can be used to carry Z-Wave commands between IP hosts and classic Z-Wave nodes as well as Z/IP nodes.

3.83.1 Z/IP Packet Command

In an IP environment, Z-Wave commands are carried in a Z/IP Packet Command.
A Z/IP Packet may be carried in the payload of a UDP frame (LAN version), in HTTP transport frames (dial-out remote access) or via plain TCP connections (dial-in remote access).

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_TUN_SERVICES							
Command = COMMAND_ZIP_PACKET							
Ack Request	Ack Response	Nack Response	Waiting Response	Web KeepAlive Request	Header ext. no support Response	*Reserved*	Use no Z-Wave ack
Home ID Included	Node IDs Included	Gateway MAC Included	Customer ID Included	Web KeepAlive delay Included	Header ext. included	*Reserved*	Z-Wave Cmd Included
Z-Wave home ID 1					(Optional)		
...					(Optional)		
Z-Wave home ID 4					(Optional)		
Z-Wave source node ID					(Optional)		
Z-Wave destination node ID					(Optional)		
Z/IP gateway MAC address 1					(Optional)		
...					(Optional)		
Z/IP gateway MAC address 6					(Optional)		
Customer ID 1					(Optional)		
...					(Optional)		
Customer ID 8					(Optional)		
Web KeepAlive delay					(Optional)		
Header extension 1					(Optional)		
...					(Optional)		
Header extension N					(Optional)		
Z-Wave command 1					(Optional)		
...					(Optional)		
Z-Wave command N					(Optional)		

Note: This Z-Wave command is a virtual command. Thus, the normal Z-Wave frame size limitation does not apply to this command.
The Z/IP Packet may be used for transport of encapsulated Z-Wave commands in an IP environment,

e.g. Z-Wave commands from a classic Z-Wave node to an IP host.
The command is also used for controlling classic Z-Wave resources from an IP host.

When receiving a Z/IP packet, the receiving Z/IP gateway or IP host must inspect the inclusion flags in order to determine the offset to use for accessing the optional fields

Ack Request (1 bit)

The `Ack Request` flag signals that the receiving end must return an Ack or Nack message in response to the actual Z/IP Packet.

Ack request	Value
Return Ack or Nack	'1'
No confirmation needed	'0'

This bit must only affect high-level acknowledgement processing for Z/IP packets. Z-Wave Acknowledgement should always be requested by a Z/IP gateway forwarding a Z-Wave command into the Z-Wave PAN.

Ack (1 bit)

The `Ack` flag signals that the target node received the Z/IP command carried in the Z/IP packet.

Ack	Value
Ack	'1'
(check Nack)	'0'

A Z/IP `Ack` or `Nack` packet must have the same `Seq No` value as the Z/IP packet being acknowledged. In case of retransmissions, multiple Acks may be received. Acks referring to `Seq No` values that are not waiting for an `Ack` must be ignored.

Nack (1 bit)

The `Nack` flag signals that the target node did not (yet) receive the Z/IP command carried in the Z/IP packet.

Nack	Value
Nack (check Waiting)	'1'
(ignore)	'0'

A Z/IP `Ack` or `Nack` packet must have the same `Seq No` value as the Z/IP packet being acknowledged. In case of retransmissions, multiple `Acks` may be received. Acks referring to `Seq No` values that are not waiting for an `Ack` must be ignored.

If the `Nack` flag is set, the `Waiting` flag must also be inspected.
If the `Nack` flag is set, the `Header extension no support` flag must also be inspected - provided that the sender used header extensions.

Waiting (1 bit)

The `Waiting` flag signals that the target Z-Wave node may have a long response time. The Z/IP gateway has not timed out yet, but is just informing the originating IP application that the Z/IP packet arrived correctly at the Z/IP gateway.
For frequently listening nodes, the waiting time may be up to one second
The `Waiting` flag is a companion flag. It should only be inspected if the `Nack` flag is true.

Waiting	Value
Waiting	'1'
(ignore)	'0'

Web KeepAlive Request (1 bit)

The `Web KeepAlive Request` flag signals to an Internet server that the Z/IP Gateway wants to open up a connection for remote access from the Internet server to the Gateway. The server must respond within 60 seconds after receiving the KeepAlive request.
If the Z/IP gateway does not receive a response within 75 seconds, the Z/IP Gateway must issue another KeepAlive Z/IP Packet.

Web KeepAlive Request	Value
KeepAlive Request	'1'
(ignore)	'0'

Header extension No Support (1 bit)

The receiver detected the presence of a `Header Extension` but the receiver does not support header extensions.

Header extension No Support	Value
Receiver cannot interpret extended header	'1'
(ignore)	'0'

Use no Z-Wave Ack (1 bit)

If enabled, the Z/IP gateway must send Z-Wave frames without Z-Wave acknowledgement request.
Use of this feature is discouraged. The default value during normal operation should be '0'.
The feature is only intended as a test feature. Unacknowledged, routed delivery of frames in a wireless environment introduces a risk of losing commands before they reach the target.

Use no Z-Wave Ack	Value
Do not request Z-Wave Ack	'1'
Request Z-Wave Ack	'0' (DEFAULT)

Home ID Included (1 bit)

The `Home ID Included` flag signals that Z-Wave homeID of the encapsulated Z-Wave frame is included in the Z/IP packet.

Home ID Included	Value
Home ID is included	'1'
Home ID NOT included	'0'

Node IDs Included (1 bit)

The `Node IDs Included` flag signals that Z-Wave source and destination nodeIDs of the encapsulated Z-Wave frame are included in the Z/IP packet.

Node IDs Included	Value
Src + Dest node IDs are included	'1'
Src + Dest node IDs NOT included	'0'

Gateway MAC Included (1 bit)

The `Gateway MAC Included` flag signals that the MAC address of the Z/IP gateway is included in the Z/IP packet.

Gateway MAC Included	Value
Z/IP gateway MAC address is included	'1'
Z/IP gateway MAC address NOT included	'0'

The gateway MAC address may be used by a service provider for authentication purposes.

Customer ID Included (1 bit)

The `Customer ID Included` flag signals that an 8 byte customer ID is included in the Z/IP packet. The customer ID may be used by a service provider for authentication purposes. The customer ID is maintained by the gateway application layer and should be accessible via the user interface.

Customer ID Included	Value
Customer ID is included	'1'
Customer ID NOT included	'0'

Web KeepAlive delay Included (1 bit)

The `Web KeepAlive delay Included` flag signals that an 8 byte delay size is included in the Z/IP packet. The delay is measured in seconds and indicates how long time the Z/IP gateway should wait before issuing another `Web KeepAlive Request` Z/IP Packet.

Web KeepAlive delay Included	Value
KeepAlive included	'1'
(ignore)	'0'

Header extension Included (1 bit)

The `Header Extension Included` flag signals that an extended header included in the Z/IP packet.

<u>The use of extended headers is reserved for future use.</u> Header extensions may be used for unforeseen application requirements that are not covered by the original definition the Z/IP packet header.
The value must be set to '0'.

The size of the extended header must be signaled in byte #0 of the extended header.
The version of the extended header must be signaled in byte #1 of the extended header.

Header extension Included	Value
Extended header included	'1'
(ignore)	'0' (DEFAULT)

A receiver not supporting extended headers should ignore the entire packet but return a Nack Z/IP packet carrying the `Header extension No Support` flag.

Reserved

Reserved bits must be set to zero.

Z-Wave Command Included (1 bit)

The `Z-Wave Command Included` flag signals that a Z-Wave command is included in the Z/IP packet.

Z-Wave Command Included	Value
Z-Wave command is included	'1'
Z-Wave command NOT included	'0'

3.83.2 Z/IP Tunnel Datagram Segment Command

The Z/IP Tunnel Datagram Segment Command can be used to carry a segment of a datagram.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_ZIP_TUN_SERVICES							
Command = COMMAND_ZIP_TUNNEL_DATAGRAM_SEGMENT							
Handle							
First segment == '1'	Last segment	Last datagram	Tunnel Mgmt	*reserved*		Sequence count	
Payload 1							
...							
Payload N							

Up to 46 payload bytes may be carried in a tunnel segment. The number of payload bytes transmitted can be determined from the length field in the frame.

Handle (8 bits)

The `Handle` identifies the tunnel during the entire life time of the tunnel.

First segment (1 bit)

The `First segment` flag signals that this is the first segment of a datagram.
It may be the last segment at the same time if the datagram is short enough to fit into one segment.

First segment	Value
First segment	'1'
Not first segment	'0'

Last segment (1 bit)

The `Last segment` flag signals that the actual segment terminates the datagram.

Last segment	Value
Last segment	'1'
Not last segment	'0'

Last datagram (1 bit)

The `Last datagram` flag signals that the actual datagram was the last datagram in the senders TX queue.

Last Datagram	Value
Last datagram	'1'
Not last datagram	'0'

Tunnel Mgmt (1 bit)

The `Tunnel Mgmt` flag signals that the datagram carries data related to the management of the tunnel.

Tunnel Mgmt	Value
Tunnel management	'1'
Normal data	'0'

Note: The following applies to the creation of a tunnel, i.e. the first Z/IP tunnel datagram sent by the Z/IP node after the Z/IP gateway has sent a Z/IP Tunnel Status with the code "NewTunnelWaitingForUrl":

If the datagram carries data related to the creation of the tunnel, byte #0 of the datagram must indicate the type of data.

Tunnel Mgmt type (Byte #0 of the management datagram)	Value
Reserved	0
URL	1
Reserved	2..255

For "URL" type management datagrams, the complete URL must follow in bytes #1..#n of the datagram. The URL must be encoded in plain ASCII. The URL string must start with "HTTP:" The URL string must be NULL terminated, i.e. a byte with the value of zero must follow the last character of the URL.

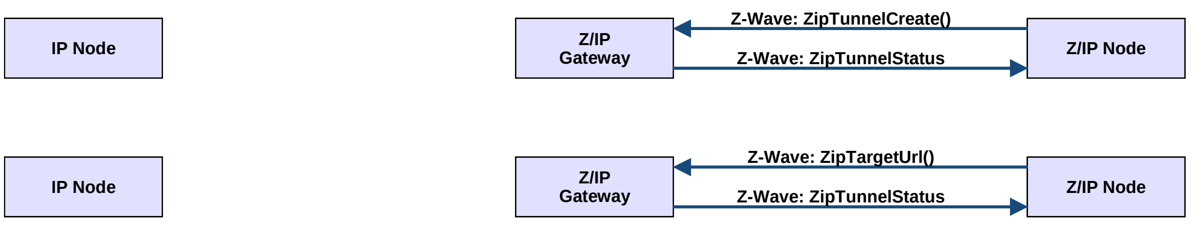


Figure 18, Creation of Z/IP tunnel

After successful reception of the URL management datagram, the Z/IP gateway may send another Z/IP Tunnel Status message with the code "NewTunnelCreated".

Sequence Count (2 bit)

The receiving end may receive multiple copies of a frame due to retransmissions caused by missing Acks. The `Sequence Count` field is a modulo 4 counter controlling the reception of datagram segments. The first segment of a segmented datagram must always hold the value '00'.

First segment	Value
First segment	'00'
Other segments	previousFrame.SeqCount + 1

Reserved

Reserved bits must be set to zero.

Payload

The payload carried in a Z/IP Tunnel datagram must be formatted as a Z-Wave command.

This serves two purposes:

1) The Z/IP gateway stays transparent to flows passing the gateway.

2) A node running an IP networked application may exchange proprietary data with an Internet server and at the same time receive standard Z-Wave control commands from that same server. This feature allows a Z/IP node to “phone home” to a third-party service provider and receive Z-Wave commands without setting any remote access parameters in the Z/IP Gateway.

In case networked applications exchange manufacturer proprietary data, data must be carried in a `PROPRIETARY` Z-Wave command.

A Manufacturer ID must be requested from Zensys as part of the certification process.
Refer to Zensys document SDS10473 / PN903103501 for full documentation of the `PROPRIETARY` command class.

APPENDIX A MANUFACTURER IDS

Customer	Manufacturer ID
2B Electronics	0x0028
3e Technologies	0x002A
A-1 Components	0x0022
ACT - Advanced Control Technologies	0x0001
AEON Labs	0x0086
Asia Heading	0x0029
Aspalis	0x005D
Atech	0x002B
Balboa Instruments	0x0018
BeSafer	0x002C
Boca Devices	0x0023
Broadband Energy Networks Inc.	0x002D
BuLogics	0x0026
Carrier	0x002E
CasaWorks	0x000B
Color Kinetics Incorporated	0x002F
ControlThink LC	0x0019
ConvergeX Ltd.	0x000F
Cooper Wiring Devices	0x001A
Cyberhouse	0x0014
Cytech Technology Pre Ltd.	0x0030
Danfoss	0x0002
Destiny Networks	0x0031
Digital 5, Inc.	0x0032
Eka Systems	0x0087
Electronic Solutions	0x0033
El-Gev Electronics LTD	0x0034
ELK Products, Inc.	0x001B
Embedit A/S	0x0035
Everspring	0x0060
Exceptional Innovations	0x0036
Exhausto	0x0004
Fakro	0x0085
Foard Systems	0x0037
HiTech Automation	0x0017
Home Automated Inc.	0x005B
Home Automated Living	0x000D
Home Director	0x0038
Homepro	0x0050
HomeSeer Technologies	0x000C
Honeywell	0x0039
Horstmann Controls Limited	0x0059
iCOM Technology b.v.	0x0011
Inlon Srl	0x003A

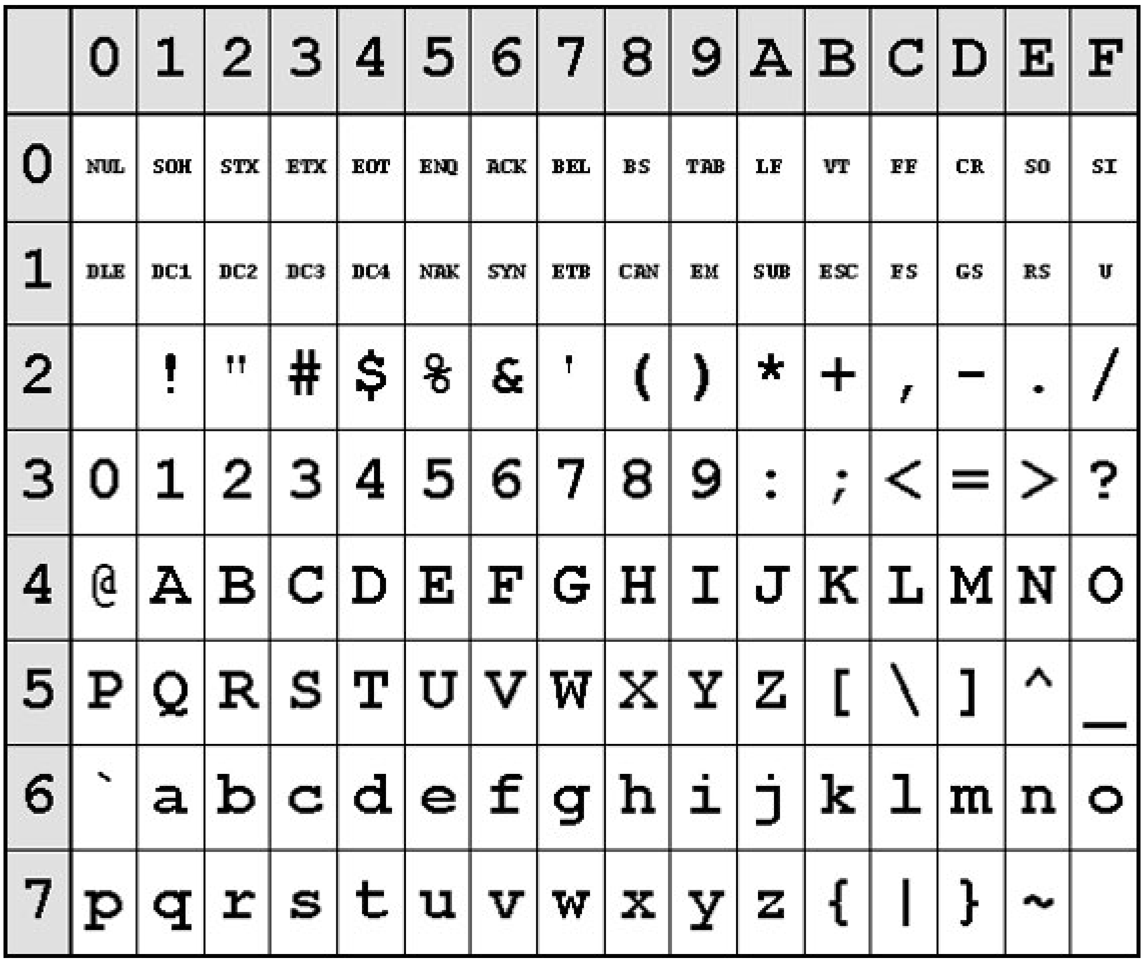
Customer	Manufacturer ID
INNOVUS	0x0077
Intel	0x0006
IntelliCon	0x001C
Intermatic	0x0005
Internet Dom	0x0013
IR Sec. & Safety	0x003B
Lagotek Corporation	0x0051
Leviton	0x001D
Lifestyle Networks	0x003C
Logitech	0x007F
Loudwater Technologies, LLC	0x0025
LS Control	0x0071
Marmitek BV	0x003D
Martec Access Products	0x003E
Merten	0x007A
Monster Cable	0x007E
Motorola	0x003F
MTC Maintronic Germany	0x0083
Novar Electrical Devices and Systems (EDS)	0x0040
OpenPeak Inc.	0x0041
PowerLynx	0x0016
Pragmatic Consulting Inc.	0x0042
Reitz-Group.de	0x0064
Residential Control Systems, Inc. (RCS)	0x0010
RS Scene Automation	0x0065
Ryherd Ventures	0x001E
Scientia Technologies, Inc.	0x001F
Seluxit	0x0069
Senmatic A/S	0x0043
Sequoia Technology LTD	0x0044
Sine Wireless	0x0045
Smart Products, Inc.	0x0046
Somfy	0x0047
Sylvania	0x0009
Techniku	0x000A
Tell It Online	0x0012
Telsey	0x0048
Trane Corporation	0x008B
Tricklestar Ltd.	0x006B
Twisthink	0x0049
Universal Electronics Inc.	0x0020
Vero Duco	0x0080
ViewSonic Corporation	0x005E
Vimar CRS	0x0007
Visualize	0x004A
Watt Stopper	0x004B

Customer	Manufacturer ID
Wayne Dalton	0x0008
Woodward Labs	0x004C
Wrap	0x0003
Xanboo	0x004D
Zdata, LLC.	0x004E
Zensys	0x0000
Z-Wave Technologia	0x004F
Zykronix	0x0021

Table 7, Manufacturer ID values

APPENDIX B ASCII CODES

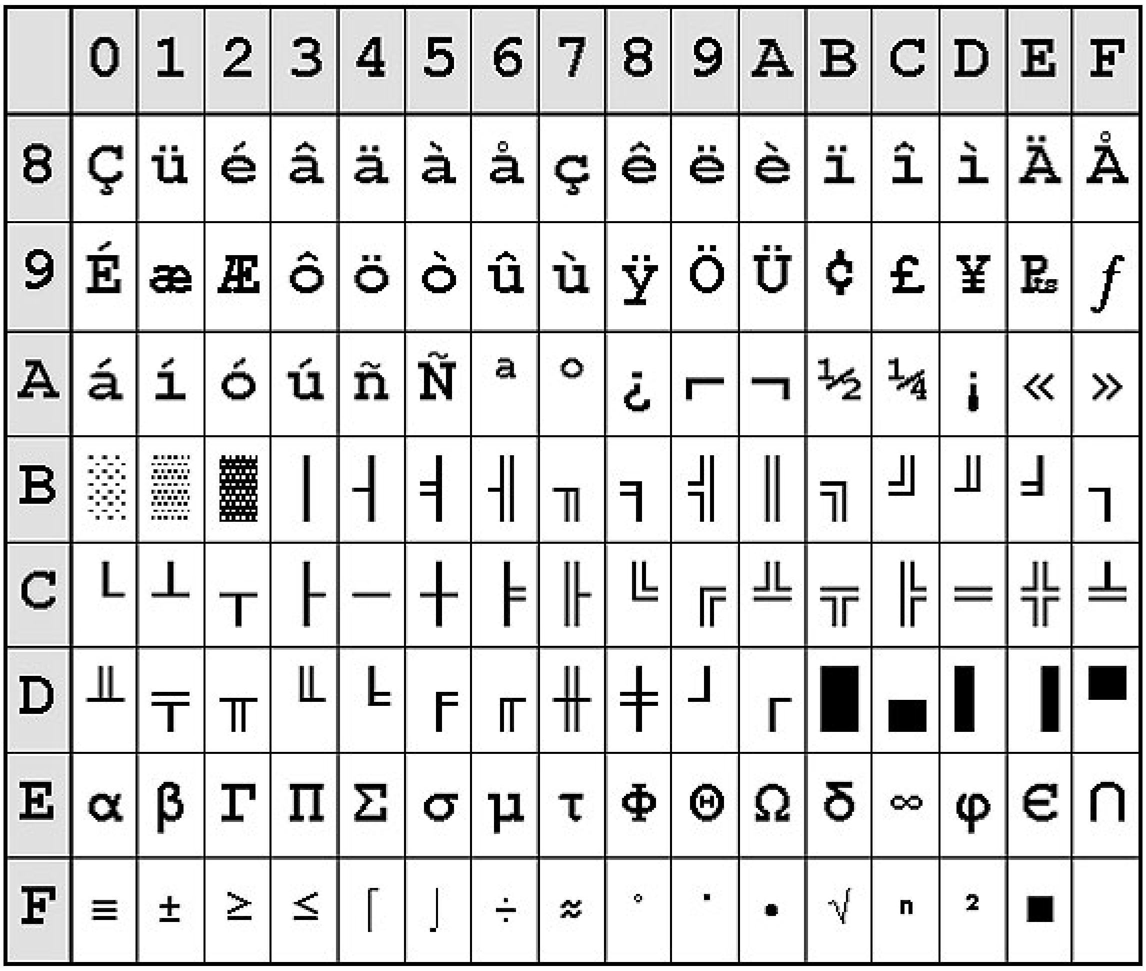
The standard ASCII table defines 128 character codes (from 0 to 127), of which, the first 32 are control codes (non-printable), and the remaining 96 character codes are printable characters. The table below shows the hexadecimal values of the ASCII character codes, e.g. the ASCII code for the capital letter "A" is equal to 0x41:



	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	NAK	SYN	ETB	CAN	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	{	|	}	~	

Table 8, The standard ASCII Table

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:



	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	¢	£	¥	₧	ƒ
A	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
B	░	▒	▓	│	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
C	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
D	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	█	▄	▌	▐	▀
E	α	β	Γ	Π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	φ	∈	∩
F	≡	±	≥	≤	⌠	⌡	÷	≈	°	·	•	√	^{n}	^{2}	■	

Table 9, OEM Extended ASCII Table

Below are listed codes for players, radios etc. as an alternative to the OEM Extended ASCII codes. Undefined values must be ignored.

	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

Table 10, Players Table

REFERENCES

[1] Zensys, SDS10242, Z-Wave Device Class Specification.
[2] Zensys, INS10247, Z-Wave ZW0102/ZW0201/ZW0301 Application Programming Guide.
[3] Zensys, APL10720, Programming the ZW0201 Flash from Internal MCU.
[4] Zensys, SDS10865, Software Design Spec., Z-Wave Application Security layer specification

INDEX