Software Design Specification

Z-Wave Transport-Encapsulation Command Class Specification

Document No.:	SDS13783
Version:	14
Description:	The document describes the Z-Wave Command Classes and associated Commands used by Z-Wave enabled products for transport or encapsulation.
Written By:	JFR;NOBRIOT;BBR;DEWASSIE
Date:	2020-07-06
Reviewed By:	NOBRIOT;DEWASSIE;JBU
Restrictions:	Public

Approved by:				
Date	CET	Initials	Name	Justification
2020-07-06	03:54:15	NTJ	Niels Johansen	

SILICON LABS

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20161015	NOBRIOT	ALL 3.3 3.6 3.7	Transferred content from the previous Command Class documents Integrated approved content from the 2016C Open Review: • Clarified Multi Command Command Class regarding extended Command Class identifiers Added Security 2 (S2) Command Class, version 1 Added Supervision Command Class, version 1
2	20170403	NOBRIOT	3.6 3.5 3.1	Integrated approved content from the 2017A Open Review: • Updated Security 2 Command Class regarding Multicast MOS situation, use of CSA and removed physical key exchange requirement • Updated and clarified Security 0 Command Class • Deprecated CRC-16 Command Class
3	20170703	NOBRIOT	3.3 3.6 3.6.6.2	Integrated approved content from the 2017B Open Review: • Clarified Multi Channel Command Class regarding what controlling nodes should do with dynamic End Points. • Added minor editorial adjustments to the Security 2 Command Class. Moved DSK format and representation requirements to [8]
4	20171002	NOBRIOT	3.3 3.6 3.1	Integrated approved content from the 2017C Open Review: • Obsoleted “Answer as asked” requirement for the Multi Command Command Class. • Added explicit timeout tolerances for Security 2 Command Class • Changed CRC-16 and Supervision Command Class to be always advertised in the NIF (i.e. supported non-securely)
5	20180110	NOBRIOT	3.7 3.6 3.2 3.1	Integrated approved content from the 2017D Open Review: • Clarified special cases for Supervision Command Class and adding examples • Changed Security 2: a node may request keys in any order during the key exchange. • Refactored Multi Channel Command Class version 3 and removed dynamic End Point considerations • Improved CRC-16 Command Class text.
6	20180305	BBR	All	Added Silicon Labs template
7	20180409	NOBRIOT	3.3 3.2 3.5.4.1 3.6.6.1	Integrated approved content from the 2018A Open Review: • Rewritten the Multi Channel Command Class, version 4 • Clarified how to use End Point Capability Report with S2 in Multi Channel version 3 • Added S0 indication about a requirement that has not been observed my most implementations • Moved a requirement from the Device Type [8] to the S2 Command Class about S0 key exchange and message encapsulation.
8	20180701	NOBRIOT	3.6.8.1	Contributions 2018B: • Clarified requirement regarding Command Class support for S2 nodes receiving S0 Commands
9	20190101	DEWASSIE NOBRIOT	2 Figure 19 3.6.5.3.3.1.2 3.6.5.3.3	Contributions to 2018D: • Added requirement numbers for generic command class section • Added missing Nonce Get/Report on S2 bootstrapping frame flow • Clarified a requirement regarding setting Group ID value in MPAN Extension • Clarified how to use sequence number in S2 encapsulation
10	20190401	DEWASSIE NOBRIOT	0 3.6.6.4.2	Contributions to 2019A contributions for Open Review: • Added clarifications for Remove type commands in Supervision Command Classes • Relaxed TB1 to be in the 10..30 seconds range for nodes supporting S2.
11	20190701	NOBRIOT	0	Contributions 2019B: • Updated old requirement about encapsulation and support of the corresponding Command Class

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
12	20191001	DEWASSIE	3.6.6.4	Contribution 2019C: • Clarified a Client-Side authentication requirement during S2 bootstrapping
13	20200101	DEWASSIE	3.2.1 3.6.5 0	Contribution 2019D: • Added a list of command classes that should only be supported by the Root Device of a Multi-Channel Device • Allowed FLiRS nodes in S2 multicast groups • Added Supervision Command Class, version 2
14	20200701	NOBRIOT	3.6.6.2 3.7.3 3.5.4.3	Contributions 2020A: • Added a new Authenticated / Unauthenticated ECDH key pair requirement • Updated Supervision SessionID duplicate requirement, it now applies only for non-secure frames. • Updated the S0 Supported Commands Report to not contain non-secure Command Classes

Table of Contents

Table of Figures

Table of Tables

Abbreviations

Abbreviation	Explanation
|	Symbol used to denote the concatenation of information fields
AAD	Additional Authenticated Data
AES	Advanced Encryption Standard
CCM	Counter with CBC-MAC
CKDF	Cipher-based Key Derivation Function
CMAC	Cipher-based Message Authentication Code
CSA	Client-side Authentication
CTR_DRBG	Counter mode Deterministic Random Byte Generator
DSK	Device Specific Key
DSK-KDF	DSK-Key Derivation Function
EI	Entropy Input (used for seeding a new CTR_DRBG)
ECDH	Elliptic Curve Diffie-Hellman
IV	Initialization vector
KEX	Key exchange
LSB	Least Significant Byte
MAC	Message Authentication Code
MGRP	Multicast Group
MITM	Man-In-The-Middle (security attack)
MOS	Multicast Out Of Sync
MPAN	Multicast Pre-Agreed Nonce
MSB	Most Significant Byte
N/A	Not Applicable
Nonce	Number used once
NWI	Network-Wide Inclusion
OOB	Out-Of-Band (security authentication method)
PRK	Pseudo Random Key
PRNG	Pseudo-Random Number Generator
QR	Quick Response code
REI	Receiver's Entropy Input
S0	Security 0 Command Class
S2	Security 2 Command Class
S2 MC	Security 2 Multicast
S2 SC	Security 2 Singlecast
S2 SC-F	Security 2 Singlecast Follow-up for multicast.
SCSG	Security Commands Supported Get
SCSR	Security Commands Supported Report
SEI	Sender's Entropy Input
SIS	SUC (Node) ID Server
SOS	Singlecast Out Of Sync
SPAN	Singlecast Pre-Agreed Nonce
SRP	Secure Remote Password

Abbreviation	Explanation
SUC	Static Update Controller

1 Introduction

Commands classes are divided in four categories:

- Application Command Classes [13]
- Management Command Classes [14]
- Transport-Encapsulation Command Classes
- Network-Protocol Command Classes [15]

The complete list of existing Command Classes with their associated category is available in [12].

This document describes the Command Classes designed for Transport or Encapsulation of other command classes.

Read also this document in conjunction with [1] for Z-Wave devices and [8], [9] for Z-Wave Plus devices.

1.1 Precedence of definitions

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

1.2 Terms used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document MUST be interpreted as described in IETF RFC 2119 [4].

Statements containing the IETF RFC 2119 [4] key words are at times marked with unique requirement numbers in the margin. The requirements numbers have the following syntax: CC:xxxx.xx.xx.xx.xxx with each x being an hexadecimal digit.

This document defines functionalities as deprecated or obsoleted.

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

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

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

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

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

2 Command Class Overview

General Command Class overview and rules are described in the Application Command Class Specification [13] and are valid for the Command Classes presented in this document.

The following subsections present the additional considerations relating to the Encapsulation Command Classes.

Encapsulation Command Classes support/control

CC:0000.00.00.11.01A A node MUST advertise support for Multi Channel Command Class only if it implements End Points. A node able to communicate using the Multi Channel encapsulation but implementing no End Point

CC:0000.00.00.11.01B MUST NOT advertise support for the Multi Channel Command Class.

A node able to receive commands encapsulated with the Transport Service, Security (S0/S2) or CRC-16

CC:0000.00.00.11.01C Command Class MUST advertise the respective Command Class as supported and MUST also be able to send commands encapsulated with the respective Command Class.

The Command Class support/control definition presented in [13] applies to the Supervision Command Class.

Node Information Frame

For the NIF definition, refer to [13].

The NIF represents the Root Device's Command Class capabilities when using no Security encapsulation.

Multi Channel Root Devices MUST advertise their non-secure capabilities via the NIF.
Multi Channel End Points MUST advertise their non-secure capabilities via the Multi Channel Capability Report Command.

Security bootstrapped nodes MUST advertise their capabilities using security encapsulation (for both Root Devices and End Points) via the Security Commands Supported Report Command or the Security 2 Commands Supported Report Command.

Multi Channel overview

The section presents the concept of Multi Channel End Points. The concept ties closely to command classes such as Multi Channel Command Class, Multi Channel Association Command Class, Association Group Information Command Class as well as the Z-Wave Plus Icon Type. Multi Channel functionality may be used for controlling as well as for supporting devices.

Terminology

A Z-Wave node is conceptually an application resource in a plastic box communicating via a Z-Wave radio. Composite devices pack multiple application resources in the same plastic box, thus sharing the same Z-Wave radio. Application resources can always be addressed individually.

Within a network, a Z-Wave node is identified by its NodeID. The NodeID represents the plastic box and the radio.
Multi-resource devices are organized as Multi Channel End Points. Each application resource is identified by its own unique End Point. The plastic box itself is referred to as the Root Device.

Multi Channel Encapsulation allows a sending node to specify a source End Point and a destination End Point. Further, the destination End Point may be structured as a multicast mask, targeting up to 7 End Points by one command. Multi Channel Encapsulation is used for transmission from one End Point to another, from an End Point to a Root Device as well as from a Root Device to an End Point.

Multi Channel Encapsulation is not used between Root Devices.

An Aggregated End Point implements a function which relates to multiple individual End Points. An Aggregated End Point is addressed just as an individual End Point.
One example of an Aggregated End Point is a common power meter of a power strip which measures the total power consumption of all End Points.
The aggregation of End Points should not be confused with multicast addressing. Sending a Meter Reset command via multi-End Point addressing to all individual End Points causes all individual End Points to reset their individual meters. Sending a Meter Reset command to the Aggregated End Point causes the common power meter to be reset.

Bridging devices may provide connectivity to other technologies than Z-Wave via dynamic End Points. Dynamic End Points may be created, changed or removed.

A controlling device MAY use Multi Channel encapsulation to communicate with Multi Channel End Points in other devices. If such a controlling device does not implement any End Points, the device MUST NOT advertise the Multi Channel Command Class in the Node Information Frame.

One may create a Multi Channel Association to allow an End Point to control another End Point. The End Point may also control a Root Device. Likewise, a Multi Channel Association may be created to allow a Root Device to control an End Point.

The Association Group Information advertises the association capabilities of each Association Group in each End Point as well as in the Root Device.

Backwards compatibility

The Multi Channel concept provides a toolbox for sub-addressing. Any controlling device should implement the functionality required to interact with supporting Multi Channel devices.

Legacy devices only understand the concept of the Root Device. Therefore, a Multi Channel device providing multiple application resources also provides a meaningful subset of the application functionality via the Root Device on behalf of one or more End Points.

One example is a composite temperature and humidity sensor, which exposes the temperature sensor functionality via the Root Device as if the device was a stand-alone temperature sensor.

Another example is a 5-output power strip implemented as 5 Multi Channel End Points. The Root Device exposes one Binary Switch resource but a command to the Root Device spawns internal control commands to all outputs, so the Root Device acts as a master switch.

It is seen that the mapping of application functionality to the Root Device depends heavily on the actual product feature set. However, a few principles apply:

1. The Root Device provides access to the primary functionality of the actual product.
2. The Root Device of a Multi Channel device does not implement any application functionality that cannot be reached via End Points.
3. The Root Device of a Multi Channel always presents functionality, which can also be reached via End Point 1. If it makes sense in the actual product, a Root Device command may affect more End Points than End Point 1.

As it is optional how to forward commands from the Root Device to multiple End Points, one cannot be sure that a command to the Root Device will target all End Points. The Multi Channel multicast feature may be used to send a Set command to the first 7 End Points. Alternatively, one may communicate to each individual End Point. This works for Set as well as Get commands.

GUI presentation

The Root Device always advertises application functionality, even when the application functionality is actually provided by forwarding commands to one or more End Points of the device.

Management tools may have a need to present expanded views of Multi Channel devices, e.g. in the floor plan view of a smart home deployment. By ignoring application-style Command Classes supported by one or more End Points, the Root Device can be presented the way it really works.

Applicability examples

- A gateway may target a destination End Point to control the individual output of a power strip
- A gateway may create a Multi Channel Association from the Root Device Lifeline association group of a two-channel indoor/outdoor temperature sensor to receive sensor reports. The Multi Channel encapsulation source End Point allows the gateway to distinguish indoor readings from outdoor readings.
- One End Point of a dual End Point indoor/outdoor temperature sensor may have a (non-Multi Channel) association created to control the Root Device of an air conditioner on basis of the measured indoor temperature.

Encapsulation order overview

Command Class encapsulation MUST be applied in the following order:

1. Encapsulated Command Class (payload), .e.g Basic Set
2. Multi Command
3. Supervision
4. Multi Channel
5. Any one of the following combinations:
    a. Security (S0 or S2) followed by transport service
    b. Transport Service
    c. Security (S0 or S2)
    d. CRC16

The encapsulation order is also shown in Figure 1

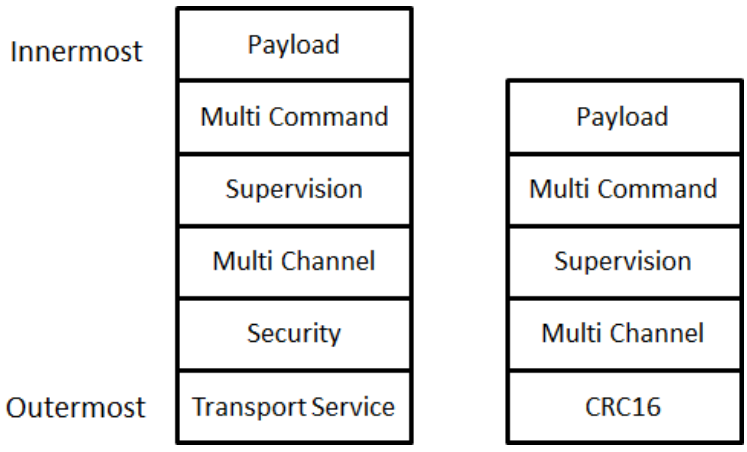

Figure 1, Encapsulation overview

An exception to Figure 1 is made when querying Multi Channel End Points about their secure capabilities. In this case, the S0/S2 Security Commands Supported Report Command is carried in a Multi

Channel encapsulation Command, thus, the encapsulation order is:
Security (Multi Channel (Security Commands Supported Get Command))

CC:0000.00.00.11.021 Responses to a given frame MUST be carried out using the same encapsulation or lack of encapsulation as it was received, unless specified otherwise in the Command Class specification.

The Transport Service Command Class and Multi Command Command Class are exceptions to the above rule.

CC:0000.00.00.11.022 The transport service MUST be used only if the payload does not fit in the Z-Wave MAC frame size.

CC:0000.00.00.12.006 The Multi Command Command Class is optional to use and SHOULD be used only if several commands are queued for transmission.

3 Command Class Definitions

The following subchapters contain definitions of Transport-Encapsulation Command Classes.

3.1 CRC-16 Encapsulation Command Class, version 1 [DEPRECATED]

THIS COMMAND CLASS HAS BEEN DEPRECATED

A device MAY implement this Command Class, but it is RECOMMENDED that new implementations use the Security 2 Command Class only.

Note: some Device Types are still REQUIRED to support this Command Class

The CRC-16 Encapsulation Command Class is used to encapsulate a command with an additional CRC-16 checksum to ensure integrity of the payload. The purpose for this command class is to ensure a higher integrity level of payloads carrying important data using 9.6/40kbps communication, in case the LRC checksum (8 bits) provided on protocol level is not sufficient to ensure integrity.

3.1.1 Compatibility Considerations

3.1.1.1 Node Information Frame (NIF)

CC:0056.01.00.21.001 A supporting node MUST always advertise the CRC-16 Command Class in its NIF, regardless of the inclusion status and security bootstrapping outcome.

CC:0056.01.00.21.002 A supporting node MUST NOT advertise the CRC-16 Command Class in its S0/S2 Commands Supported Report.

3.1.1.2 Control and support

CC:0056.01.00.21.003 The CRC-16 Encapsulation Command Class MUST NOT be encapsulated by any other Command Class.

Alternatives to using CRC-16 Encapsulation Command Class are:

- The Security (S0) or Security 2 (S2) Command Class to ensure privacy and integrity of data.
- The 100kbps communication speed already provides a CRC-16 checksum at the protocol level.

CC:0056.01.00.21.004 A node supporting the CRC-16 Encapsulation Command Class may receive a combination of encapsulated and normal non-encapsulated requests and the response MUST be as follows:

CC:0056.01.00.21.005 a) If the request is sent encapsulated, the response MUST be returned encapsulated.

CC:0056.01.00.21.006 b) If the request is sent non-encapsulated, the response MUST be sent non-encapsulated.

CC:0056.01.00.21.007 A node supporting the CRC-16 Encapsulation Command Class MUST be able to receive and interpret the encapsulated version of all the command classes that it lists in the NIF.

CC:0056.01.00.21.008 Before sending an encapsulated command, the controlling node MUST ensure that the destination supports the CRC-16 Encapsulation Command Class.

3.1.2 CRC-16 Encapsulated Command

The CRC-16 Encapsulation Command is used to encapsulate a command with an additional checksum to ensure integrity of the payload. Be aware of the payload limitations with respect to a routed single cast frame.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_CRC_16_ENCAP							
Command = CRC_16_ENCAP (0x01)							
Command Class (1 or 2 bytes)							
Command							
Data 1							
…							
Data N							
Checksum 1							
Checksum 2							

Command Class (8 bits or 16 bits)

CC:0056.01.01.11.001 This field MUST specify the Command Class identifier of the encapsulated Command. This field MUST carry a normal Command Class (8 bits) or an Extended Command Class (16 bits).

Command (8 bits)

This field MUST specify the Command identifier of the encapsulated command.

Data (N bytes)

CC:0056.01.01.11.003 This field MUST carry the payload of the encapsulated command.

Checksum (16 bits)

This field is used to advertise the checksum of the data contained in the actual command.

CC:0056.01.01.11.004 The checksum MUST be calculated using the CRC-CCITT polynomium using initialization value equal to 0x1D0F and 0x1021 (normal representation). Refer to [14] Appendix B for the CRC_CCITT source code.

CC:0056.01.01.11.005 The checksum data MUST be built by taking all bytes starting from the CRC16 Command Class identifier (COMMAND_CLASS_CRC_16_ENCAP) until the last byte of the Data field.

CC:0056.01.01.11.006 The first byte of this field MUST be the most significant byte. For example, a node sending a Basic Get Command encapsulated with CRC-16 MUST be according to Table 1.

Table 1, Basic Set Command with CRC-16 encapsulation

CRC-16 Command fields		Value	Description
1	COMMAND_CLASS_CRC_16_ENCAP	0x56	CRC-16 Command Class identifier
2	CRC_16_ENCAP	0x01	CRC-16 Encapsulation Command id
3	COMMAND_CLASS_BASIC	0x20	Basic Command Class identifier
4	BASIC_GET	0x02	Basic Get Command
5	Checksum 1	0x4D	MSB for CRC-16 checksum
6	Checksum 2	0x26	LSB for CRC-16 checksum

3.2 Multi Channel Command Class, version 3

The Multi Channel command class is used to address one or more End Points in a Multi Channel device.

Refer to 0 for an introduction to the Multi Channel concept.

3.2.1 Compatibility considerations

CC:0060.03.00.23.001 A Multi Channel device MAY implement from 1 to 127 End Points.

CC:0060.03.00.21.002 A Multi Channel device MUST implement all application functionality in End Points.

CC:0060.03.00.21.003 End Point 1 MUST implement the primary application functionality of the actual Multi Channel device.

CC:0060.03.00.23.002 Additional End Points MAY implement an identical functionality; as an example, a power strip may implement five End Points (one for each outlet) with identical functionality.

CC:0060.03.00.21.004 For backwards compatibility, the Root Device MUST mirror the application functionality of End Point 1.

CC:0060.03.00.23.003 Further, the Root Device MAY mirror the application functionalities of additional End Points. As an example, Basic Off and On commands for the Root Device may control all outlets of a power strip with five outlets.

CC:0060.03.00.21.005 With the exception of dynamic End Points, the Root Device MUST advertise all End Point functionality which is mirrored by the Root Device.

CC:0060.03.00.22.001 The Root Device SHOULD NOT advertise the application functionality of any dynamic End Point.

CC:0060.03.00.21.006 The Root Device of a Multi Channel device MUST only advertise application functionality that can be reached via one or more End Points. However, if the node supports the following Command Classes,

CC:0060.03.00.22.003 they SHOULD only be supported and advertised by the Root Device:

- Central Scene Command Class
- Configuration Command Class
- Anti-Theft Command Class
- Anti-Theft Unlock Command Class
- Clock Command Classes
- Geographic Location Command Class

CC:0060.03.00.21.007 It MUST NOT be possible to limit the functionality, or enable non-compliant behavior of any End Point in the device by sending a command to the Root Device.

3.2.1.1 Node Information Frame (NIF)

CC:0060.03.00.21.001 A node supporting this Command Class MUST set the Optional Functionality bit in its NIF.

CC:0060.03.00.21.00C Multi Channel Root Devices MUST advertise their non-secure capabilities via the NIF.

CC:0060.03.00.21.00D Multi Channel End Points MUST advertise their non-secure capabilities via the Multi Channel Capability Report Command.

CC:0060.03.00.21.00E Security bootstrapped nodes MUST advertise their capabilities using security encapsulation (for both Root Devices and End Points) via the S0 Security Commands Supported Report Command or the S2 Security 2 Commands Supported Report Command.

Secure End Point capabilities are therefore requested using the following encapsulation:

- S0/S2 Encapsulation
- Multi Channel Encapsulation
- S0/S2 Commands Supported Get/Report Commands

3.2.1.2 Dynamic End Point considerations

CC:0060.03.00.23.004 An End Point MAY be dynamic. Dynamic End points are intended End Points able to change their capabilities or that can be added and removed from a Multi Channel device.

CC:0060.03.00.22.002 When creating a new dynamic End Point, it SHOULD be assigned an End Point identifier which has not been used recently to allow applications to discover the removal of a dynamic End Point.

CC:0060.03.00.21.009 When a dynamic End Point is removed, all other End Points MUST maintain their current End Point identifiers.

CC:0060.03.00.23.005 A supporting device implementing dynamic End Points MAY advertise the creation, change or removal of a dynamic End Point via the Root Device Lifeline association group by issuing a Multi Channel Capability Report.

CC:0060.03.00.21.00A A node MUST NOT advertise changes to dynamic End Points via broadcast transmission. A receiving node MUST ignore such broadcasted advertisements.

CC:0060.03.00.21.00B After advertising the removal, a removed dynamic End Point MUST ignore all commands.

3.2.2 Interoperability considerations

CC:0060.03.00.53.001
CC:0060.03.00.51.001 A controlling node MAY use Multi Channel Encapsulation Command to communicate with Multi Channel End Points in other nodes. If such a controlling node does not implement any End Points, it MUST NOT advertise the Multi Channel Command Class in its Node Information Frame (NIF) or S0/S2 Commands Supported Report Command.

An example of such device would be a controller or gateway which can control End Points in other supporting nodes via commands from the Root Device of the gateway. Likewise, the Root Device of the gateway may receive unsolicited commands from other nodes End Points.

3.2.3 Multi Channel End Point Get Command

This command is used to query the number of End Points implemented by the receiving node.

CC:0060.03.07.11.001 The Multi Channel End Point Report Command MUST be returned in response to this command.

CC:0060.03.07.51.001 This command MUST NOT be issued via multicast addressing.

CC:0060.03.07.11.002 A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_END_POINT_GET (0x07)							

3.2.4 Multi Channel End Point Report Command

This command is used to advertise the number of End Points implemented by the sending node.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_END_POINT_REPORT (0x08)							
Dyna-mic	Iden-tical	Res					
Res	End Points						

Dynamic (1 bit)

This field is used to advertise if the node implements a dynamic number of End Points.

CC:0060.03.08.11.001 The value 1 MUST be used to indicate that the number of End Points is dynamic.
The value 0 MUST be used to indicate that the number of End Points is static.

Identical (1 bit)

This field is used to advertise if all end points have identical capabilities

CC:0060.03.08.11.002 This bit MUST be set to 1 if all End Points advertise the same Generic and Specific Device Class and support the same Command Classes.

CC:0060.03.08.11.003 This bit MUST be set to 0 if End Points do not advertise the same Device Class or Command Class information.

Res

CC:0060.03.08.11.004 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

End Points (7 bits)

This field is used to advertise the number of End Points implemented by the sending node.

CC:0060.03.08.11.005 This field MUST be in the range 1..127.

CC:0060.03.08.11.006
CC:0060.03.08.13.001 If the sending node implements dynamic End Points, this field MUST advertise the number of End Points currently instantiated by the node. A dynamic End Point MAY be assigned any End Point identifier in the range 2..127.

3.2.5 Multi Channel Capability Get Command

This command is used to query the non-secure Command Class capabilities of an End Point.

CC:0060.03.09.11.001 The Multi Channel Capability Report Command MUST be returned in response to this command unless it is to be ignored.

CC:0060.03.09.51.001 This command MUST NOT be issued via multicast addressing.

CC:0060.03.09.11.002 A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_CAPABILITY_GET (0x09)							
Res	End Point						

Res

CC:0060.03.09.11.003 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

End Point (7 bits)

CC:0060.03.09.11.004 This field MUST specify the End Point for which the capabilities MUST be returned.

CC:0060.03.09.11.005 If the specified End Point does not exist, this command MUST be ignored.

If the specified End Point represents a removed dynamic End Point, this command MUST be ignored.

3.2.6 Multi Channel Capability Report Command

This command is used to advertise the Generic and Specific Device Class and the supported command classes of an End Point.

CC:0060.03.0A.11.002 When advertising the removal of a dynamic End Point, this command MUST carry the following values:

- Dynamic MUST be set to 1
- End Point MUST be set to the actual End Point identifier
- Generic Device Class MUST be set to 0xFF (GENERIC_TYPE_NON_INTEROPERABLE)
- Specific Device Class MUST be set to 0x00 (SPECIFIC_TYPE_NOT_USED)

CC:0060.03.0A.11.003
- The Command Class field MUST be omitted.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_CAPABILITY_REPORT (0x0A)							
Dynamic	End Point						
Generic Device Class							
Specific Device Class							
Command Class 1							
…							
Command Class N							

Dynamic (1 bit)

This field is used to advertise if the advertised End Point is dynamic.

CC:0060.03.0A.11.001 This field MUST be set to 1 if this is a dynamic End Point.
This field MUST be set to 0 to indicate that this is a static End Point.

End Point (7 bits)

CC:0060.03.0A.11.004 This field MUST advertise the actual End Point for which the capabilities are advertised.

Generic Device class (8 bits)

CC:0060.03.0A.11.005 This field MUST carry the Generic Device Class of the advertised End Point. For a detailed description of all available Generic Device Classes, refer to [1] for Z-Wave nodes and [8], [9] for Z-Wave Plus nodes.

Specific Device class (8 bits)

CC:0060.03.0A.11.006 This field MUST carry the Specific Device Class of the advertised End Point. For a detailed description of all available Specific Device Classes, refer to [1] for Z-Wave nodes and [8], [9] for Z-Wave Plus nodes.

Command Class (N bytes)

This field is used to advertise the non-secure supported Command Classes by the actual End Point.

CC:0060.03.0A.11.007 This field MUST be omitted if the advertised End Point does not exist or have been removed. The number of Command Class bytes MUST be determined from the length of the frame.

CC:0060.03.0A.11.009 This field MUST represent the capabilities of an End Point with no security encapsulation.

CC:0060.03.0A.11.008 The Multi Channel Command Class MUST NOT be advertised in this list.

CC:0060.03.0A.11.00A Non-secure End Point capabilities MUST also be supported securely and MUST also be advertised in the S0/S2 Commands Supported Report Commands unless they are encapsulated outside Security or Security themselves.

CC:0060.03.0A.11.00B Nodes supporting S0 MUST advertise S0 as supported for each End Point that can be addressed with S0 encapsulation

CC:0060.03.0A.11.00C Nodes supporting S2 MUST support addressing every End Point with S2 encapsulation and MAY explicitly list S2 in the non-secure End Point capabilities.

3.2.7 Multi Channel End Point Find Command

This command is used to request End Points having a specific Generic or Specific Device Class in End Points.

CC:0060.03.0B.11.001 The Multi Channel End Point Find Report Command MUST be returned in response to this command.

CC:0060.03.0B.51.001 This command MUST NOT be issued via multicast addressing.

CC:0060.03.0B.11.002 A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_END_POINT_FIND (0x0B)							
Generic Device Class							
Specific Device Class							

Generic Device Class (8 bits)

CC:0060.03.0B.11.003 This field MUST indicate the receiving node to return the list of End Points having the specified Generic Device Class.

CC:0060.03.0B.11.004 The value 0xFF MUST indicate that all existing End Points MUST be returned. If this field is set to 0xFF, the Specific Device Class field MUST also be set to 0xFF.

Specific Device Class (8 bits)

CC:0060.03.0B.11.005 This field MUST indicate the receiving node to return the list of End Point having the specified Specific Device Class.

CC:0060.03.0B.11.006 The value 0xFF MUST indicate that the list of all End Points having the specified Generic Device Class MUST be returned.

3.2.8 Multi Channel End Point Find Report Command

This command is used to advertise End Points that implement a given combination of Generic and Specific Device Classes.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_END_POINT_FIND_REPORT (0x0C)							
Reports to Follow							
Generic Device Class							
Specific Device Class							
Res	End Point 1						
…							
Res	End Point N						

Reports to Follow (8 bits)

This field is used if multiple Report Commands are necessary for returning all the requested End Points.

CC:0060.03.0C.11.001 This field MUST advertise the number of Multi Channel End Point Find Report Command following the actual frame.

Generic Device Class (8 bits)

This field is used to advertise the Generic Device Class of all advertised End Points in this command.

CC:0060.03.0C.11.002 The value 0xFF MUST be advertised if this value was specified in the Multi Channel End Point Find Command.
If 0xFF is advertised, the *Specific Device Class* field MUST also advertise the value 0xFF.

CC:0060.03.0C.13.001 If the value 0xFF is advertised, the advertised End Points MAY implement different Generic and Specific Device Classes.

Specific Device Class (8 bits)

This field is used to advertise the Specific Device Class of all advertised End Points in this command.

CC:0060.03.0C.13.002 If the value 0xFF is advertised, the advertised End Points MAY implement different specific device classes.

CC:0060.03.0C.11.004 This field MUST be set to 0xFF if the *Generic Device Class* field is set to 0xFF.

Res

CC:0060.03.0C.11.005 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

End Point (N * 7 bits)

This field is used to advertise the list of End Point identifier(s) that matches the advertised Generic and Specific Device Class values.

CC:0060.03.0C.11.006 If no End Point matches the advertised Generic Device Class and/or Specific Device Class, the sending node MUST set this field to 0x00 and this field's size MUST be 7 bits (only 1 list entry).

3.2.9 Multi Channel Command Encapsulation Command

This command is used to encapsulate commands to or from a Multi Channel End Point.

CC:0060.03.0D.11.001 The Multi Channel Command Encapsulation Command MUST NOT carry Source End Point and Destination End Point fields that are both zero.

CC:0060.03.0D.13.001 CC:0060.03.0D.11.002 A receiving node MAY respond to a Multi Channel encapsulated command if the Destination End Point field specifies a single End Point. In that case, the response MUST be Multi Channel encapsulated.

CC:0060.03.0D.11.003 A receiving node MUST NOT respond to a Multi Channel encapsulated command if the Destination End Point field specifies multiple End Points via bit mask addressing.

CC:0060.03.0D.11.004 A node MUST NOT return a Multi Channel Encapsulated command in response to a non-encapsulated command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_CMD_ENCAP (0x0D)							
Res	Source End Point						
Bit address	Destination End Point						
Command Class (1 or 2 bytes)							
Command							
Parameter 1							
...							
Parameter N							

Res

CC:0060.03.0D.11.005 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Source End Point (7 bits)

CC:0060.03.0D.11.006 This field is used to advertise the originating End Point. The Source End Point MUST be in the range 0..127.

CC:0060.03.0D.11.007 The value 0 MUST indicate that the encapsulated command is issued by the Root Device.
Values in the range 1...127 MUST indicate the actual End Point identifier which issues the encapsulated command.

CC:0060.03.0D.11.008 This field MUST be set to a different value than 0 if the *Destination End Point* field is set to 0.

CC:0060.03.0D.11.009 A node returning a response to a Multi Channel encapsulated command MUST swap the Source and Destination End Point identifiers in this command.

CC:0060.03.0D.11.00A This field MUST be set to 0 if a sending node does not implement Multi Channel End Points or if the Root Device of the Multi Channel device is issuing a command.

Bit address (1 bit)

This bit is used to advertise if the destination End Point is specified as a bit mask.

CC:0060.03.0D.11.00B The value 1 MUST indicate that the Destination End Point field is specified as a bit mask.
The value 0 MUST indicate that the Destination End Point field is specified as an End Point identifier.

Destination End Point (7 bits)

This field is used to advertise the destination End Point.

CC:0060.03.0D.11.00C This field MUST be interpreted based on the “Bit address” value.

CC:0060.03.0D.11.00D If the Bit Address field is set to 0, the Destination End Point field MUST carry a single End Point identifier value in the range 0..127.

CC:0060.03.0D.11.00E If the Bit Address field is set to 1, the Destination End Point MUST use the following encoding:

- Bit 0 in the Destination End Point indicates if End Point 1 is a destination
- Bit 1 in the Destination End Point indicates if End Point 2 is a destination
- …

CC:0060.03.0D.11.00F The bit value 0 MUST be used to advertise that the corresponding End Point is not a destination.
The bit value 1 MUST be used to advertise that the corresponding End Point is a destination.

Command Class (8 bits or 16 bits)

CC:0060.03.0D.11.011 This field MUST specify the Command Class identifier of the encapsulated Command. This field MUST carry a normal Command Class (8 bits) or an Extended Command Class (16 bits)

Command (8 bits)

CC:0060.03.0D.11.012 This field MUST specify the Command identifier of the encapsulated command.

Parameter (N bytes)

CC:0060.03.0D.11.010 This field MUST carry the payload of the encapsulated command. The length of this field MUST be determined from the Z-Wave frame length.

3.3 Multi Channel Command Class, version 4

The Multi Channel Command Class is used to address one or more End Points in a Multi Channel device.

Refer to 0 for an introduction to the Multi Channel concept.

3.3.1 Compatibility considerations

CC:0060.04.00.21.001 Compatibility considerations requirements from version 3 MUST also be observed by a version 4 supporting node. Refer to 3.2.1 Compatibility considerations.

CC:0060.04.00.21.002 Multi Channel Command Class, version 4 is backwards compatible with Multi Channel Command Class, version 3. Fields and commands not described in this version MUST remain unchanged from version 3.

The Multi Channel Command Class, version 4 introduces Aggregated End Points. Aggregated End Points are assigned End Point identifiers following immediately after the identifiers allocated to individual End Points. Thus:

- Aggregated End Points are invisible to devices supporting Multi Channel Command Class, version 3 or older.
- Individual End Points are identical in version 3 and version 4.
- A version 3 controlling node can discover and control individual End Points.
- A version 4 controlling node can discover and control individual End Points and Aggregated End Points.

3.3.2 Interoperability considerations

CC:0060.04.00.31.001 Interoperability considerations from version 3 MUST also apply in this version.
Refer to 3.2.2 Interoperability considerations

Aggregated End Point design principles

CC:0060.04.00.31.002 An Aggregated End Point MUST implement a function which relates to multiple individual End Points.

CC:0060.04.00.31.003 An Aggregated End Point MUST NOT forward commands to individual End Points. In other words, communication to a number of individual End Points MUST be done via multiple singlecast commands or via bit mask addressing.

CC:0060.04.00.31.004 A command issued to an Aggregated End Point MUST NOT cause any individual End Point to return a command in response.

CC:0060.04.00.31.005 Aggregated End Point MUST be assigned End Point identifiers from a continuous range starting immediately after the last individual End Point.

CC:0060.04.00.31.006 An Aggregated End Point MUST NOT support other Command Classes and types than the ones explicitly listed in Table 2.

Table 2, Aggregated End Point Command Class support

Command Class	Type	Measurement mode
Meter	Electricity	Instant, Accumulated
Meter	Gas	Instant, Accumulated
Meter	Water	Instant, Accumulated
Multilevel Sensor	Power	Instant
Multilevel Sensor	Current	Instant
Multilevel Sensor	Air flow	Instant
Multilevel Sensor	Tank Capacity	Instant

3.3.2.1 Dynamic End Point considerations

In the case a node implements both Dynamic and Aggregated End Points, the Aggregated End Points identifiers will vary accordingly to the last active dynamic End Point. An illustration is given in Figure 2

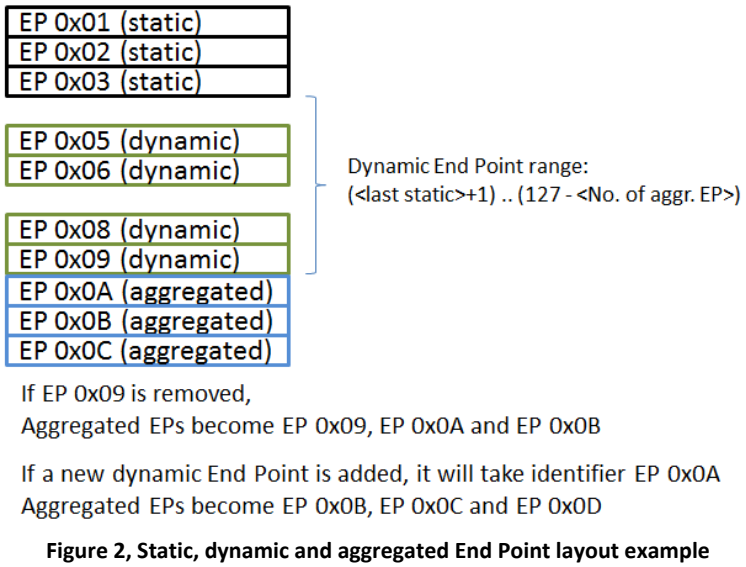

Figure 2, Static, dynamic and aggregated End Point layout example

3.3.3 Multi Channel End Point Report Command

This command is used to advertise the number of Multi Channel End Points and other relevant Multi Channel attributes.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_END_POINT_REPORT (0x08)							
Dyna-mic	Iden-tical	Res					
Res	Individual End Points						

Res	Aggregated End Points

CC:0060.04.08.11.001 Fields not described below MUST remain unchanged from version 3. Refer to 3.2.4 Multi Channel End Point Report Command

Res

CC:0060.04.08.11.002 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Individual End Points (7 bits)

This field is used to advertise the number of individual End Points implemented by the sending node.

CC:0060.04.08.11.003 This field MUST be in the range 1..127.

CC:0060.04.08.11.004 CC:0060.04.08.13.001 If the sending node implements dynamic End Points, this field MUST advertise the number of End Points currently instantiated by the node. A dynamic End Point MAY be assigned any End Point identifier in the range 2..127.

CC:0060.04.08.11.005 The sum of the values advertised by this field and the *Aggregated End Points* field MUST be in the range 1..127.

Aggregated End Points (7 bits)

This field is used to advertise the number of Aggregated End Points implemented by this node.

CC:0060.04.08.11.006 This field MUST be in the range 0..127.
The value 0 MUST indicate that no Aggregated End Points are implemented by the sending node.

3.3.4 Multi Channel Aggregated Members Get Command

This command is used to query the members of an Aggregated End Point.

CC:0060.04.0E.11.001 The Multi Channel Aggregated Members Report MUST be returned in response to this command.

CC:0060.04.0E.51.001 This command MUST NOT be issued via multicast addressing.

CC:0060.04.0E.11.002 A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_AGGREGATED_MEMBERS_GET (0x0E)							
Res			Aggregated End Point				

Res

CC:0060.04.0E.11.003 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Aggregated End Point (7 bits)

CC:0060.04.0E.11.004 This field MUST specify the Aggregated End Point identifier for which the aggregated members MUST be returned.

CC:0060.04.0E.11.005 The value MUST be in the range of advertised Aggregated End Points. If the value does not indicate valid aggregated End Point identifier, a receiving node MUST return a response with the *Number of Bit Masks* field set to zero.

3.3.5 Multi Channel Aggregated Members Report Command

This command is used to advertise the members of an Aggregated End Point.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CHANNEL							
Command = MULTI_CHANNEL_AGGREGATED_MEMBERS_REPORT (0x0F)							
Res	Aggregated End Point						
Number of Bit Masks							
Aggregated Members Bit Mask 1							
…							
Aggregated Members Bit Mask N							

Res

CC:0060.04.0F.11.001 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Aggregated End Point (7 bits)

This field is used to advertise the Aggregated End Point identifier for which the members are advertised in this command.

Number of Bit Masks (8 bits)

This field is used to advertise the length in bytes of the *Aggregated Member Bit Mask* field.

CC:0060.04.0F.11.002 The value 0 MUST indicate that the Aggregated Members Bit Mask field is MUST be omitted.
Values in the range 1..255 MUST indicate the length of the *Aggregated Members Bit Mask* field in bytes.

Aggregated Members Bit Mask (N bytes)

This field is used to advertise the End Point members of the actual Aggregated End Point.

CC:0060.04.0F.11.003 The length of this field in bytes MUST be according to the *Number of Bit Masks* field value.

This field MUST be treated as a bit mask and MUST use the following encoding for advertising members:

- Bit 0 in Bit Mask 1 indicates if End Point 1 is a member
- Bit 1 in Bit Mask 1 indicates if End Point 2 is a member
- …

CC:0060.04.0F.11.004 The bit value 0 MUST be used to advertise that the corresponding End Point is not a member.
The bit value 1 MUST be used to advertise that the corresponding End Point is a member.

CC:0060.04.0F.11.005 The first byte of this field MUST represent End Points 1..8.

3.4 Multi Command Command Class, version 1

The Multi Command Command Class is used to bundle multiple commands in one encapsulation Command. This command class may be used to limit the number of transmissions and to extend battery lifetime.

3.4.1 Interoperability considerations

The "Answer-as-asked" requirement for Multi Command Encapsulation commands carrying Get type commands has been OBSOLETED.

This allows for a simple parser design in end devices, enables battery power savings and supports the deployment of gateways with a part of the application logic placed in the cloud.

Refer to section 3.4.2 for updated requirements text.

Older implementations may expect Get type commands to be answered with the same encapsulation.

CC:008F.01.00.31.001 However, responding nodes MUST NOT return answers with the same encapsulation if the destination does not advertise the Multi Command Command Class as supported.

CC:008F.01.00.32.001 Supporting nodes SHOULD NOT return an answer Multi Command encapsulated if returning a single command and SHOULD NOT return individually Multi Command encapsulated response commands.

3.4.2 Compatibility Considerations

3.4.2.1 Multi Command Support

CC:008F.01.00.21.001 A node supporting this Command Class MUST be able to receive Multi Command Encapsulated commands.

CC:008F.01.00.21.002 A supporting node MUST support the Multi Command Encapsulation of all command classes advertised as supported (for the received Security Class) except for command classes that are encapsulated outside Multi Command. Refer to the encapsulation order defined in section 0.

CC:008F.01.00.21.003 A supporting node MUST respond to an encapsulated command requiring an answer to be returned, e.g. a Get type Command.

CC:008F.01.00.21.004 A responding node MUST NOT return Multi Command encapsulated commands in response to encapsulated requests if the sender does not support the Multi Command Command Class.

3.4.2.2 Multi Command Control

A node controls the Multi Command Command Class if it sends Multi Command Encapsulated commands to supporting nodes.

CC:008F.01.00.23.001 It means that a node MAY issue unsolicited Multi Command Encapsulated commands without advertising support for the Multi Command Command Class.

CC:008F.01.00.51.001 A controlling node MUST verify that the destination supports Multi Command in its NIF before using Multi Command Encapsulation or be explicitly enabled by another controller node to use Multi Command encapsulation.

3.4.2.3 Node Information Frame (NIF)

CC:008F.01.00.21.005 A supporting node MUST always advertise the Multi Command Command Class in its NIF, regardless of the security bootstrapping outcome.

This allows other nodes bootstrapped on any security level to know that they can use the Multi Command encapsulation with the supporting node.

3.4.3 Multi Command Encapsulated Command

The Multi Command Encapsulated Command used to contain multiple Commands.

CC:008F.01.01.11.001 The encapsulated Commands MUST be executed in the order they are received. In case Get type Commands in a Multi Command Encapsulated Command are received by a device, the Report type Commands MUST be returned in the same order as the Get type Commands were received.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MULTI_CMD							
Command = MULTI_CMD_ENCAP (0x01)							
Number of Commands							
Command Length 1							
Command Class 1 (1 or 2 bytes)							
Command 1							
Data 1,1							
…							
Data 1,N							
…							
Command Length X							
Command Class X (1 or 2 bytes)							
Command X							
Data X,1							
…							
Data X,N							

Number of Commands (8 bits)

CC:008F.01.01.11.002 This field MUST specify the number of encapsulated commands.

CC:008F.01.01.12.001 This field SHOULD be set to a value greater than 1.

CC:008F.01.01.11.003 Each block carrying an encapsulated command MUST comprise the following fields:

- Command length
- Command Class
- Command
- Data

CC:008F.01.01.11.004 A supporting node MUST accept and execute all encapsulated commands contained in this command. A supporting node MUST NOT discard any encapsulated command based on the number of commands encapsulated in the command.

Command Length (8 bits)

CC:008F.01.01.11.005 This field MUST specify the number of bytes occupied by the Command Class, Command and the Data fields in the actual command block.

Command Class (8 bits or 16 bits)

CC:008F.01.01.11.006 This field MUST specify the Command Class identifier of the encapsulated command. This field MUST carry a normal Command Class (8 bits) or an Extended Command Class (16 bits).

Command (8 bits)

CC:008F.01.01.11.007 This field MUST specify the Command identifier of the encapsulated Command.

Data (N bytes)

CC:008F.01.01.11.008 This field MUST carry the payload of the encapsulated command.

3.5 Security 0 (S0) Command Class, version 1

The Security Command Class 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.

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.

Compatibility considerations

A node supporting the S0 Command Class MAY use the S2 CTR_DRBG as a PNRG.

3.5.1.1 Node Information Frame (NIF)

A supporting node MUST advertise the Security 0 Command Class in its NIF before inclusion.

A supporting node MUST advertise the Security 0 Command Class in its NIF after successful S0 security bootstrapping.

A supporting node MAY advertise the Security 0 Command Class in its NIF after inclusion without Security bootstrapping.

A supporting node MUST NOT advertise the Security 0 Command Class in its S0/S2 Commands Supported Report list.

3.5.2 Message Encapsulation and Command Class Handling

For encapsulating messages, Z-Wave requires four commands. Before sending an encrypted frame, the sender MUST request a nonce (number used once) from the recipient. The sender subsequently uses this number along with the locally generated nonce and the network key to generate the Security Message Encapsulation Command as illustrated in Figure 3.

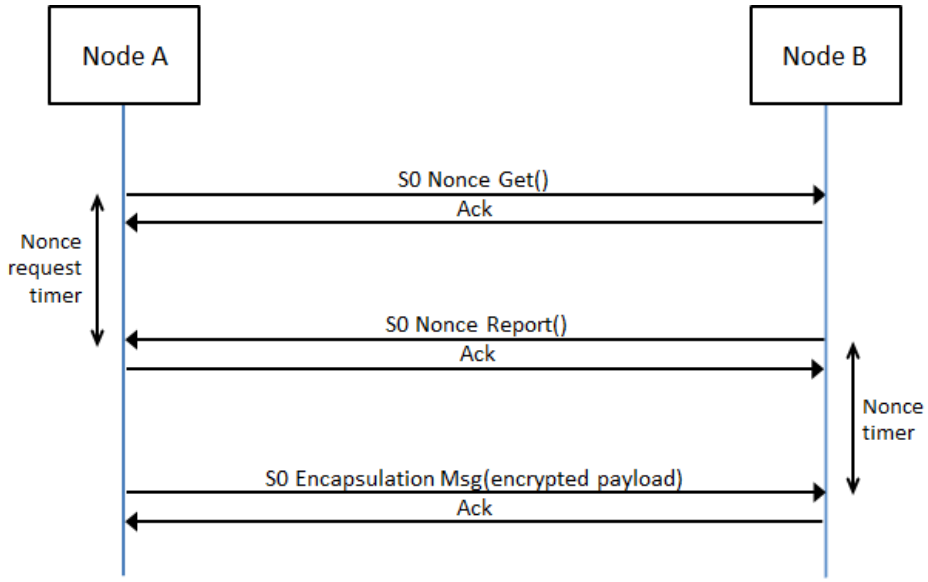

Figure 3, Sending secure messages

This mechanism generates an overhead of three commands for each single frame that is sent encrypted (plus acknowledge frames).

A number of timers have to be implemented in order to mitigate attacks.

A timer denoted *Nonce request timer* in Figure 3 and Figure 4 SHOULD be started by a node sending a Nonce Get Command. If the *Nonce request timer* is started, the Nonce Report MUST be received before the timer runs out. The duration of this timer will depend on the application it is trying to protect.

A timer denoted *Nonce timer* in Figure 3 and Figure 4 MUST be started by a node after sending a Nonce Report Command. The S0 Encapsulated Message MUST be received within the specified timeout in order to be accepted.

The *Nonce timer* MUST implement a timeout in the range 3..20 seconds.

Note that the *Nonce timer* and the *Nonce request timer* MUST be started when the command has been sent and not when the transmission has been acknowledged, since an attacker could delay the acknowledgement frame.

Both 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 converge towards two (instead of three) transmissions as the number of frames increases.

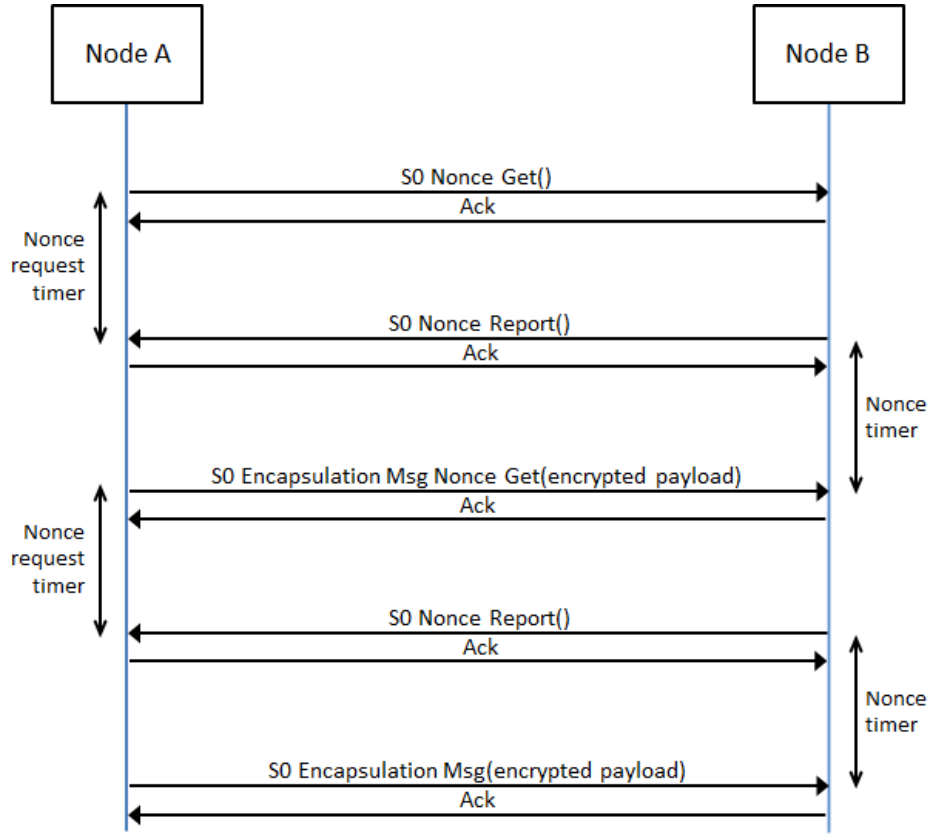

Figure 4, Streaming secure messages

Notice: The maximum command size is reduced by 20 bytes due to the security encapsulation command overhead. Larger commands can use sequencing as described in 3.5.2.3.

3.5.2.1 Nonce Get Command

This command is used to request an external nonce from the receiving node.

Note that a nonce will only be valid for one encrypted command attempt. The nonce is discarded when the receiver has used it for decrypting the next received command. A new nonce MUST be exchanged for each new command.

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

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

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Security Header = SECURITY_NONCE_GET							

3.5.2.2 Nonce Report Command

This command is used to return the next nonce to the requesting node.

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 (8 bytes)

This field contains the 8 bytes external nonce used for encryption, generated with the PNRG by the sending node.

3.5.2.3 Security Message Encapsulation Command

The device uses the Security Message Encapsulation command to encapsulate Z-Wave commands using AES-128.

A sending node is also requesting a new nonce from the receiving node when transmitting the Security Message Encapsulation Nonce Get Command. The sending node uses the new nonce when streaming multiple secure messages without having to send a separate Nonce Get Command after sending each command as shown in Figure 4, Streaming secure messages.

A device MUST ignore the received Security Message Encapsulation Command if the generated Nonce has timed out.

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 (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.

Reserved

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

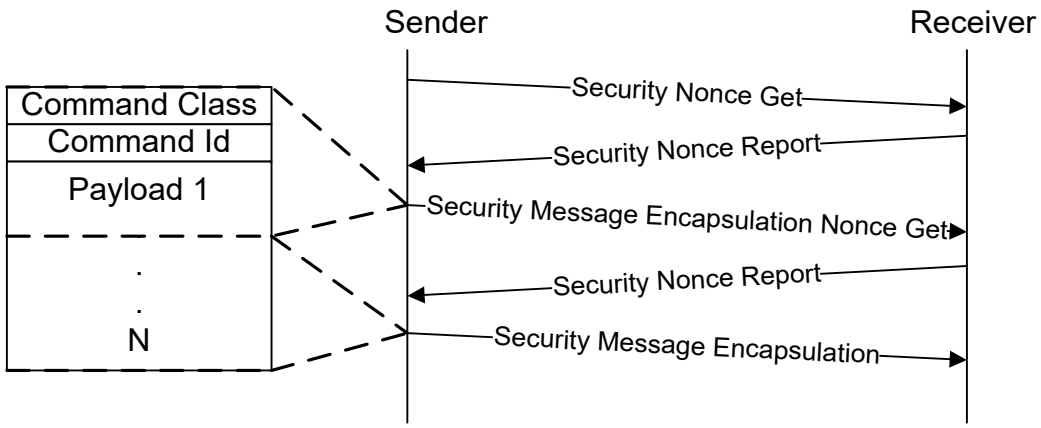

Figure 5, Frame flow for sequenced frames

Sequenced (1 bit)

This flag MUST be set if the command is transmitted using multiple frames. This flag MUST not set if the command is contained entirely in a single (this) frame. As shown in figure, the first frame in a sequence MUST be sent using Security Message Encapsulation Nonce Get. To minimize overhead, following frames SHOULD be sent using the Security Message Encapsulation Nonce Get command. The last frame MAY be sent using Security Message Encapsulation.

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:

Table 3, Security message encapsulation::Second Frame combinations

	Sequenced 1	**Sequenced 0**
Second Frame 1	Second frame of two	-
Second Frame 0	First frame of two	Single Frame

Sequence Counter (4 bits)

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 NodeID 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 bits) (Part of Encrypted Payload)

This field contains the identifier of the Command class, which the device sends to the NodeID.

Command identifier (8 bits) (Part of Encrypted Payload)

This field contains the identifier of the Command, which the device sends to the NodeID.

Command byte (N bytes) (Part of Encrypted Payload)

These fields contain the parameters, which the device sends to the NodeID.

Receiver's nonce Identifier (8 bits)

Identifies nonce being used.

Message Authentication Code byte (8 bytes)

Data used for authenticating the received message to prevent tampering.

3.5.3 Network Key Management

The same network key is used by all secure nodes in the network. 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.

3.5.3.1 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.

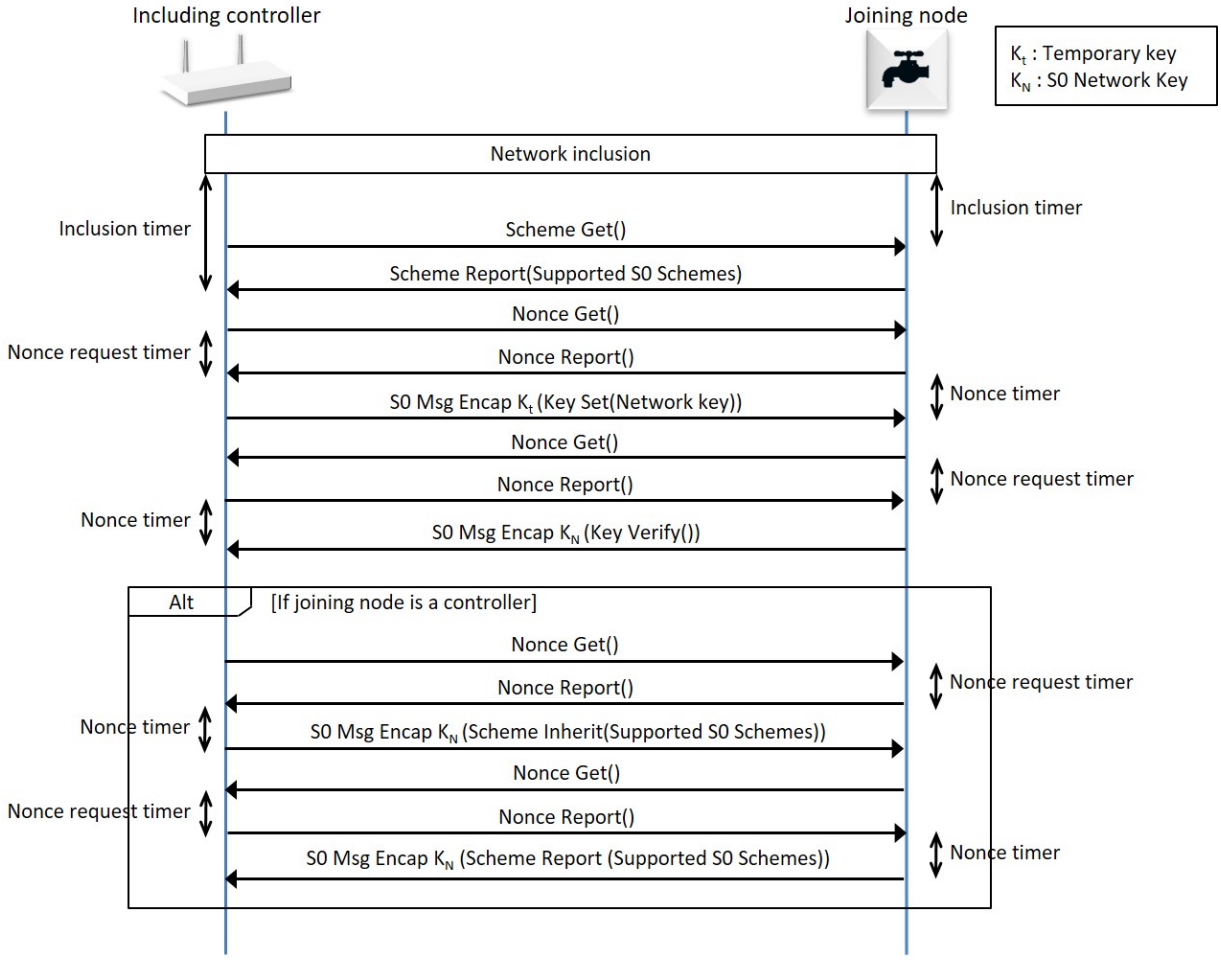

Figure 6, Inclusion into a secure network

To protect the security of a secure network, 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 currently available Security 0/N scheme, the same network key is used by all nodes in the network.

For the including controller to allow S0 bootstrapping into the secure network, a common security scheme needs to be supported by both nodes. When supporting multiple common schemes, the highest possible scheme MUST be used. If no common schemes are supported the node MUST NOT be S0 bootstrapped.

When controller nodes in the secure network wish to establish a connection to a node that supports the Security 0 Command class, they MUST send the Security 0 Command Supported Get Command to the node. Receiving no Security Command Supported Report (since the receiving node does not have the key to decrypt the request), means that it will not be able to talk to this node securely. The same applies for the situation where a secure node does not become part of the secure network because it was included by a non-secure controller.

A node based on a slave Role Type MUST NOT consider a secure inclusion successful until the Network Key Set has been received.

A node based on a controller Role Type MUST NOT consider the secure inclusion successful until the Security Scheme Inherit Command has been received.

3.5.3.1.1 Inclusion through Non-Secure Inclusion controller

A Security-enabled SIS MAY perform secure setup after inclusion from a non-secure inclusion controller. As soon as the Security enabled SIS (hereafter SIS), receives information from the non-secure inclusion controller that a node with support for the Security command class has been included, the SIS MAY start the secure setup process of sending the network key to the newly included node as illustrated in Figure 7. At this stage the SIS acts as if it, itself had performed the inclusion and MAY carry out all the steps REQUIRED for secure setup, included making sure the timeouts are not exceeded.

Before starting the Secure inclusion process, the SIS MUST be put into a state that allows it to carry out the secure setup for 1 node for the next 3 minutes and no longer. The SIS MUST be put in this state through a password-protected menu to avoid unintentional reveal of the network key by a fake controller.

It should be noted that performing the secure setup on behalf of a non-secure inclusion controller might add to the complexity of the actions required by the user, and thus make it easier for a hacker to perform social engineering to circumvent the security so care must be taken to inform the user accordingly.

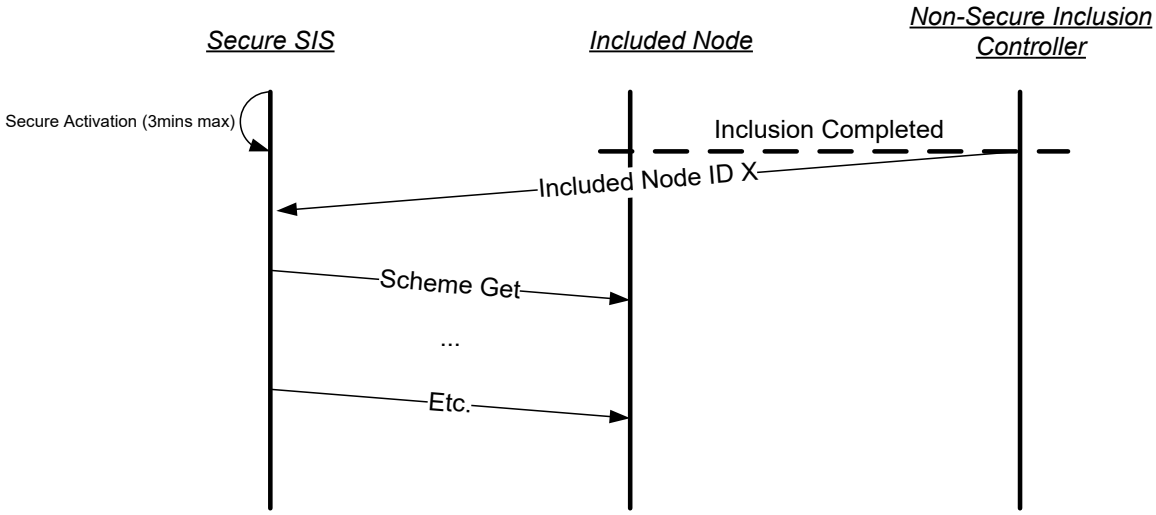

Figure 7, Secure Inclusion through Non-Secure Inclusion Controller

3.5.3.1.2 Inclusion Timers

As shown in Figure 6, a number of timeout MUST be complied with. For the including controller see Figure 8.

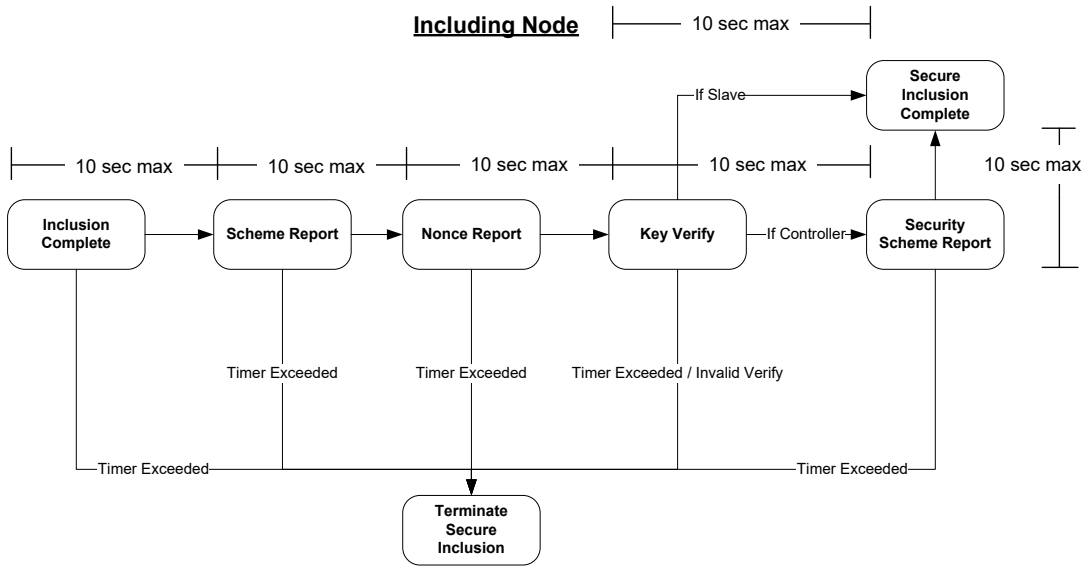

Figure 8, Timers on Including Controller

For the new included node, the timers in Figure 9 MUST be complied with.

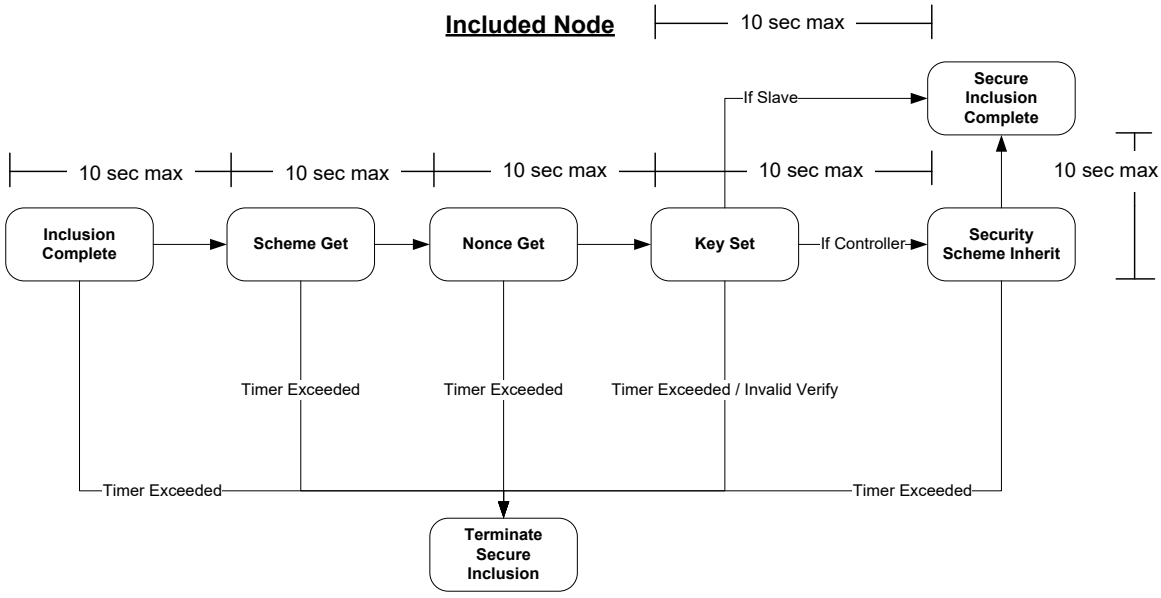

Figure 9, Timers on newly Included Node

The Network Key MUST NOT be sent to the new node if a Security Scheme Report Command is received by the including controller later than 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 NOT accept and respond to a Scheme Get it is received later than 10 seconds after network inclusion. When a valid frame is received before the timeout, the timeout is extended to allow the next part of the inclusion process. The S0 bootstrapping process MUST be terminated if any message times out.

3.5.3.2 Security Scheme Get Command

A controlling device MUST send Security Scheme Get Command immediately after the successful inclusion of a node that supports the Security Command class.

A node is considered newly included if it has been included for less than 10 seconds.

A newly included node MUST return the Security Scheme Report Command in response to this command.

Whether a node has been included securely or non-securely, the node MUST NOT respond to the Security Scheme Get command if it is not newly included.

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

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Command = SECURITY_SCHEME_GET							
Supported Security Schemes							

Supported Security Schemes (8 bits)

The Security Schemes which are supported by the primary/inclusion controller. At least one security scheme MUST be supported. Values MUST comply withTable 4.

Table 4, Security Scheme Get::Supported Security Schemes encoding

Bit	Supports
0	Security 0 using normal power = 0

Bit 0 MUST always be set to 0, indicating support for Security 0. All other bits are reserved and MUST be set to zero by a sending node. Reserved bits MUST be ignored by a receiving node.

3.5.3.3 Security Scheme Report Command

This command is used to advertise security scheme 0 support by the node being included. Upon reception, the including controller MUST send the network key immediately without waiting for input, by using 16 times 0x00 as the temporary key. The including controller MUST NOT perform any validation of the Supported Security Schemes byte.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Command = SECURITY_SCHEME_REPORT							
Supported Security Schemes							

Supported Security Schemes (8 bits)

Refer to Security Scheme Get Command.

3.5.3.4 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 MUST handle the Network Key Set command according to the guidelines in section 3.5.3.

This command MUST be sent 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 byte (N bytes)

The Network key to exchange application data secure in the network.

3.5.3.5 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 SHOULD be concluded that the node has not been included properly.

This command MUST be sent encapsulated by the Security Message Encapsulation command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Command = NETWORK_KEY_VERIFY							

3.5.3.6 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 6.

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 be sent 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 (8 bits)

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 6. 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.

3.5.4 Encapsulated Command Class Handling

The Node Info Frame is only used to advertise all the command classes that are supported non-securely. Command classes supported securely MUST be advertised by using the Security Commands Supported Get/Report.

- All non-securely supported command classes MUST also be supported securely.
- All non-securely controlled command classes MUST also be controlled securely.
- All non-securely supported command classes MUST NOT be explicitly advertised in the Security Commands Supported 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 advertise all non-securely supported command classes, while the Node Info Frame MAY advertise non-securely controlled command classes.

If the node is included into a secure network, it MAY 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.

If an S0 node is included into a non-secure network, it MAY choose to support command classes it would not support non-securely if it had been included into a secure network.

An example of this could be a relay as shown in Table 5.

Table 5, S0 Node Command Class support depending on inclusion (example)

	Before Inclusion	**Included Non-Secure**	**Included Secure**
Security Command Supported Report Frame	-N/A	-N/A	Binary Switch Version
Node Info Frame	Security Binary Switch Version	Binary Switch Version	Security

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 securely it MUST NOT be listed in the node info frame, while it MUST be advertised in the security commands supported report frame.

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.

3.5.4.1 Multi Channel Handling

Any device that supports the Security and Multi Channel Command Classes MAY choose to support a different set of Command Classes securely for each Multi Channel End Point. An End Point with support for Security MUST report the Security Command Class as supported for that End Point. The command classes supported for each endpoint securely is determined by using the Security Commands Supported Get command sent to each individual endpoint Security Encapsulated. Hence, the encapsulation order is: Security Encapsulation – Multi Channel Encapsulation – Security Commands Supported Get Command

When communicating with a device that supports multiple Multi Channel End Points, the Security Encapsulation MUST be added outside of the Multi Channel Command Class. Thus, a receiving node MUST first remove the Security Encapsulation and then forward it to the actual destination Multi Channel End Point.

- A Multi Channel End Point MUST be considered as a separate device, with separate NIF – given by Multi Channel Capability Report and Security Commands Supported Report.
- Multi Channel End Points are logical abstractions. Only the Root Device is included in the network.

This means:

- Inclusion always deals with the Root Device.
- A Security Command Support Get must reply as a Root Device. If the Multi Channel Command Class is not supported non-securely, it will only be listed in the Security Command Supported Report.
- The Multi Channel Capability Report MUST advertise the Security 0 Command Class as supported for all End Points that implement command classes that are supported securely. It has been found that legacy nodes do not always advertise the S0 Command Class in their Multi Channel Capability Report and still accept all their Command Class using S0 encapsulation. A controlling node SHOULD try to control End Points with S0 encapsulation even if S0 is not listed in the Multi Channel Capability Report.
- The implicit rule that all non-secure command classes for an End Point must be controllable securely is still in effect, if the endpoint is reported secure.
- An End Point only inherits the security capabilities of the End Point itself. I.e. each End Point is considered a device itself.

3.5.4.2 Security Commands Supported Get Command

This command is used to query the commands supported by the device when using secure communication.

The Security Commands Supported Report Command MUST be returned in response to this command.

A node MAY choose only to advertise a Command Class as 'supported' and/or 'controlled', when secure communication is used. In that case the Command Class MUST NOT be advertised in the NIF, while it MUST be advertised in the Security Commands Supported Report Command.

Secure communication MUST be used when transmitting this command.

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

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY							
Command = SECURITY_COMMANDS_SUPPORTED_GET							

3.5.4.3 Security Commands Supported Report Command

This command advertises which command classes are supported using security encapsulation..

- All non-securely supported command classes MUST NOT be advertised in the Security Commands Supported Report.
- All securely supported command classes MUST be advertised in the Security Commands Supported Report if they are only supported securely.

Secure communication MUST be used when transmitting this 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 MAY be mixed.

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 bits)

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

Command Class (N * 8 or 16 bits)

The Command Class identifier.

Command Class Mark (8 bits)

The COMMAND_CLASS_MARK is used to indicate that all preceding command classes are supported, and all following command classes are controlled.

3.6 Security 2 (S2) Command Class, version 1

The Security 2 Command Class is a framework for allowing nodes to communicate securely in a Z-Wave network.

The Security 2 Command Class provides backwards compatibility to nodes implementing the Security 0 Command Class. Security 2 Command Class also defines a new encapsulation format, new Security Classes and a new KEX Scheme 1, which together offers a number of advantages over the Security 0 Command Class. Security 2 Command Class is scalable and allows more KEX Schemes, Security Classes and encapsulation formats to be introduced in the future if necessary.

Communication may be protected for a number of purposes. Known as CIA, the three main areas addressed by communications security are Confidentiality, Integrity and Authenticity. Confidentiality ensures that only the recipient can decode the communication. Integrity ensures that the recipient can determine if the communication has been modified or replayed. Authentication ensures that the communication really comes from the advertised sender.

Security 2 provides Confidentiality, Integrity and Authentication through:

- Key Exchange, that allows distribution of Network Keys in a Secure Network while preventing interception through:
  - Out-of-Band verification
  - Narrow time windows
  - Physical Activation
- Secure Message Encapsulation between nodes that have a shared network key.
  - A Pre-Agreed Nonce (PAN) is used to prevent Replay Attacks where communication is recorded and played back later, e.g. to unlock a door. In S2 a Nonce is exchanged once and then used to compute subsequent Singlecast PANs (SPAN) and Multicast PANs (MPAN), respectively.
  - Secure Singlecast communication, supports one-frame secure messages; allowing for lower latency and faster response times.
  - Secure Multicast communication, allows a sending node to simultaneously send secure messages to multiple nodes.
  - Cryptographic algorithms are intimately dependent on a perfect Pseudo Random Number Generator (PRNG). The PRNG is seeded with a unique noise pattern to ensure a unique starting point in the long sequence of random numbers generated by the PRNG. Many radio systems feature a special mode where the radio is capable of generating white noise which is not affected by the RF signal received from the antenna.

3.6.1 Compatibility Considerations

3.6.1.1 Command Class dependencies

CC:009F.01.00.21.001 Nodes supporting the Security 2 Command Class MUST also support the following command classes:

- Transport Service Command Class, version 2
- Supervision Command Class

CC:009F.01.00.21.002 In addition, nodes based on a controller Role Type MUST support the following Command Class:

- Inclusion Controller Command Class [15]

3.6.1.2 Node Information Frame (NIF)

CC:009F.01.00.21.008 A supporting node MUST always advertise the Security 2 Command Class in its NIF, regardless of the inclusion status and security bootstrapping outcome.

CC:009F.01.00.21.00C A supporting node MUST NOT advertise the Security 2 Command Class in its S0/S2 Commands Supported Report list.

3.6.1.3 Mixed Security Classes

With the advent of the Security 2 Command Class, three new security classes have been added to the existing non-secure and Security 0. This makes for a total of five different security levels of which neither can communicate directly with each other.

In Security 0, this could be alleviated to some extent by allowing a device to choose to support certain of its command classes as non-secure. However, the impact of this is that a device is not entirely secure.
CC:009F.01.00.22.001 For this reason, command classes SHOULD NOT be supported non-securely by S0 enabled nodes if they leak information about the state of the device.

CC:009F.01.00.21.00B In Security 2, a node MUST support its command classes only when communication is using its highest Security Class granted during security bootstrapping.
The above rule does not apply to certain command classes, such as Transport Service or Z-Wave Plus Info, which must always be supported non-securely and present in the NIF if they are supported by a node. In this case, non-secure support requirements are specified in each individual command class definition. Command Classes present in the NIF (supported non-securely) MUST be supported at any granted Security Class level unless they are encapsulated outside security encapsulation.

CC:009F.01.00.23.002 To allow inter-device communication for different security classes, the SIS (or Primary Controller) MAY perform Security Class elevation, by working as a middle-man and thus elevating the Security Class of a sending device to reach a different Security Class on a receiving device.
CC:009F.01.00.21.005 In case a forwarding rule is created from a lower Class to a higher Class, the UI MUST issue a warning to the user.

CC:009F.01.00.23.003 A node supporting S2 MAY control Command Classes at any of the granted Security Classes.

CC:009F.01.00.22.002 A controlling node attempting to communicate with a supporting node SHOULD try using its highest Security Class. If the communication is not successful and the controlling node has been granted several
CC:009F.01.00.23.004 Security Classes, the controlling node MAY try using any lower Security Classes.

3.6.1.4 Migration of existing devices to the Security 2 Command Class

An included node may be upgraded to support S2 via an OTA firmware update. Since non-secure operation as well as the Security 0 Command Class provide a lower level of trust, it is not possible to
CC:009F.01.00.21.006 automatically switch to S2 protection. Instead the Device Specific Key (DSK) MUST be generated by the

CC:009F.01.00.21.007

running firmware on the device after being updated. Having the DSK generated internally, means it is not possible to input it directly on the S2 bootstrapping controller (for Access Control and Authenticated Security Classes), instead it MUST be possible to input the DSK of the S2 bootstrapping controller on the joining node.

This process is described in further details in 3.6.6.3.

3.6.2 Security Considerations

3.6.2.1 Application enabled delivery confirmation

The use of SPAN and MPAN enables secure communication without preparations for each message. It is powerful, yet it requires application awareness. Safety and security related applications like door locks may require immediate command confirmations via the Supervision Command Class. Further, the risk of a delay attack can be mitigated through the use of the Supervision Command Class. This applies to S2 Singlecast as well as S2 Multicast transmissions.

A firmware update process may prefer to transfer firmware fragments as fast as possible while accepting the minor risk that the process stops for a moment in the unlikely event that the SPAN needs to be updated by the transmitter.

3.6.2.2 Potential Singlecast Delay Attack via interception and jamming

Since the SPAN is not limited by a timeout or synchronized clock, it is possible to perform a delay attack by intercepting an encrypted message while at the same time jamming the intended receiver. This way, the receiver SPAN is not incremented and the receiver will accept the encrypted message when an attacker decides to transmit the delayed message. This attack further requires that the attacker returns a MAC layer acknowledgement to the sender to avoid that the user gets error messages.

CC:009F.01.00.42.001 The Supervision Command Class SHOULD be used for S2 delivery acknowledgement. Only the intended receiver can respond correctly to a Supervision Get command.

3.6.2.3 Potential Multicast Delay Attack

Since the MPAN is not limited by a timeout or synchronized clock, it is possible to perform a delay attack by intercepting an encrypted message while at the same time jamming the intended receivers. This way the receiver MPAN is not incremented and the receivers will accept the encrypted message when an attacker decides to transmit the delayed message.
There is no way to distinguish this attack from simple radio interference phenomena as S2 Multicast messages are not acknowledged on the Z-Wave MAC layer.

While the singlecast follow-up message is primarily intended to ensure command execution in all multicast group members, the message also makes the receiver increment its MPAN inner state to invalidate previous multicast messages. This effectively eliminates an attacker's options for mounting a multicast delay attack.

S2 Multicast and singlecast follow-up messages are described in 3.6.5.2.

3.6.2.4 Circumventing DSK authentication

3.6.2.4.1 Controller-side authentication

The first 2 bytes of the public key of the joining node are obfuscated when requesting access to the "S2 Access Control" or "S2 Authenticated" class. The purpose of the DSK validation procedure is to force the user to enter information that can only be gathered from the joining device.

In other words, a user entering a part of the DSK proves to the including controller that the joining node is indeed the node that the user wants to add to the network. With this information, the including controller can construct the full public key of the joining node.
It is theoretically possible for the including controller to guess the complete public key of the joining node in less than 65536 calculations without user interaction. The including controller needs to try decrypting the received frame until a valid KEX Set (Echo) command is found.

The DSK verification is a device authentication step that ensures that a joining node can be trusted. The security of the system does not depend on the public key being secret but an including controller that skips authentication runs the risk of handing out network keys to joining nodes that cannot be trusted. A man-in-the-middle attacker may establish a shared secret with the joining node by using the joining node's public key but the user's including controller will detect a mismatch between the DSK of the joining node and the first 2 or 16 bytes of the attacker's public key. Therefore the including controller rejects to complete the security bootstrapping before the network key is exposed to the attacker.

This applies to the S2 Access Control Class as well as the S2 Authenticated Class.

3.6.2.4.2 Client-side authentication

When upgrading existing devices to support Security 2 through an over-the-air (OTA) firmware update, there is no DSK printed on the node, which is required for S2 bootstrapping of the S2 Authenticated and S2 Access Control Classes.

In this case, a reverse verification procedure can be carried out, where the controller obfuscates the first 4 bytes of its public key and the joining node is input the controller's DSK in order to perform the authentication.

3.6.2.5 Protecting keys from physical extraction

A device may be mounted in an outdoor or public location. In such locations, there is a risk that the device is removed physically.

CC:009F.01.00.42.002 The hardware of the device SHOULD be designed to prevent the read-back of ECDH keys and network keys via debug connectors.

CC:009F.01.00.42.003 The hardware of the device SHOULD be designed to prevent the read-back of ECDH keys and network keys via a malicious firmware image.

CC:009F.01.00.42.004 The non-volatile memory used for storing security keys SHOULD be automatically cleared in case a firmware image is programmed in the NVM via the debug interface; only allowing keys to survive if firmware update is handled entirely via internal software APIs.

3.6.3 Interoperability considerations

3.6.3.1 Pragmatic Decryption calculations in constrained environments

This specification recommends that a receiving node accepts incoming S2 Multicast frames which are up to 4 iterations into the future relative to the current MPAN inner state for the actual Group ID.

This specification also recommends that a sending node issues S2 Singlecast Follow-up frames after sending S2 Multicast frames.

Decryption calculations may put a significant load on constrained processors. Thus, the receiving node may still be trying to decrypt a received S2 Multicast frame when an S2 Singlecast Follow-up is received.

CC:009F.01.00.31.001 A receiving node MUST respond correctly to an S2 Singlecast frame received immediately after an S2 Multicast frame.

CC:009F.01.00.32.001 A receiving node SHOULD monitor the arrival of S2 Multicast frames in order to avoid repeated MPAN synchronization in conditions where the S2 Multicast frame is only rarely received correctly.

3.6.4 Building Blocks

S2 functionality is relying on a number of building blocks for different parts of the protocol.

3.6.4.1 ECDH Key pair generation

Each S2 node has one or more ECDH key pairs used to setup a temporary secure channel for the Network Key exchange. Key pair generation is described in [27].

CC:009F.01.00.11.09D The ECDH private key MUST be created from 32 random bytes, which are generated using the PRNG function (3.6.4.6).
The public key is calculated from the private key using Curve25519 [27].

3.6.4.2 Key exchange overview

In the following, the term Node A refers to the including node (typically a primary controller or SIS) while the term Node B refers to the joining node.

- **Inclusion Step 1: Create a shared secret between Node A and Node B**
  Both nodes calculate a shared secret based on an Authenticated Elliptic Curve Diffie Hellman key exchange (*AuthECDH*). Node A takes as input the Public Key of B, `KeyPub_B` and its own Private Key, `KeyPriv_A`. Node B takes as input the Public Key of A, `KeyPub_A` and its own Private Key, `KeyPriv_B`. Both returning the same `ECDH Shared Secret`.
  - *AuthECDH* is based on ECDH using Curve25519 [27]. Authentication is achieved through user verification as specified in Section 3.6.6.2.
- **Inclusion Step 2: Derive shared symmetric key for key exchange**
  To establish a temporary Network Key for AES128-CCM and CTR_DRBG, two steps are needed:
  - To convert the `ECDH Shared Secret` into a 16-byte Pseudo Random Key (`PRK`). *`CKDF-TempExtract`* takes as input the `ECDH Shared Secret` along with `KeyPub_A` and `KeyPub_B`.
  - Temporary symmetric keys are derived based on CKDF-TempExpand, by giving the `PRK`, `KeyPub_A` and `KeyPub_B` as input. This returns the following keys:
    - Temporary CCM Key, combined Encryption and Authentication Key, denoted `TempKeyCCM`
    - Temporary Personalization String, denoted `TempPersonalizationString`.
- **Inclusion Step 3: Exchange permanent Network Keys**
  To exchange one or several Permanent Network Key (`PNK`), Singlecast Message Encapsulation is used with temporary symmetric derived keys (`TempKeyCCM` and `TempPersonalizationString`).
  - All Permanent Network Key Exchanges are carried out using the temporary symmetric key.
  - All Permanent CCM Keys, `KeyCCM`, `KeyMPAN` and `PersonalizationString`, are derived from the corresponding `PNK` using *`CKDF-NetworkKeyExpand`*
  - All *`CKDF`* functions are based on `AES128-CMAC`.
- **Singlecast Message Encapsulation:**
  The algorithm uses an authenticated encryption scheme conforming to AES128-CCM [24]. It is used to encrypt and authenticate secure payloads. AES128-CCM takes the `KeyCCM` and a `Nonce` as input. The algorithm returns a CCM Authenticated Ciphertext.

- A Nonce is a “Number used Once”. Nonces are initially exchanged between the two nodes and then mixed into a Mixed Entropy Input, `MEI`, using *`CKDF-MEI-Extract`* and *`CKDF-MEI-Expand`*. With both nodes holding the same `MEI`, they use the `MEI` and `PersonalizationString` as input into CTR_DRBG to generate a new Nonce.

- **Multicast Message Encapsulation:**
  Both singlecast and multicast frames use AES-128 CCM for encryption and authentication but multicast frames use a different algorithm to generate the IV. For multicast, the CCM IV is called the Multicast Pre-Agreed Nonce (`MPAN`).

CC:009F.01.00.11.001 MPANs are pushed from Node A, to multiple receiving nodes, B1, B2, etc. MPANs MUST be generated by Node A (see section 3.6.4.10) and MUST be distributed securely to each node B1, B2 etc. using singlecast messages.

CC:009F.01.00.11.002 Implementations of the Security 2 Command Class MUST provide AES-128 cryptographic services in the following modes of operation:

- **AES-128 “RAW”**
  This is also known as the AES-Electronic Code Book (ECB) and defines the basic operation of encrypting a 16 Byte Plaintext into a 16 Byte Ciphertext using an encryption key. AES-128 is used as a foundation for the following modes.

- **AES-128 CCM**
  The Counter with CBC-MAC (CCM) [24] is an authenticated encryption scheme.

- **AES-128 CMAC**
  The Cipher based Message Authentication Code (CMAC) is an essential part of the Key Derivation functions and used to mix Nonce contributions for SPAN synchronization.

- **AES-128 CTR_DRBG**
  The Counter mode Deterministic Random Byte Generator (CTR_DRBG) [25] is a block cipher based Pseudo Random Number Generator (PRNG) that is used to create Initialization Vectors and Network Keys.

CC:009F.01.00.11.003 In addition to the AES modes, the following building blocks MUST also be provided:

- **AuthECDH**
  Authenticated Elliptic Curve Diffie Hellman key exchange. Provides the temporary shared key material for protecting the network key exchange during inclusion.

3.6.4.3 Core AES (AES)

CC:009F.01.00.11.004 The security layer MUST implement the AES-128 encryption algorithm. This algorithm encrypts a single 128-bit block of plaintext using the Advanced Encryption Standard, AES. It receives a 128-bit key and a 128-bit plaintext block as input and produces a 128-bit cipher text block.

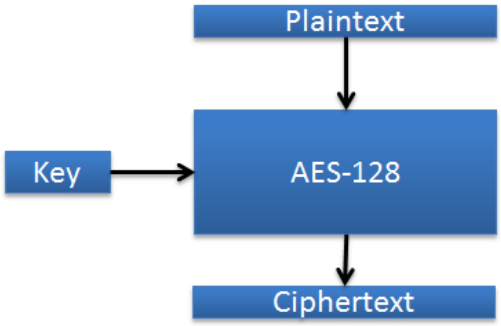

Figure 10, AES-ECB (Electronic Code Book)

3.6.4.4 AES-128 CCM Encryption and Authentication

CC:009F.01.00.11.005 The payload in the S2 Message Encapsulation Command MUST be encrypted and authenticated using AES-128 CCM. For details about AES-128 CCM and message decryption and validation, refer to [24] and [26].

CCM takes the following input:

- A combined encryption and authentication key `denoted KeyCCM`
- A Nonce
- A variable-length additional authenticated data structure (AAD)
- A variable-length payload to encrypt and authenticate

CCM yields as output:

- An encrypted version of the payload combined with an authentication tag covering the payload and the AAD

3.6.4.4.1 CCM profile

CC:009F.01.00.11.006 [26] defines a number of parameters that together determine the CCM profile used. S2 nodes MUST use the following parameter values for the CCM profile:

- Additional Authenticated Data (AAD) MUST be used (Adata = 1)
  The actual AAD is defined in section 3.6.4.4.2.
- The Length field MUST be 2 bytes long (L = 2 bytes)
- The Authentication tag length MUST be 8 bytes (M = 8 bytes)
- The Nonce length MUST be 13 bytes (N = 13 bytes)

CC:009F.01.00.11.007 The following length requirements MUST be observed:

- AAD structures of up to 30 bytes MUST be supported. Larger AAD structures MAY be supported.

CC:009F.01.00.11.008 The 13 byte Nonce MUST be generated by taking the 13 most significant bytes of the *NextNonce* or Nonce0 as described in section 3.6.4.8.

3.6.4.4.2 Additional Authenticated Data (AAD)

CC:009F.01.00.11.009 The following data structure MUST be used as AAD input for each CCM operation.

7	6	5	4	3	2	1	0
Sender NodeID							
Destination Tag							
HomeID Byte 1							
…							
HomeID Byte 4							
Message Length Byte 1 (MSB)							
Message Length Byte 2 (LSB)							
Sequence Number							
Reserved						Enc Ext	Ext
Extension Data 1							
…							
Extension Data M							

Sender NodeID (1 byte)

NodeID of the sending node.

Destination Tag (1 byte)

The use of this field depends on the actual frame.

CC:009F.01.00.11.00A If the field is used for a Singlecast frame, this field MUST carry the Receiver NodeID.
If the field is used for an S2 Multicast frame, this field MUST carry the S2 Multicast Group ID.

HomeID (4 bytes)

HomeID of the sending node.

Message length (2 bytes)

This field indicates the total length in bytes of the Security 2 Message Encapsulation Command.

Sequence Number (1 byte)

Refer to the Security 2 Message Encapsulation Command (3.6.5.3.3).

Ext (1 bit)

Refer to the Security 2 Message Encapsulation Command (3.6.5.3.3).

Enc Ext (1 bit)

Refer to the Security 2 Message Encapsulation Command (3.6.5.3.3).

Reserved

Refer to the Security 2 Message Encapsulation Command (3.6.5.3.3).

Extension Data (M bytes)

CC:009F.01.00.11.00B This field MUST contain all non-encrypted extension objects.

CC:009F.01.00.11.00C This field MUST include the Length and Type fields prepending the actual data of each extension.

3.6.4.5 Message Authentication Code – AES-128 CMAC

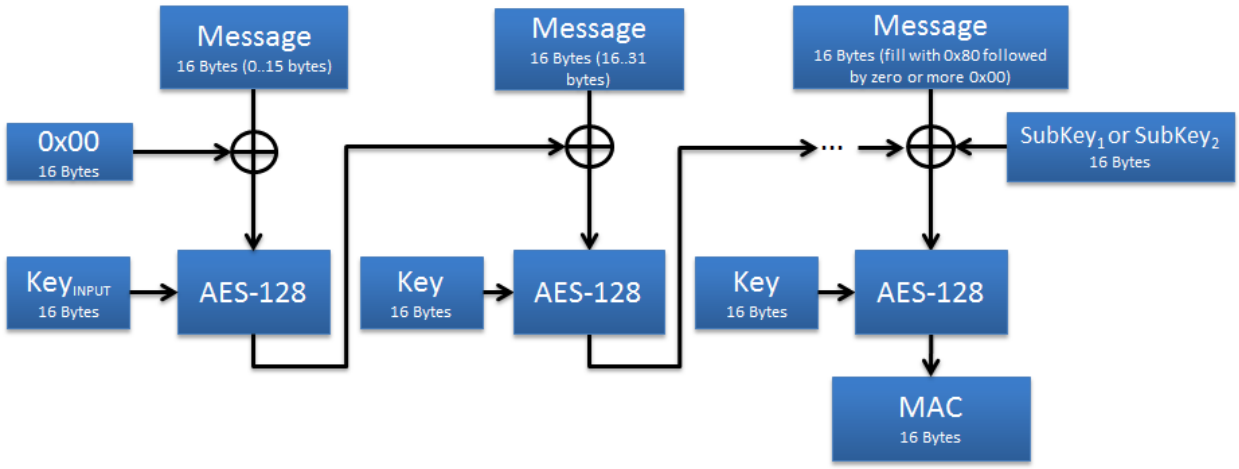

Figure 11, AES-128 CMAC building blocks

AES-128 CMAC is used for key derivation and Nonce mixing. The CMAC operation is specified in [23].

3.6.4.6 Pseudo Random Number Generator (PRNG)

CC:009F.01.00.11.015 The PRNG MUST be used for:

- Generating new network keys when provisioning a new network.
- Generating Nonce contributions for synchronizing the SPAN with peer nodes.

CC:009F.01.00.11.016 The PRNG MUST be implemented as an AES-128 CTR_DRBG as specified in [25]. The following profile MUST be used:

- No derivation function
- No reseeding counter
- Personalization string of 0x00 repeated 32 times
- Output length = 16 bytes
- security_strength is not used

CC:009F.01.00.11.017 The entropy_input [25] for instantiating the PRNG MUST be generated by a truly random source, e.g. white radio noise. The PRNG MUST be hardware seeded.

CC:009F.01.00.11.018 The inner state of the PRNG MUST be separated from the SPAN table.

3.6.4.7 Key extraction and derivation

The functions described in this section are used during key exchange.

3.6.4.7.1 CKDF-TempExtract

The *CKDF-TempExtract* function is used to extract the key entropy from the non-uniformly distributed ECDH Shared Secret.

CKDF-TempExtract(Constant$_{PRK}$, ECDH Shared Secret, KeyPub_A, KeyPub_B) -> PRK

- The function's input is defined by:
  - Constant$_{PRK}$ = 0x33 repeated 16 times
  - ECDH Shared Secret is the output of the ECDH key exchange
  - Public Keys of Nodes A and B
  - PRK = CMAC(Constant$_{PRK}$, ECDH Shared Secret | KeyPub_A | KeyPub_B)

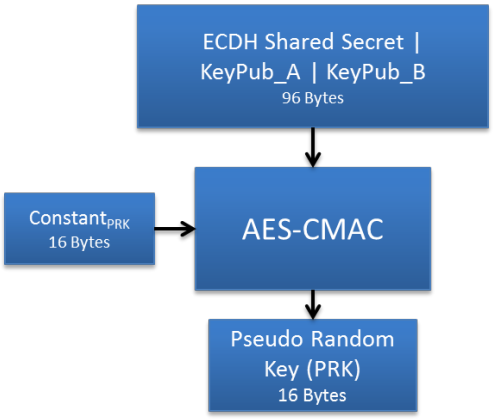

Figure 12, CKDF-TempExtract function block diagram

3.6.4.7.2 CKDF-TempExpand

CC:009F.01.00.11.08D Once the PRK has been computed, the temporary Authentication, Encryption and Nonce Keys MUST be derived using the *CKDF-TempExpand* function [22].

CKDF-TempExpand(PRK, Constant$_{TE}$) -> {TempKeyCCM, TempPersonalizationString}

- The function's input is defined by:
  - PRK is calculated in the previous section 3.6.4.7.1
  - Constant$_{TE}$ = 0x88 repeated 15 times
- Calculations are performed as follows:
  - T1 = CMAC(PRK, Constant$_{TE}$ | 0x01)
  - T2 = CMAC(PRK, T1 | Constant$_{TE}$ | 0x02)
  - T3 = CMAC(PRK, T2 | Constant$_{TE}$ | 0x03)
- Output is defined as follows:
  - TempKeyCCM = T1. Temporary CCM Key, combined Encryption and Authentication Key.
  - TempPersonalizationString = T2 | T3

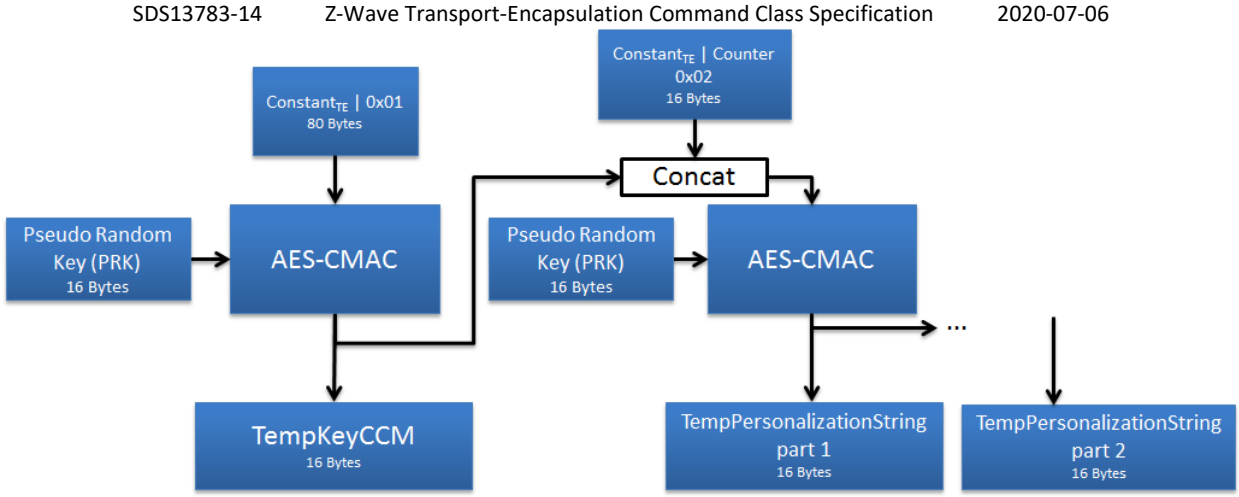

Figure 13, CKDF-TempExpand function block diagram

3.6.4.7.3 Permanent Key Exchange

A node that has been security bootstrapped into the network is characterized by being in possession of one or more Network Key(s). This key(s) is used throughout the lifetime of the network, typically measured in years to decades. Different nodes MAY have different Class Keys. A central controller node MUST manage all the keys and select at inclusion time which class key(s) to share with a given node.

CC:009F.01.00.13.014
CC:009F.01.00.11.08E

CC:009F.01.00.11.08F Network Keys for all Security 2 Classes MUST be 16 random bytes, generated by using the PRNG function described in section 3.6.4.6.

3.6.4.7.3.1 Key Derivation

CC:009F.01.00.11.090 Once the Network Key(s) has been exchanged, the `KeyCCM`, `PersonalizationString` and `KeyMPAN` values MUST be derived using the *CKDF-NetworkKeyExpand* function [22].

`CKDF-NetworkKeyExpand (PNK, Constant`$_{NK}$`) -> {KeyCCM, PersonalizationString, KeyMPAN}`

- The function's input is defined by:
  - `PNK is the permanent network key.`
  - `Constant`$_{NK}$ `= 0x55 repeated 15 times`
- Calculations are performed as follow:
  - `T1 = CMAC(PNK, Constant`$_{NK}$ `| 0x01)`
  - `T2 = CMAC(PNK, T1 | Constant`$_{NK}$ `| 0x02)`
  - `T3 = CMAC(PNK, T2 | Constant`$_{NK}$ `| 0x03)`
  - `T4 = CMAC(PNK, T3 | Constant`$_{NK}$ `| 0x04)`
- Output is obtained by:
  - `KeyCCM = T1; CCM Key, combined Encryption and Authentication`
  - `PersonalizationString = T2 | T3`
  - `KeyMPAN = T4`

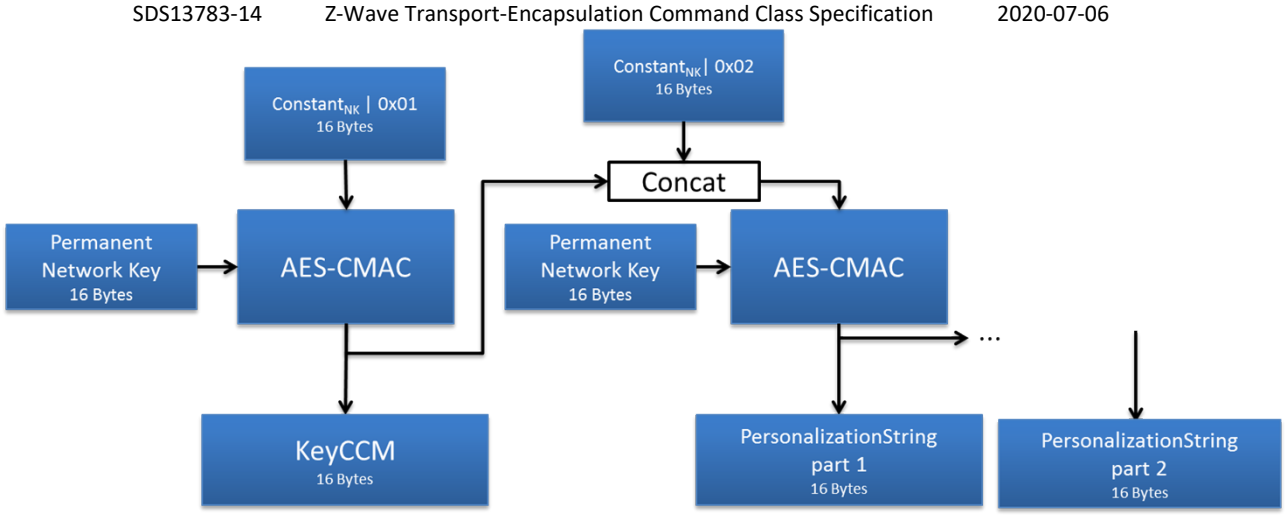

Figure 14, CKDF-NetworkKeyExpand function block diagram

3.6.4.8 Nonces for CCM

The Nonce is used for the CCM encryption in a Security 2 Message Encapsulation Command. Nonces provide these security benefits:

- Replay protection. Replaying an old message will result in the replay being discarded because the Nonce has already been used.
- The security proofs for CCM rely on the assumption that the same plaintext is never encrypted under the same key twice. Having unique Nonces upholds this assumption.

Singlecast and Multicast transport use different mechanisms for generating Nonces as described in the following sections.

Singlecast Pre-Agreed Nonces are referred to as `SPAN`.

Multicast Pre-Agreed Nonces are referred to as `MPAN`.

3.6.4.9 SPAN NextNonce Generator

CC:009F.01.00.11.00D Singlecast Nonces MUST be generated in the following way:

Skip steps 1 through 3 if a SPAN is already established

1. Exchange 16 bytes of Entropy between the Sender and the Receiver
2. Mix the entropy contributions
3. Instantiate CTR_DRBG and store the working state in the SPAN table
4. Generate 13 bytes of `Nonce` from the established `SPAN` using the *`NextNonce function`*.
5. Save the updated working state in the SPAN table

3.6.4.9.1 SPAN Instantiation

CC:009F.01.00.11.00E The Sender and Receiver MUST instantiate the CTR_DRBG as follows:

1. The Sender and Receiver MUST exchange 16 bytes of Entropy Input (EI), resulting in 32 bytes of shared entropy
   a. Both contributions MUST be generated using the PRNG (see Section 3.6.4.6)
2. Mix the 32 bytes EI into `MEI`, using *`CKDF-MEI-Extract`* and *`CKDF-MEI-Expand`* `functions`

CC:009F.01.00.11.00F The CTR_DRBG MUST be instantiated using the following profile:

a. Entropy Input = `MEI (obtained with` *`CKDF-MEI_Expand`*`)`
b. Personalization_String = `PersonalizationString`
c. Output length = 16
d. No derivation function
e. No reseeding counter

f. No Security_strength

3. The CTR_DRBG now has its inner state set, referred to as the `InnerSPAN`

3.6.4.9.1.1 Mixing

Mixing of the two EIs is done by first calling *`CKDF-MEI-Extract`* with the EIs as input and using the result as input for *`CKDF-MEI-Expand`*

3.6.4.9.1.1.1 CKDF-MEI-Extract

```
CKDF-MEI-Extract(ConstNonce, SenderEI | ReceiverEI) -> NoncePRK
```

- The Input is defined by:
  - `ConstNonce = 0x26 repeated 16 times`
- The Output is obtained by:
  - `NoncePRK = CMAC(ConstNonce, SenderEI | ReceiverEI)`

3.6.4.9.1.1.2 CKDF-MEI-Expand

```
CKDF-MEI-Expand(NoncePRK, ConstEntropyInput) -> MEI
```

- The Input is defined by:
  - `NoncePRK is the pseudo random value obtained in the Extract step.`
  - `ConstEntropyInput = 0x88 repeated 15 times`
- The Output is obtained by:
  - `T0 = ConstEntropyInput | 0x00`
  - `T1 = CMAC(NoncePRK, T0 | ConstEntropyInput | 0x01)`
  - `T2 = CMAC(NoncePRK, T1 | ConstEntropyInput | 0x02)`
  - `MEI = T1 | T2`

3.6.4.9.2 Generation

CC:009F.01.00.11.010 With the CTR_DRBG having its inner state set, new `SPANs` MUST now be generated by subsequent calls to the *`NextNonce`* function.

3.6.4.9.2.1 NextNonce

The *`NextNonce`* function is defined as the CTR_DRBG_Generate_algorithm [25] with the following parameters:

- working_state = `InnerSPAN`
- requested_number_of_bits = 128
- additional_input = “” (empty string, i.e. zero bytes)

CC:009F.01.00.11.011 The *NextNonce* function MUST return a new `SPAN` for each call. The `InnerSPAN` MUST be stored in the SPAN table as described in Section 3.6.5.1.1.

CC:009F.01.00.11.012 The 16 bytes of Nonce MUST be truncated to the 13 most significant bytes before passing to the CCM module.

3.6.4.10 MPAN NextNonce Generator

CC:009F.01.00.11.013 Multicast Pre-Agreed Nonces (MPAN) are generated by encrypting successive values of a counter. The initial value MUST be a 16 byte random number generated by the PRNG (3.6.4.6). Whenever an MPAN is generated and consumed by the CCM module, the inner MPAN state is incremented.

MPAN instantiation and synchronization is described in section 3.6.5.2.

3.6.4.10.1 MPAN Generation

MPAN generation is needed when a node sends or receives a secure multicast message. The generation
CC:009F.01.00.11.014 procedure, called *NextMPAN*, MUST comply with the following description:

1. The node reads the 16-byte inner MPAN state N from the MPAN table.
2. The node performs an *AES-128-ECB* encryption of the inner MPAN state N using KeyMPAN (obtained during key derivation 3.6.4.7.3.1) as key. The result is MPAN N.
3. The node increments the inner MPAN state N. The result is the inner MPAN state N+1 which is stored in the MPAN table. The increment operation MUST treat the inner MPAN state as an unsigned 16-byte integer. Thus, incrementing all ones MUST yield all zeros.

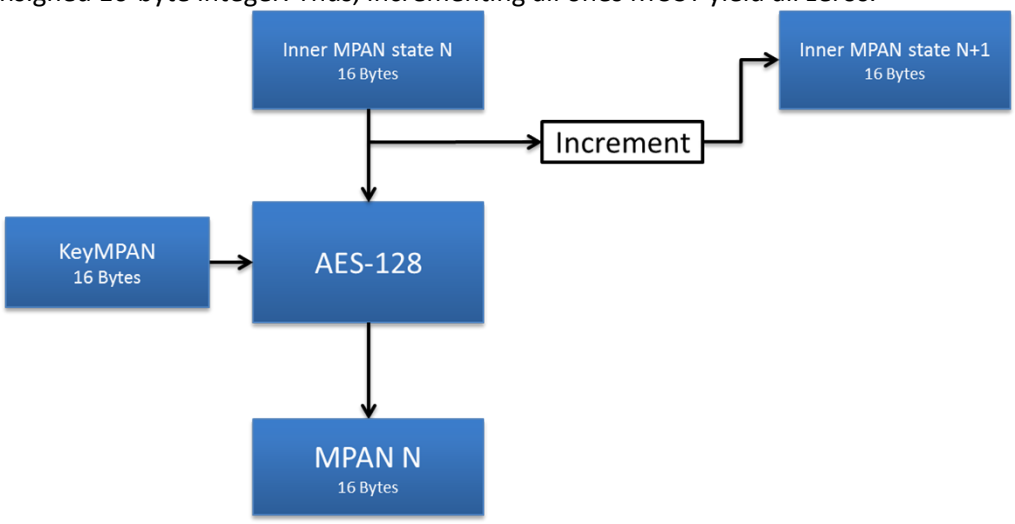

Figure 15, MPAN generation

3.6.5 Message Encapsulation

The Security 2 Command Class supports Singlecast as well as Multicast communication

CC:009F.01.00.11.019 The S2 Transport Layer MUST NOT provide retransmission if the security layer discards a message due to SPAN synchronization failure or failed authentication.

CC:009F.01.00.12.001 An application SHOULD use the Supervision Command Class for delivery acknowledgement of Security 2 Encapsulated commands.

CC:009F.01.00.12.002 The Supervision Report command returns high-level status information on the execution status of the transmitted command which SHOULD be used by a controlling application instead of polling the destination node repeatedly.

3.6.5.1 Singlecast messages and SPAN Management

Compared to the Security 0 Command Class, the Security 2 Command Class provides a more efficient way to communicate securely. The Security 2 Command Class eliminates the frame overhead of the Security 0 Command Class challenge-response handshake.

Singlecast transport MUST comply with the steps described below:

CC:009F.01.00.11.01A
CC:009F.01.00.11.01B

1. Before sending an encrypted frame, the **Sender** MUST establish a Singlecast Pre-Agreed Nonce (SPAN) with the **Receiver** if a SPAN is not already established.
    a. If a SPAN exists between the two nodes:
        i. Skip to step 3
    b. If the Sender is in possession of a matching Receiver's Entropy Input:

i. The **Sender** uses its PRNG to generate a random 16-byte Nonce (SEI).

ii. The **Sender** uses the combined 32-byte Sender's and Receiver's Entropy Input to instantiate a *NextNonce* Generator, hence creating the inner SPAN state. (described in 3.6.4.9)

iii. Skip to step 3

c. If no SPAN or matching Receiver's Entropy Input exists between the two parties:

i. The **Sender** holds back the frame for subsequent transmission.

ii. The **Sender** sends a Nonce Get to the **Receiver**

2. The **Receiver** uses its PRNG to generate a random 16-byte Nonce, store it in the SPAN table and send it in a Nonce Report to the **Sender**, with the Singlecast Out of Sync (SOS) flag set, to indicate that the frame contains a Receiver's Entropy Input (REI).

a. The **Sender** stores the Receiver's Entropy Input in the SPAN table in the entry matching the receiver's NodeID.

b. If the **Sender** has a pending frame for transmission then proceed to step 3. Otherwise no further action is taken.

3. The **Sender** constructs a Security 2 Message Encapsulation Command**.**

a. If the inner SPAN state was just created in step 2.a, the SPAN extension containing the SEI is added to the S2 Message encapsulation Command to allow the **Receiver** to compute the same inner SPAN state.

b. The **Sender** creates the next SPAN with the *NextNonce* function.

c. The **Sender** uses the SPAN as IV for the AES-128 CCM encryption and authentication (3.6.4.4) of the plaintext payload using `KeyCCM`**.**

d. The Security 2 Message Encapsulation Command is transmitted to the **Receiver**.

4. The **Receiver** inspects the received Security 2 Message Encapsulation Command

CC:009F.01.00.12.003

a. If the **Receiver** is unable to authenticate the singlecast message with the current SPAN, the **Receiver** SHOULD try decrypting the message with one or more of the following SPAN values, stopping when decryption is successful or the maximum number of iterations is reached. The maximum number of iterations performed by a receiving node MUST be in the range 1..5.

CC:009F.01.00.11.01C

CC:009F.01.00.11.01D

CC:009F.01.00.11.01E

If the maximum number of iterations is reached without successful decryption, a Nonce Report MUST be sent to the **Sender** with the SOS flag set and containing a new REI**.** At the same time, the **Receiver** MUST invalidate the SPAN table entry for the Sender NodeID.

CC:009F.01.00.11.01F

b. If a SPAN Extension is present, the **Receiver** MUST:

i. Instantiate a new SPAN Generator using the Receiver's Entropy Input stored locally and the Sender's Entropy Input just received.

ii. Store the inner SPAN state in a SPAN table entry with the **Sender** as Peer NodeID.

iii. Generate a SPAN by running the *NextNonce* function on the newly instantiated inner SPAN state.

iv. Attempt authentication with the SPAN.

v. If the authentication succeeds, skip to step 5.

vi. If the authentication fails, the **Receiver** MUST go to step 2.

CC:009F.01.00.11.020

CC:009F.01.00.11.021

c. If the SPAN is already established and a SPAN Extension is not present, the **Receiver** MUST calculate a new inner SPAN state by passing the old inner SPAN state through the *NextNonce* function.

5. After the **Receiver** has successfully authenticated the encapsulation command, the decrypted payload can be presented to the application layer.

The process is also illustrated in Figure 16.

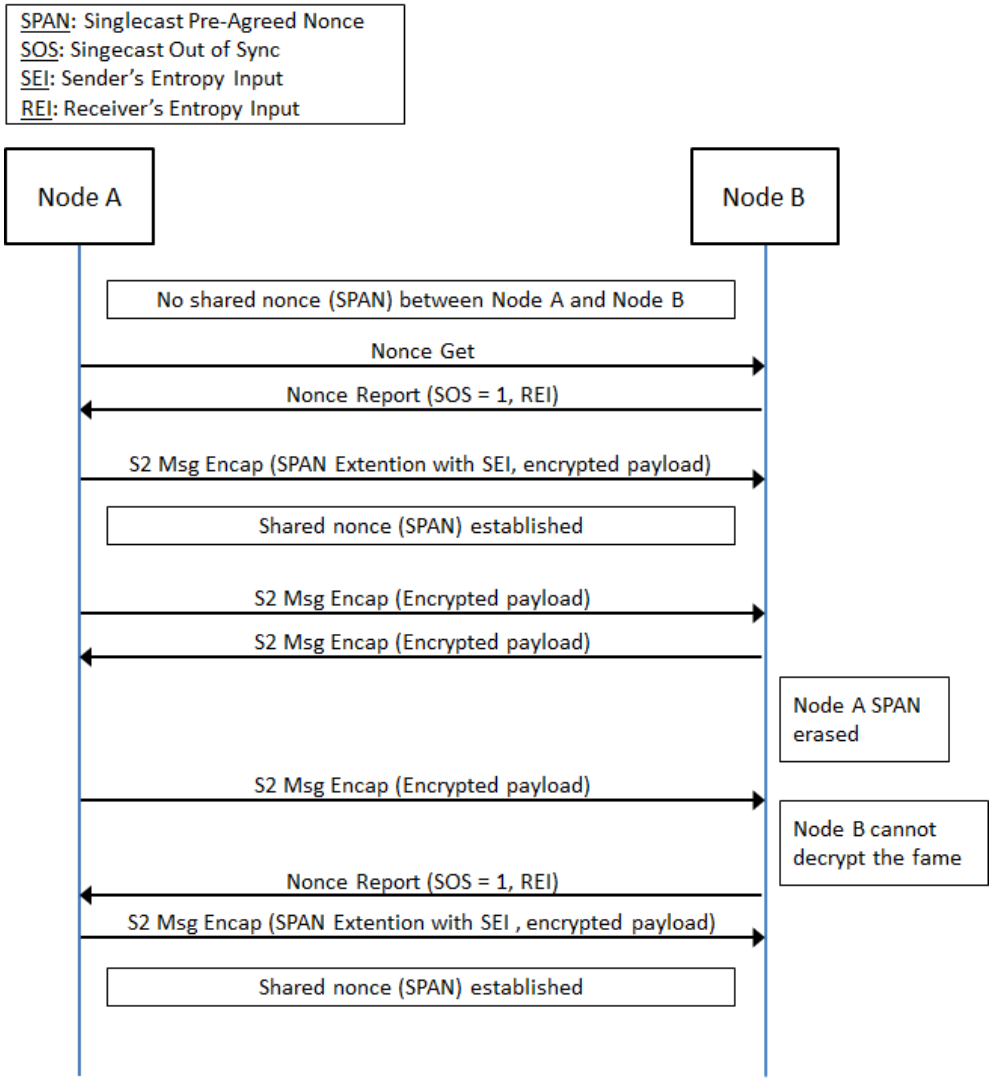

Figure 16 Singlecast communication frame flow: SPAN establishment

3.6.5.1.1 SPAN Table

CC:009F.01.00.11.022 A receiving S2 node MUST accept a SPAN table update even if the SPAN table is full.

CC:009F.01.00.12.004 It is RECOMMENDED that the node discards the least recently used entry from the SPAN table to make room for the new entry.

CC:009F.01.00.11.0A0 A supporting node MUST support at least 1 singlecast session (1 SPAN table entry).
Supporting always listening nodes MUST support at least 5 concurrent singlecast sessions (5 SPAN table entries).

CC:009F.01.00.12.005 The following format is RECOMMENDED:

Table 6, SPAN table format

7	6	5	4	3	2	1	0
Reserved		Security key				Nonce Type	
Sequence Number							
SPAN State Byte 1							
...							
SPAN State Byte 32							
Peer NodeID							

Security key(4 bits)

This field is used to remember which to Security key the SPAN is associated to. An example is given in Table 7.

Table 7, SPAN table::Security key

Value	Description
0x00	ECDH Temporary Key
0x01	S2.2: Access Control
0x02	S2.1: Authenticated
0x03	S2.0: Unauthenticated
0x04	S0

Nonce Type (2 bits)

This field is used to advertise the format of the SPAN State field.

CC:009F.01.00.12.006 The field SHOULD be encoded according to Table 8.

Table 8, SPAN table:: Nonce Type

Nonce Type	Type	Description
0x00	Free	Table entry is not in use.
0x01	Receiver's Entropy Input (Locally generated)	Bytes 1-16 are set to zero Bytes 17-32 contain the Receiver's Entropy Input
0x02	SPAN state	Bytes 1-32 contain the SPAN state

Sequence Number (1 byte)

This field is used to store the sequence number. Refer to description in 3.6.5.3.3 Security 2 Message Encapsulation Command.

CC:009F.01.00.11.024 A receiving node MUST validate the Sequence Number and the actual sender's NodeID before accepting a SPAN table update.

SPAN State (32 bytes)

When the Nonce Type is "SPAN state", this contains the internal state of the *NextNonce* Generator. When the Nonce Type is Receiver's Entropy Input this field contains a locally generated Nonce awaiting the matching Sender's Entropy Input.

Peer NodeID (1 byte)

This field is used to store the NodeID of the corresponding peer.

3.6.5.2 Multicast messages and MPAN Management

The S2 Multicast protocol allows a sending node to reach a group of always listening nodes by using pre-agreed information for authentication and encryption.

The following terminology is used in this section:

- S2 SC = S2 Singlecast (carried in Z-Wave singlecast frame)
- S2 MC = S2 Multicast (carried in Z-Wave broadcast or multicast frame, with the MGRP extension)
- S2 SC-F = S2 Singlecast Follow-up (carried in Z-Wave singlecast frame, with the MGRP extension)

CC:009F.01.00.11.025 An S2 Multicast group MUST be represented by a unique (sender NodeID, Group ID) combination.
CC:009F.01.00.13.001 A multicast sender or group owner MAY maintain multiple S2 Multicast groups.
CC:009F.01.00.11.026 Sending and receiving nodes MUST maintain a unique Multicast Pre-Agreed Nonce (MPAN) value for each Group ID.
CC:009F.01.00.11.027 The MPAN is sender specific, and MUST NOT be used by the receiver to send S2 Multicast frames. S2 Multicast can only be performed within a group of nodes in the same S2 Security Class.

All nodes in the network will receive the S2 Multicast frame and will not try to decrypt it if the MPAN of the corresponding (sender NodeID, Group ID) is not known. The S2 Multicast protocol uses S2 SC-F frames for MPAN synchronization.

CC:009F.01.00.12.007 A sending node SHOULD send an S2 SC-F frame to each S2 Multicast group member after sending an S2 MC frame. This ensures that all group members receive the frame and at the same time eliminates the risk of the S2 MC frame being replayed in a delay attack.

CC:009F.01.00.12.008 S2 SC-F SHOULD be sent frequently in order to ensure that MPAN is synchronized at every group member.

S2 Multicast employs a self-healing algorithm for MPAN synchronization. The reception of an S2 SC-F allows the receiving node to return an MPAN Out of Sync (MOS) indication (Either using a Nonce Report with the MOS flag or a S2 Message Encapsulation with the MOS extension, refer to 3.6.5.3.2, 3.6.5.3.3.1.4) if necessary.

CC:009F.01.00.12.009 Thus, a sending node SHOULD NOT push an up-to-date MPAN before sending an S2 Multicast frame to a new S2 Multicast receiver.

CC:009F.01.00.11.028 A sending node MUST maintain the MPAN inner state according to Table 9.

Table 9, Sending node MPAN maintenance

Frame type	Description	MPAN increase after transmission
S2 Singlecast	S2 SC frame	0
S2 Unacknowledged Multicast	S2 MC frame	1
S2 Acknowledged Multicast	S2 MC frame followed by an S2 SC-F frame to each S2 Multicast group member.	2

CC:009F.01.00.11.029 A receiving node MUST maintain the MPAN inner state according to Table 10.

Table 10, Receiving node MPAN maintenance

Frame type	Description	MPAN increase after reception
S2 Singlecast	S2 SC frame	0
S2 Multicast	S2 MC frame	1
S2 Singlecast Follow-up	S2 SC-F frame	1

CC:009F.01.00.13.002 If the Receiver is unable to decrypt the S2 MC frame with the current MPAN, the Receiver MAY try decrypting the frame with one or more of the subsequent MPAN values, stopping when decryption is successful or the maximum number of iterations is reached.

CC:009F.01.00.12.00A The maximum number of iterations performed by a receiving node MUST be 5.

CC:009F.01.00.11.02A If the maximum number of iterations is reached without successful decryption, the MOS flag MUST be set for the actual Group ID. The MOS state is reported back to the sending node only if the node receives a SC-F for the actual group.

CC:009F.01.00.11.0A1 An always listening node MUST support being member of at least 5 multicast groups at the same time (5 MPAN table entries)

MPAN synchronization and state maintenance examples are given in Figure 17 and Figure 18.

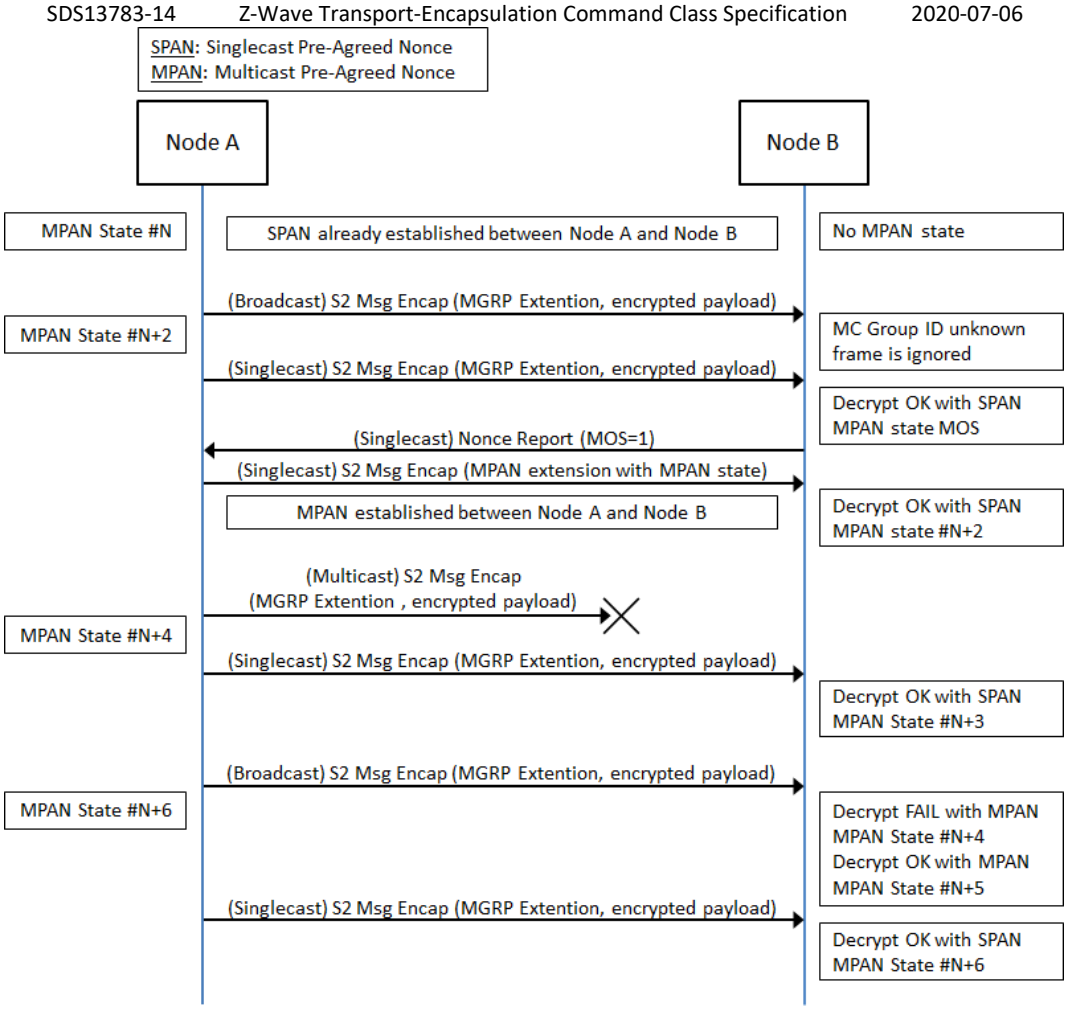

Figure 17, Next MPAN calculation example

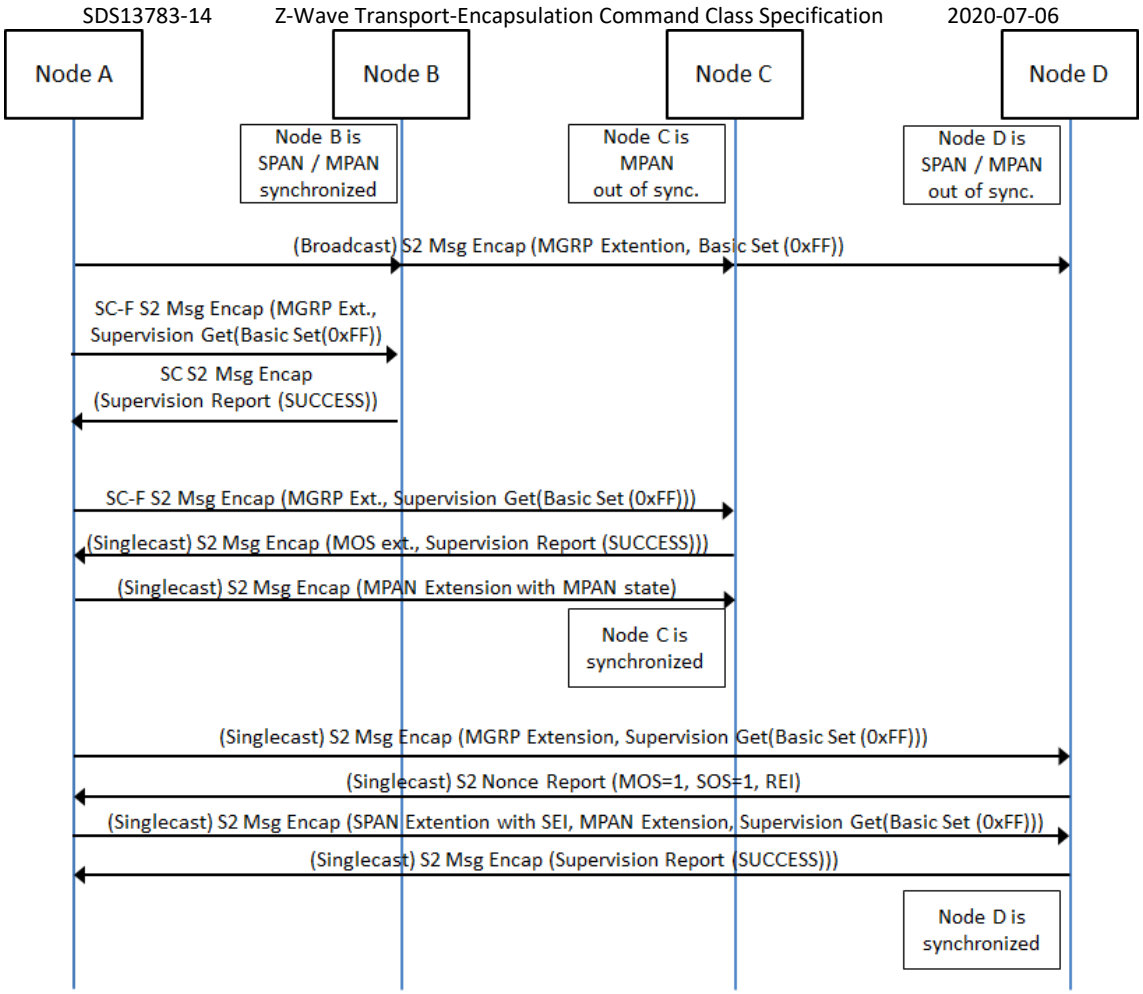

Figure 18, Multicast communication example with receivers out of sync and Supervision

Figure 18 shows Node C returning a Supervision Report with the MOS extension after the Supervision encapsulated Singlecast follow-up. In this case, a node MUST NOT return a Nonce Report with the MOS flag and subsequently another frame carrying the Supervision Report.

CC:009F.01.00.11.0A2

The following behavior is REQUIRED when a node is in MOS state and receives a Singlecast Follow-up:

CC:009F.01.00.11.0A3

- If the Singlecast Follow-up uses Supervision encapsulation, the receiving node MUST return a Supervision Report with the MOS extension.

CC:009F.01.00.11.0A4

- If the Singlecast Follow-up does not use Supervision encapsulation, the receiving node MUST return a Nonce Report with the MOS field set to 1.

CC:009F.01.00.11.0A5

3.6.5.2.1 MPAN table

CC:009F.01.00.12.00B A node SHOULD maintain the information indicated in Table 11 for each multicast group. The actual implementation format is out of scope of this specification.

Table 11, MPAN table entry, example

Information	Description
Owner NodeID	The NodeID of the node distributing this MPAN (rx only)
Group ID	Unique ID chosen by the owner node
MPAN Inner state	MPAN inner state used for encryption and decryption
MPAN entry state	MPAN_FREE: This entry is currently not in use MPAN_USED: This entry is currently in use MPAN_MOS: This entry is out of sync Waiting to send Nonce Report in response to singlecast follow-up

CC:009F.01.00.11.0AC A supporting node MUST support at least 1 multicast session as a receiver (1 MPAN table entry).

3.6.5.2.2 Adding an S2 Multicast group member

A group member can be added to a S2 Multicast group by sending a Singlecast follow-up message to the new NodeID after a S2 Multicast message. The newly added NodeID will notify the sender that it is not synchronized for the given Group ID and it will be synchronized when receiving the next singlecast frame.

CC:009F.01.00.12.017 S2 Multicast will not work with sleeping nodes A sending node SHOULD NOT try to add sleeping nodes to a Multicast group.

3.6.5.2.3 Removing an S2 Multicast group member

A node can be removed from an S2 Multicast group by shuffling the MPAN state. At the next S2 Multicast frame, the newly removed node will set in MOS state and wait for re-synchronization in the next singlecast messages. When the newly removed node receives subsequent singlecast messages without MPAN extension, the newly removed node MUST forget about the Multicast group ID.

CC:009F.01.00.11.099

3.6.5.2.4 Handling a missing S2 Multicast frame

An S2 MC frame may be missing due to simple radio interference or a deliberate delay attack.

In either case, the MPAN needs to be re-synchronized. Further, the MPAN needs to be advanced immediately to close the time window where an attacker can replay the stolen S2 MC frame.

The transmission of a singlecast follow-up frame ensures re-synchronization and prevents a potential delay attack. Figure 17 outlines an example of the MPAN state changes when an S2 MC frame is missing.

In the case both multicast and singlecast follow-up frames are intercepted for a delay attack, the attacker has the possibility to replay the multicast frame at a later time. Using the Supervision Command Class in singlecast follow-up frames prevent this attack as the receiving node needs to report the application level status in a new singlecast frame.

3.6.5.3 Message encapsulation commands

This section presents the commands of the Security 2 Command Class used for message encapsulation.

3.6.5.3.1 Security 2 Nonce Get Command

This command is used to request a fresh Nonce.

CC:009F.01.01.11.001 The Security 2 Nonce Report Command MUST be returned in response to this command.

CC:009F.01.01.11.002 A node sending this command MUST accept a delay up to <Previous Round-trip-time to peer node> + 250 ms before receiving the Security 2 Nonce Report Command.

CC:009F.01.01.11.003 This command MUST NOT be issued via multicast addressing.
A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY_2							
Security Header = SECURITY_2_NONCE_GET							
Sequence Number							

Sequence Number (1 byte)

CC:009F.01.01.11.004 A sending node MUST specify a unique sequence number starting from a random value. Each message MUST carry an increment of the value carried in the previous outgoing message.

CC:009F.01.01.11.005 A receiving node MUST validate the Sequence Number and the actual sender's NodeID before accepting this command.

CC:009F.01.01.11.006 A receiving node MUST use this field for duplicate detection. Refer to section 3.6.5.4.

3.6.5.3.2 Security 2 Nonce Report Command

This command is used to advertise a fresh Nonce in preparation for secure communication. This
CC:009F.01.02.11.001 command MUST NOT be issued unless it is done in response to the S2 Nonce Get or the S2 Message Encapsulation Command.

CC:009F.01.02.11.002 A sending node MUST set at least one of the MOS and SOS flags to the value '1' in this command. The command MUST NOT be sent if both the MOS and SOS flags are cleared.

CC:009F.01.02.11.003 A receiving node MUST update the MPAN and/or SPAN of the node sending this command by adding a SPAN and/or MPAN extension to the next Security 2 Message Encapsulation Command.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY_2							
Command = SECURITY_2_NONCE_REPORT							
Sequence Number							
Reserved						MOS	SOS
Receiver's Entropy Input Byte 1 (Optional)							
…							
Receiver's Entropy Input Byte 16 (Optional)							

Sequence Number (1 byte)

CC:009F.01.02.11.004 A sending node MUST specify a unique sequence number starting from a random value. Each message MUST carry an increment of the value carried in the previous outgoing message.

CC:009F.01.02.11.005 A receiving node MUST use this field for duplicate detection. Refer to section 3.6.5.4.

Reserved

CC:009F.01.02.11.006 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

SOS – SPAN out of Sync (1 Bit)

CC:009F.01.02.11.007 When set by a sending node, the value '1' MUST indicate that the sending node does not have a SPAN established for the receiving node or was unable to decrypt the most recently received singlecast Security 2 Message Encapsulation Command from the destination of this command.

CC:009F.01.02.11.008 The value '0' MUST indicate that there was no problem decrypting the most recently received singlecast Security 2 Message Encapsulation Command.

CC:009F.01.02.11.009 If the SOS flag is set to '1', the REI field MUST be included in the command.
If the SOS flag is set to '0', the REI field MUST NOT be included in the command.

CC:009F.01.02.11.00A A receiving node MUST establish a new Singlecast Pre-Agreed Nonce (SPAN) and return a Security 2 Message Encapsulation Command with the SPAN extension, if this flag is set to '1' by a sending node.

MOS – MPAN Out of Sync (1 Bit)

CC:009F.01.02.11.00B When set by a sending node, the value '1' MUST indicate that the sending node does not have a MPAN state for Multicast group used in the most recently received singlecast follow-up Security 2 Message Encapsulation Command from the destination of this command.

CC:009F.01.02.11.00C The value '0' MUST indicate that there was no problem decrypting the most recently received multicast Security 2 Message Encapsulation Command that was followed by a singlecast follow-up.

CC:009F.01.02.11.00D A sending node MUST NOT advertise the MOS flag for a given (source NodeID, Group ID) tuple more than one time.

CC:009F.01.02.11.00E If this flag is set to '1' by a sending node, a receiving node MUST transfer the Multicast Pre-Agreed Nonce (MPAN) of the most recent singlecast follow-up transmission in a subsequent singlecast message if the sending node belongs to the multicast group. The MPAN MUST be transferred by sending a singlecast Security 2 Message Encapsulation Command with the MPAN extension.

Receiver's Entropy Input (REI) (16 Bytes)

This field is optional. When present, this field is used to carry the Receiver's Entropy Input in preparation for new S2 transmissions based on the SPAN. Refer to 3.6.5.1.

CC:009F.01.02.11.00F If the SOS flag is set to '1', the REI field MUST be included in the command.
If the SOS flag is set to '0', the REI field MUST NOT be included in the command.

3.6.5.3.3 Security 2 Message Encapsulation Command

CC:009F.01.03.11.001 The Security 2 Nonce Report Command MUST be returned in response to this command if the receiving node fails decrypting the command or if the message contains an MGRP extension with a Group ID that is unknown or out of sync (MOS).

CC:009F.01.03.13.001 This command MAY be issued via Z-Wave Multicast addressing.

CC:009F.01.03.11.002 A receiving node MUST NOT return a response if this command is received via Z-Wave Multicast addressing. The Z-Wave Multicast frame and the broadcast NodeID are both considered Z-Wave Multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY_2							
Security Header = SECURITY_2_MESSAGE_ENCAPSULATION							
Sequence Number							
Reserved						Encrypted Extension	Extension
Extension 1 Length							
More to follow	Critical	Extension 1 Type					
Extension 1 Byte 1							
…							
Extension 1 Byte K							
Extension 2 Length							
More to follow	Critical	Extension 2 Type					
Extension 2 Byte 1							
…							
Extension 2 Byte L							
Encrypted Extension 1 Length							
More to follow	Critical	Encrypted Extension 1 Type					
Encrypted Extension 1 Byte 1							
…							
Encrypted Extension 1 Byte M							
Encrypted Extension 2 Length							
More to follow	Critical	Encrypted Extension 2 Type					
Encrypted Extension 2 Byte 1							
…							
Encrypted Extension 2 Byte N							
CCM Ciphertext object Byte 1							
…							
CCM Ciphertext object byte P							

Sequence Number (1 byte)

CC:009F.01.03.11.003 A sending node MUST specify a unique sequence number starting from a random value. Each new message MUST carry an increment of the value carried in the previous singlecast command.

CC:009F.01.03.11.004 A receiving node MUST use this field for singlecast duplicate detection. Refer to section 3.6.5.4.

Reserved

CC:009F.01.03.11.005 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node. However, the value MUST be used as part of the AAD input (3.6.4.4.2) used for authentication of the frame by the sending node as well as the receiving node.

Extension (1 bit)

This field is used to indicate if one or more non-encrypted extensions are included.

CC:009F.01.03.11.006 The value '1' MUST indicate that non-encrypted extensions are included.
The value '0' MUST indicate that non-encrypted extensions are not included.

Refer to the Extension object field description.

Encrypted Extension (1 bit)

This field is used to indicate if one or more encrypted extensions are included.

CC:009F.01.03.11.007 The value '1' MUST indicate that encrypted extensions are included.
The value '0' MUST indicate that encrypted extensions are not included.

Refer to the Encrypted Extension object field description.

[Extension Object] (N instances)

7	6	5	4	3	2	1	0
Extension Length							
More to follow	Critical	Type					
Extension Byte 1							
…							
Extension Byte L							

Extension Length (1 byte)

This field specifies the length of this extension, in bytes, including the "Extension Length" field.

Type (6 bit)

This field defines the type of this extension. Valid extension types are listed in 3.6.5.3.3.1.

Critical (1 bit)

CC:009F.01.03.11.008 A receiving node MUST discard the entire command if this flag is set to '1' and the Type field advertises a value that the receiving node does not support.

CC:009F.01.03.11.009 If this flag is set to '0' and the Type field advertises a value that the receiving node does not support, the actual extension MUST be ignored.

CC:009F.01.03.12.001 A receiving node SHOULD continue processing of the encapsulation command after the discarded extension.

More to Follow (1 bit)

CC:009F.01.03.11.00A If the More to Follow flag is set to '1', another Extension Object MUST follow this Extension Object.

Extension (Variable length)

CC:009F.01.03.11.00B This field carries the actual extension. Refer to 3.6.5.3.3.1.The length of this field MUST comply with the length specified in the Extension Length field of this Extension Object.

[Encrypted Extension Object] (N instances)

CC:009F.01.03.11.00C The format of this object is identical to the unencrypted Extension Object. However, this object MUST be part of the encrypted payload.

CCM Ciphertext Object (P bytes)

This field contains an encrypted message. It comprises:

- (Optional) Complete Z-Wave command (comprising Command Class Identifier, Command Identifier and optional command payload)
- CCM control data
- CCM authentication tag.

The preceding Encrypted Extension fields are technically also a part of the CCM ciphertext object. But conceptually they are separate from the message and have been described as such.

3.6.5.3.3.1 Valid Extensions and Encrypted Extensions

The defined extension types are listed in Table 12.

Table 12, Security 2 Encapsulation Command::Extension Types

Extension Type Identifier	Format	Content protection
0x01	SPAN Extension	Not Encrypted
0x02	MPAN Extension	Encrypted
0x03	MGRP Extension	Not Encrypted
0x04	MOS Extension	Not Encrypted

CC:009F.01.03.11.00D All other values are reserved and MUST NOT be used by a sending node. Reserved values MUST be ignored by a receiving node.

3.6.5.3.3.1.1 SPAN Extension

The SPAN Extension is used by the sender to establish a SPAN by sending a Sender's Entropy Input to the Receiver. The combined Sender's and Receiver's Entropy Input may then be passed through the *NextNonce* function to generate the SPAN.

CC:009F.01.03.11.00E This extension MUST be sent unencrypted.

7	6	5	4	3	2	1	0
Length = 18							
More to follow	Critical = 1	Type = SPAN Extension = 1					
Sender's Entropy Input Byte 1							
…							
Sender's Entropy Input Byte 16							

Length (1 byte)

CC:009F.01.03.11.00F This field MUST be set to 18.

Type (6 bits)

CC:009F.01.03.11.010 This field MUST be set to the SPAN Extension type (1).

Critical (1 bit)

CC:009F.01.03.11.011 This flag MUST be set to '1'.

More to Follow (1 bit)

CC:009F.01.03.11.012 If this flag is set to '1', another unencrypted Extension Object MUST follow this unencrypted Extension Object.
If this flag is set to '0', this MUST be the last unencrypted Extension Object.

Sender's Entropy Input (16 bytes)

CC:009F.01.03.11.013 This field MUST carry the entropy input contribution used to instantiate a *NextNonce* Generator.

3.6.5.3.3.1.2 MPAN Extension

The MPAN Extension is used by the sender to establish or update an MPAN by sending the full 16 bytes MPAN state to the receiver to enable the reception of encrypted multicast transmissions.

CC:009F.01.03.11.014 This extension MUST be sent encrypted.

7	6	5	4	3	2	1	0
Length = 19							
More to follow	Critical = 1	Type = MPAN Extension = 2					
Group ID							
Inner MPAN state Byte 1							
…							
Inner MPAN state Byte 16							

Length (1 byte)

CC:009F.01.03.11.015 This field MUST be set to 19.

Type (6 bits)

CC:009F.01.03.11.016 This field MUST be set to the MPAN Extension type (2).

Critical (1 bit)

CC:009F.01.03.11.017 This flag MUST be set to ‘1’.

More to Follow (1 bit)

CC:009F.01.03.11.018 If this flag is set to ‘1’, another encrypted Extension Object MUST follow this encrypted Extension Object.
If this flag is set to ‘0’, this MUST be the last encrypted Extension Object.

Group ID (1 byte)

CC:009F.01.03.11.019 This field is used to identify MPAN instances. A sending node MUST set this value to the Group ID of the most recently transmitted multicast frame. The value MUST be in the range 0..255.

CC:009F.01.03.11.02D A sending node creating a new group SHOULD NOT use the value 0.

CC:009F.01.03.11.01A A receiving node MUST use this value in combination with the Owner NodeID for identifying the correct MPAN table entry when an MPAN table entry to use for frame decryption or MPAN updates.

CC:009F.01.03.11.01B If a receiving node already has an MPAN for the actual Group ID, that MPAN MUST be updated.
If this is a new Group ID, the new information MUST be stored. This may require the deletion of another
CC:009F.01.03.12.002 MPAN table entry. It is RECOMMENDED that the least recently used entry is deleted.

Inner MPAN state (16 bytes)

This field carries the value to be used in the encryption and decryption of the next S2 Multicast frame.

3.6.5.3.3.1.3 MGRP Extension

CC:009F.01.03.11.01C The Multicast Group (MGRP) Extension MUST be included in all S2 Multicast and S2 Singlecast follow-up frames.

CC:009F.01.03.11.01D A receiving node MUST use the Group ID to select the MPAN for decrypting this message.

CC:009F.01.03.11.01E When receiving this extension, the matching MPAN (if found) MUST be incremented by one after decryption.

CC:009F.01.03.11.02B If the Group ID is not found in the receiver MPAN table and the received frame is a singlecast (follow-up), the node MUST return a Nonce Report with the MOS flag set, or alternatively, return another Security 2 Message Encapsulation Command with the MOS extension included.

CC:009F.01.03.11.020 This extension MUST NOT be sent together with the MPAN extension.

CC:009F.01.03.11.021 This extension MUST be sent unencrypted.

7	6	5	4	3	2	1	0
Length = 3							
More to follow	Critical = 1	Type = MGRP Extension = 3					
Group ID							

Length (1 byte)

CC:009F.01.03.11.022 This field MUST be set to 3.

Type (6 bits)

CC:009F.01.03.11.023 This field MUST be set to the MGRP Extension type (3).

Critical (1 bit)

CC:009F.01.03.11.024 This flag MUST be set to ‘1’.

More to Follow (1 bit)

CC:009F.01.03.11.025 If this flag is set to ‘1’, another unencrypted Extension Object MUST follow this unencrypted Extension Object.
If this flag is set to ‘0’, this MUST be the last unencrypted Extension Object.

Group ID (1 byte)

This field is used by the sender to uniquely identify which MPAN was used to encrypt this multicast frame.

3.6.5.3.3.1.4 MOS Extension

The MOS extension is used to indicate that the sending node does not have a MPAN state for the Multicast group used in the most recently received singlecast follow-up Security 2 Message Encapsulation Command from the destination of this command

CC:009F.01.03.13.002 The receiver MAY choose to re-synchronize the sending node by returning a new Security 2 Message Encapsulation Command with a MPAN extension included.

CC:009F.01.03.11.026 This extension MUST be sent unencrypted.

7	6	5	4	3	2	1	0
Length = 2							
More to follow	Critical = 0	Type = MOS Extension = 4					

Length (1 byte)

CC:009F.01.03.11.027 This field MUST be set to 2.

Type (6 bits)

CC:009F.01.03.11.028 This field MUST be set to the MOS Extension type (4).

Critical (1 bit)

CC:009F.01.03.11.02C This flag MUST be set to ‘0’.

More to Follow (1 bit)

CC:009F.01.03.11.02A If this flag is set to ‘1’, another unencrypted Extension Object MUST follow this unencrypted Extension Object.
If this flag is set to ‘0’, this MUST be the last unencrypted Extension Object.

3.6.5.4 Duplicate Message Detection

CC:009F.01.00.11.0AD A sending node MUST set the Sequence Number to a random value on startup. The Sequence Number MUST be incremented for the transmission of each new unique singlecast (and singlecast follow-up) transmission.

CC:009F.01.00.11.02D A receiving node MUST use the Sequence Number field in the Nonce Get, Nonce Report and Message Encapsulation commands for duplicate detection.

CC:009F.01.00.11.02E Duplicates of recently received commands MUST be discarded.

CC:009F.01.00.12.00C The following algorithm SHOULD be used for duplicate detection of singlecast messages:

CC:009F.01.00.11.02F 1. If an incoming sequence number and Peer NodeID match an entry in the SPAN table, the frame MUST be discarded.

CC:009F.01.00.11.030 2. If it is a new sequence number, the received sequence number and Peer NodeID MUST be stored in the SPAN table.

3.6.6 Key Management

CC:009F.01.00.11.033 S2 communication relies on two separate derived keys that MUST be shared between any nodes communicating with each other; the combined Encryption and Authentication Key denoted `KeyCCM` and the CTR_DRBG Key denoted `PersonalizationString`. Both keys MUST be derived from one Network Key that is exchanged between the nodes using the *CKDF-NetworkKeyExpand* function detailed in 3.6.4.7.3.1.

CC:009F.01.00.11.034 In S2, not all nodes share the same network key. S2 separates devices into different Security Classes, so that if one Security Class is compromised, it does not affect other Security Classes. The joining node MUST advertise the supported Security Classes during the S2 bootstrapping process.

CC:009F.01.00.13.003 The including node MAY grant membership of one or more Security Classes.

CC:009F.01.00.11.035 The joining node MUST then request the granted keys, one at a time.

The Security Classes are listed in Table 13.

S2 Access Control and S2 Authenticated Security Classes are equivalent from a technical perspective, but do not share the same network key. This is simply to prevent compromising the Access Control key if the Authenticated key is compromised.

Client-side authentication is an alternative authentication mechanism intended for the authenticated security bootstrapping of devices which do not have a DSK. Refer to section 3.6.6.3.

Table 13, S2 Security Class Overview

Value	Class Name	Example devices	Controller authentication (refer to 3.6.6.2)		Client-side authentication (refer to 3.6.6.3)	
			Display DSK	***Input DSK***	***Display DSK***	***Input DSK***
2	S2 Access Control	Door locks, Garage doors	14 bytes	5 decimal digits (2 bytes)	12 bytes (Optional)	10 decimal digits (4 bytes)
			None (QR code)	QR code (16 bytes)	None (QR code)	QR code (16 bytes)
1	S2 Authenticated	Lighting and Sensors (Managed systems and security systems)	14 bytes	5 decimal digits (2 bytes)	12 bytes (Optional)	10 decimal digits (4 bytes)
			None (QR code)	16 bytes (QR code)	None (QR code)	16 bytes (QR code)
0	S2 Unauthenticated	Lighting and Sensors (Unmanaged systems)	0 to 16 bytes	0	0	0
7	S0	Legacy door locks	N/A	N/A	N/A	N/A

CC:009F.01.00.11.036 All other values are reserved and MUST NOT be used by a sending node. Reserved values MUST be ignored by a receiving node.

The manufacturer decides which classes are relevant to advertise for the intended use of the product. For instance:

- A light dimmer device may advertise the S2 Unauthenticated Class and the S2 Authenticated Class as the intended classes. If a constrained key fob without display is used to create a small system, the wall controller may grant only the S2 Unauthenticated Class to the light dimmer, which then requests only the S2 Unauthenticated Class key.
- A light bulb may advertise the S2 Unauthenticated Class as its only intended class because it does not provide a DSK. Depending on the configured security level, an authentication capable gateway may accept including S2 Unauthenticated Class devices or it may reject including S2 Unauthenticated Class-only devices.
- A door lock device may advertise the S2 Access Control Class as its only intended class because it requires the highest protection level. A constrained key fob, which can grant only the S2 Unauthenticated Class key, rejects including the door lock device and returns an error indication to the user because it cannot authenticate the door lock.

3.6.6.1 Key Exchange

CC:009F.01.00.11.037 The S2 key exchange MUST comply with Table 14.

Table 14, Key exchange and key verification

Key to be exchanged	Key to use for the Key Exchange	Key to use for the Key Verification
S2.2: Access Control	ECDH Temporary Key	S2.2
S2.1: Authenticated	ECDH Temporary Key	S2.1
S2.0: Unauthenticated	ECDH Temporary Key	S2.0
S0	ECDH Temporary Key	S0

CC:009F.01.00.11.039 A number of Security Class keys may be granted to the joining node. A temporary key and SPAN MUST be used for the exchange of the Security Class keys.

CC:009F.01.00.11.03A The temporary SPAN MUST be initialized with the Shared Nonce that is established after the S2 temporary key is established.

CC:009F.01.00.11.03B The SPAN value MUST be updated after each Security Class key exchange.

CC:009F.01.00.11.03C Verification of an assigned key (including S0) MUST always use the newly exchanged key to encrypt the S2 verification message.

CC:009F.01.00.11.040 If a joining node is unsuccessful requesting all the Security Class keys that it was granted, the node MUST abort the S2 bootstrapping entirely.

CC:009F.01.00.11.0B4 The S0 key can be exchanged with the Security 0 Command Class or with the Security 2 Command Class. The S0 message encapsulation MUST be done with the Security 0 Command Class encapsulation Command after S2 bootstrapped is completed.

3.6.6.2 ECDH key pairs, Device Specific Key and User Verification

A Device Specific Key (DSK) is used to protect against man-in-the-middle attacks (MITM) where a malicious attacker tries to intercept and manipulate the key exchange.

CC:009F.01.00.11.0B6 A node able to perform S2 bootstrapping MUST have several separate ECDH key pairs:

- Authenticated Learn Mode ECDH key pair:
  This key pair is used for joining a network and when the S2 Bootstrapping requires Authentication. This key pair MUST be static.
- Unauthenticated Learn Mode ECDH key pair:
  This key pair is used for joining a network and when the S2 Bootstrapping does not require Authentication. This key pair MAY be dynamic.
- Add Mode ECDH key pair:
  This keypair is used for adding nodes into a network. (controller side). This key pair MUST be dynamic.

CC:009F.01.00.11.0B7 A unique Authenticated Learn Mode ECDH key pair MUST be assigned to each individual node, if they request an Authenticated Security Class.

CC:009F.01.00.13.018 A node MAY create a new Unauthenticated Learn Mode ECDH key pair for every S2 bootstrapping attempt.

CC:009F.01.00.11.0B8 A node MUST keep the same Authenticated Learn Mode ECDH key pair for the lifetime of the node.

CC:009F.01.00.11.0B9 The DSK is defined as the first 16 bytes of the Authenticated ECDH Public Key of a node. An S2 node MUST respect the DSK requirements listed in [8].

The DSK may be used for out-of-band (OOB) authentication in two ways.

- The including controller uses QR code scanning to read the entire DSK off the joining device and match it with the obfuscated public key received via RF from the joining device.
- Else the including controller asks the user to visually validate that the rest of the DSK matches with the Public Key received via RF. The including controller additionally asks the user to enter the PIN code (the 5 first digits of the DSK string) in order to substitute the obfuscated bytes of the joining node's Public Key.

CC:009F.01.00.12.011 An including controller with support for the S2 Authenticated Class or the S2 Access Control Class SHOULD provide a QR code scanning capability for user friendly inclusion.

CC:009F.01.00.11.04E If scanning capability is not available, the including controller MUST provide an interface that allows the user to enter a 5-digit PIN code and perform visual DSK string validation.

CC:009F.01.00.11.04F The requested PIN code MUST be the first 5 decimal digits of the DSK string.

3.6.6.3 Client-Side Authentication

CC:009F.01.00.11.06E When upgrading existing devices to support Security 2 through an over-the-air (OTA) firmware update, there is no DSK printed on the node and the upgraded node MUST therefore generate its public key and DSK internally with the updated firmware.

CC:009F.01.00.13.00C
CC:009F.01.00.11.06F A device MAY request the use of Client-Side authentication (CSA) if it does not possess a DSK label. If the joining node requests CSA, the including controller MUST ask the user if this should be permitted, and if permitted, the including controller MUST display its own DSK.

CC:009F.01.00.11.070 As stated in Table 13, the first 10 digits MUST be input on the Joining Node, meaning it MUST have a method of input, like a keypad.

CC:009F.01.00.11.071 Compared to controller-side authentication where the entire DSK MUST also be visually verified, CSA does come with a higher risk of having the bootstrapping attempt manipulated by an attacker. The attacker, however, has to guess 4 random bytes in one attempt.

3.6.6.4 Initial Key Exchange

CC:009F.01.00.11.0A6 Add mode MUST be initiated physically. Physical activation includes button press, applying power, remote user activation, etc.

CC:009F.01.00.11.052 Initial Key Exchange MUST be carried out using the ECDH temporary key with the including controller.

CC:009F.01.00.11.053 The Including Node A MUST create a new Add Mode ECDH Key Pair for each new bootstrapping process (regardless of the outcome of each bootstrapping attempt), if it supports the S2 Access Control and/or S2 Authentication Security Classes.

CC:009F.01.00.11.055 The Key Exchange process MUST follow the frame flow illustrated in Figure 19.

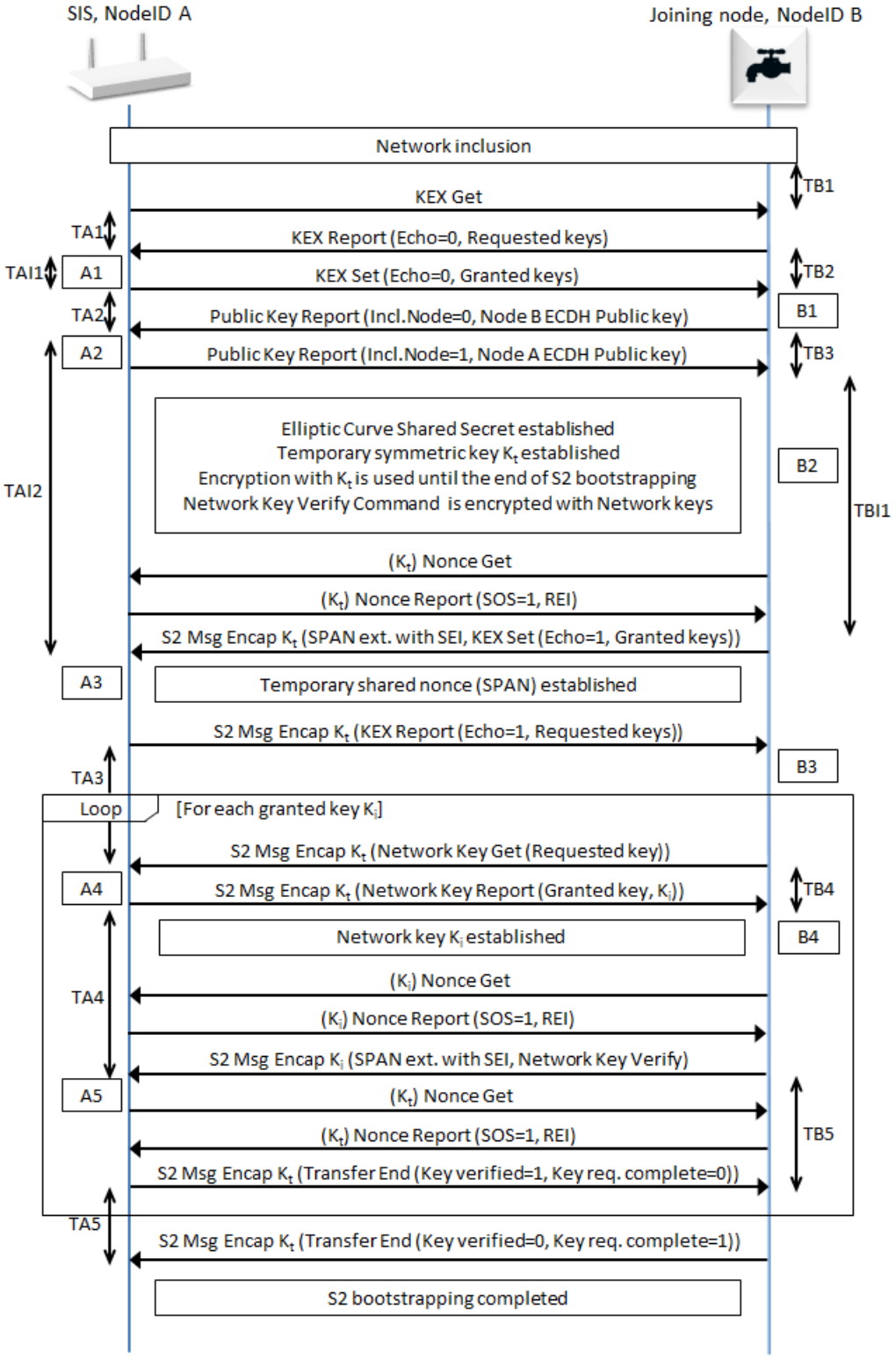

Figure 19, S2 bootstrapping frame flow

CC:009F.01.00.11.056 The key exchange MUST comply with the following steps:

1. **Network inclusion completed:**
   CC:009F.01.00.11.057 Immediately following a successful network inclusion or after receiving an Inclusion Controller Initiate Command (refer to [15]), the Security 2 enabled controller A MUST start the S2 bootstrapping
2. **A->B : KEX Get :**
   Including Node A, requests KEX Report from Joining Node B.
3. **B->A : KEX Report :**
   Sent as response to the KEX Get command, this command contains:
   a. Scheme Report – Schemes supported by Node B
   b. Curve Report – Elliptic Curves supported by Node B
   c. Requested Key List – List of Keys (such as S2 Class keys and Security 0 Network Key) which is requested by Node B
   d. Request for Client-Side authentication
4. **A1:**
   CC:009F.01.00.11.058 Node A MUST verify the KEX Report and, if required, cancel the S2 bootstrapping as described in Section 3.6.6.4.1.
   CC:009F.01.00.13.007 CC:009F.01.00.11.059 a. **Optional:** Node A MAY present a dialog allowing the installer to select which specific keys will be granted to Node B. If presented, the installer MUST either confirm a list of granted keys or cancel the security bootstrapping.
   CC:009F.01.00.11.0B5 b. If Client-Side authentication is requested and Node A supports inclusion using CSA, Node A MUST present a dialog asking if Client-Side authentication should be allowed.
      CC:009F.01.00.11.05B i. If Client-Side authentication is used, Node A MUST subsequently present a dialog with its own DSK.
      CC:009F.01.00.13.016 CC:009F.01.00.11.093 ii. Node A MAY reject Client-Side authentication. In this case, Node A MUST either abort the S2 bootstrapping with a KEX_FAIL_CANCEL or only grant a subset of keys that does not require CSA, e.g. Security 0 and Unauthenticated.
5. **A->B** : **KEX Set :**
   CC:009F.01.00.13.008 The KEX Set Command contains parameters selected by Node A. The list of class keys MAY be reduced to a subset of the list that was requested in the previous KEX Report from Node B.
   a. Scheme Set – The selected Scheme by Node A for bootstrapping
   b. Curve Set – The selected Curve by Node A for bootstrapping
   c. Key List Set – The list of granted keys to Node B by Node A
   d. Client-Side authentication – Indicating whether Client-side authentication will be used
6. **B1:**
   CC:009F.01.00.11.05D Node B MUST verify the KEX Set command and, if required, cancel security bootstrapping as described in Section 3.6.6.4.1
   CC:009F.01.00.13.009 CC:009F.01.00.11.05E CC:009F.01.00.13.00A a. **Optional:** Node B MAY present the Node A's DSK for verification or input. If presented, installer MUST either confirm or cancel security bootstrapping. This step MAY also be carried out in step B2 instead is CSA is used.
   CC:009F.01.00.11.092 b. Node B MUST accept what it has been granted, even if it is only a subset of what was requested.
   CC:009F.01.00.11.091 c. If Node A wrongfully grants CSA without being requested, Node B MUST cancel security bootstrapping.
7. **B->A : Public Key B :**
   CC:009F.01.00.11.0BA Public Key B is the ECDH Public Key of Node B and is used for the ECDH Key Exchange.

a. If Authentication is required (KEX Set granted an Authenticated Security Class): Node B MUST use its Authenticated ECDH Public key and the DSK bytes 1..2 MUST be obfuscated by zeros.
b. If Authenticated is not required (KEX Set granted no authenticated Security Class): Node B MUST use its Unauthenticated ECDH Public Key and transmit the full key non-obfuscated.

8. **A2:**

CC:009F.01.00.11.05F

If authentication is required, Node A MUST request that the user enters the PIN code or scans the QR code from Node B in order to verify the DSK (refer to 3.6.6.2 and 3.6.6.4.1).

a. If Node A was input a PIN code, it MUST substitute the bytes 1 and 2 of the Node B public key with the 2 bytes received in the PIN code. The user MUST be prompted a dialog to visually validate the bytes 3..16 of Node B's DSK.
b. If Node A has received the 16 bytes DSK of Node B via QR scanning, it MUST substitute the first 16 bytes of Node B's Public Key with the 16 bytes received via QR code.

CC:009F.01.00.12.012

c. Node A SHOULD continue verifying the DSK input until A3 to allow ECDH calculations to take place while the user is verifying the DSK.

9. **A->B : Public Key A :**

Public Key A is the Elliptic Curve Public Key of Node A and will be used for the temporary ECDH Key Exchange.

CC:009F.01.00.11.060

a. **Mandatory:** If Client-Side authentication is used, the DSK bytes 1..4 MUST be obfuscated by zeros.

10. **B2:**

CC:009F.01.00.11.061

a. **Mandatory:** If Client-Side authentication is used, the DSK MUST be input on Node B.
    i. If Node B was input a PIN code, it MUST substitute the 4 first bytes of Node A's Public Key with the 4 bytes received in the PIN code. The user MAY be prompted a dialog to visually validate the bytes 5..16 of Node A's DSK.
    ii. If Node B has received the 16 bytes DSK of Node A via QR scanning, it MUST substitute the first 16 bytes of Node A's Public Key with the 16 bytes received via QR code

11. **Elliptic Curve Shared Secret Established:**

If B2 is passed, Node A and Node B have performed an ECDH Key Exchange, resulting in an Elliptic Curve Shared Secret.

12. **Temporary Symmetric Key Established:**

Both Node A and Node B derive a Temporary Symmetric Key from the ECDH Shared Secret based on *CKDF-TempExpand* (refer to 3.6.4.7.2).

13. **B->A : Nonce Get :**

Node B requests a Nonce from Node A that will allow Node B to send messages securely using the Temporary Symmetric Key.

14. **A->B : Nonce Report :**

A's Nonce

CC:009F.01.00.11.062

15. *From this point all frames sent between Node A and Node B MUST be encrypted using the ECDH Temporary Symmetric Key (With the exception of Nonce Get / Report for each Security Class which MUST NOT be encrypted and the Network Key Verify Command, which MUST be encrypted with the most recently exchanged key. Refer to Section 3.6.6.1).*

16. **B->A : KEX Set (echo) :**

The KEX Set command received from Node A in step 5 is confirmed via the temporary secure channel.

CC:009F.01.00.11.097

a. The frame MUST be retransmitted by node B every 10 seconds for 240 seconds in total until either event is detected:

i. Step 17, KEX Report(Echo) is received
ii. KEX Fail is received

b. If neither events are detected after 240 seconds, Node B times out and aborts S2 bootstrapping silently

17. **A3:**
CC:009F.01.00.11.063 Node A MUST abort S2 bootstrapping if the KEX Set(Echo) received in step 16is not identical to KEX Set previously sent by Node A in step 5. Refer to Section 3.6.6.4.1.
18. **A->B : KEX Report (echo) :**
The KEX Report Command received from Node B in step 3 is confirmed via the temporary secure channel.
19. **B3:**
CC:009F.01.00.11.064 Node B MUST abort S2 bootstrapping if the KEX Report(Echo) received in step 18 is not identical to KEX Report previously sent by Node B in step 3. Refer to Section 3.6.6.4.1.
CC:009F.01.00.11.098 If a Node B node has been granted zero keys, Node B MUST go to step 30 and indicate to Node A that it is terminating the bootstrapping process by returning an S2 Transfer End Command. Node A and Node B will consider Node B to be included non-securely at the end of the bootstrapping.

CC:009F.01.00.11.065 *Authentication has been completed, and network key exchange begins. Steps* 20 *through 29 MUST be repeated for each network key Node A has granted. Key Exchange MUST follow the order described in Section 3.6.6.1.*

20. **B->A : Security 2 Network Key Get:**
Node B requests a specific Key from Node A.
21. **A4:**
CC:009F.01.00.11.0B2 Node A MUST cancel the S2 bootstrapping if the requested key is not in the Key List granted by Node A.
22. **A->B :Security 2 Network Key Report:**
This command returns the requested key.
23. **B4:**
CC:009F.01.00.11.094 Node B MUST cancel security bootstrapping if the received key was not requested. Refer to 3.6.6.4.1.
CC:009F.01.00.11.067 After receiving the key, Node B MUST perform key derivation as described in 3.6.4.7.3.1
24. **Security 2 Network Key Established:**
Node A and Node B are now in possession of a shared network key.
25. **B->A : Nonce Get :**
Node B requests a Nonce from Node A that will allow Node B to send messages securely using the recently exchanged Network Key.
26. **A->B : Nonce Report :**
A's Nonce
27. **B->A : Security 2 Network Key Verify :**
CC:009F.01.00.11.095 This command MUST be sent encrypted by Node B with the newly received Network Key and the recently established SPAN for this key
28. **A5:**
CC:009F.01.00.11.06A Node A MUST verify that it can successfully decrypt the Key Verify command using the newly exchanged key.
29. **A->B : Security 2 Transfer End:**
CC:009F.01.00.11.06B If Node A is able to decrypt and verify the Key Verify command, it MUST respond with Security 2 Transfer End with the field "Key verified" set to '1'.

a. If there are more granted keys to request, Node B continues to step 20 and requests the next granted key

b. If it was the last granted key, Node B continues to step 30.

All Keys have been requested.

30. **B->A** : **Security 2 Transfer End :**

CC:009F.01.00.11.06C When Node B has no more keys to request it MUST finish the secure setup by sending a Security 2 Transfer End command with the field “Key Request Complete” set to ‘1’.

a. If this frame is received before all keys have been requested, the controller MUST consider the S2 Bootstrapping process failed.

CC:009F.01.00.11.06D All timeouts described in Section 3.6.6.4.2 MUST be implemented. If a node times out, it MUST silently abort the S2 bootstrapping.

3.6.6.4.1 Security 2 bootstrapping Interrupt Points

CC:009F.01.00.11.077 The including node and the joining node MUST interrupt the Security 2 bootstrapping process if any of the events outlined in Table 15 occur.

Table 15, Security 2 bootstrapping Interrupt Points

Including Node Interrupts, A1-5			
Name	**Interrupt Reason**	**KEX Fail Command Encryption**	**KEX Fail Type (see description in Table 20)**
A1	Key List is invalid or unsupported KEX Scheme is invalid or unsupported Curves are invalid or unsupported. User rejects the requested keys	None	KEX_FAIL_KEX_KEY KEX_FAIL_KEX_SCHEME KEX_FAIL_KEX_CURVES KEX_FAIL_CANCEL
A2	User cancelled the bootstrapping.	None	KEX_FAIL_CANCEL
A3	KEX Set(Echo) is not identical to the previous KEX Set	Temp. Key	KEX_FAIL_AUTH KEX_FAIL_DECRYPT
A4	The requested Key was not granted in the original KEX Set	Temp. Key	KEX_FAIL_KEY_GET
A5	The Key Verify Command cannot be decrypted using most the recent key	Temp. Key	KEX_FAIL_KEY_VERIFY
Joining Node Interrupts, B1-4			
Name	**Interrupt Reason**	**KEXFail Command Encryption**	**KEX Fail Type (see description in Table 20)**
B1	Key List is invalid or unsupported KEX Scheme is invalid or unsupported Curves are invalid or unsupported Node A wrongfully grants CSA	None	KEX_FAIL_KEX_KEY KEX_FAIL_KEX_SCHEME KEX_FAIL_KEX_CURVES KEX_FAIL_KEX_KEY
B2	If using CSA, user cancelled the bootstrapping	-	KEX_FAIL_CANCEL
B3	KEX Report(Echo) is not identical to the previous KEX Report	Temp. Key	KEX_FAIL_AUTH KEX_FAIL_DECRYPT
B4	The assigned key was never requested.	Temp. Key	KEX_FAIL_KEY_REPORT

CC:009F.01.00.12.018 An aborted Security 2 bootstrapping process due to one of the interrupt points in Table 15 SHOULD be followed by a Security 2 KEX Fail Command sent to the other party of the S2 bootstrapping.

CC:009F.01.00.11.0A8 If sent, encryption MUST be used according to Table 15.

CC:009F.01.00.12.019 If a command is received using a Security level (non-secure, encrypted with temporary or network key) whereas the command had to be sent encrypted using another Security level, the receiving node SHOULD return a Security 2 KEX Fail Command with the KEX_FAIL_AUTH Fail Type encrypted using the received security level.

CC:009F.01.00.11.07C In any case, a node MUST NOT return any error indication if no S2 bootstrapping process is currently ongoing.

3.6.6.4.2 Security 2 bootstrapping Timeouts

CC:009F.01.00.11.07D The including Node or the joining node MUST apply timeouts according to Table 16.

Table 16, Security 2 bootstrapping Timeouts

Including Node Timers, TA1-7		
Name	**Interrupt Reason**	**Timeout (Seconds)**
CC:009F.01.00.11.07E TA1	KEX Report MUST be received after sending KEX Get	10
CC:009F.01.00.11.07F TA2	Public Key Report MUST be received after sending KEX Set	10
CC:009F.01.00.11.082 TA3	S2 Network Key Get MUST be received after sending KEX Report(Echo)	10
CC:009F.01.00.11.083 TA4	S2 Network Key Verify MUST be received after sending S2 Network Key Report	10
CC:009F.01.00.11.084 TA5	S2 Transfer End OR S2 Network Key Get MUST be received after sending S2 Transfer End	10
CC:009F.01.00.13.012 TAI1	User MAY change Key List for Advanced Joining mode	240
CC:009F.01.00.11.085 TAI2	User MUST have verified / input the DSK	240
Joining Node Timers, TB1-7		
Name	**Description**	**Timeout (Seconds)**
CC:009F.01.00.11.0B3 TB1	KEX Get MUST be received after non-secure inclusion	10..30 for nodes supporting S0 30 for nodes supporting S2 only
CC:009F.01.00.11.087 TB2	KEX Set MUST be received after sending KEX Report	240
CC:009F.01.00.11.088 TB3	Public Key Report MUST be received after sending Public Key Report	10
CC:009F.01.00.11.08B TB4	S2 Network Key Report MUST be received after sending S2 Network Key Get	10
CC:009F.01.00.11.08C TB5	S2 Transfer End MUST be received after sending S2 Network Key Verify	10
CC:009F.01.00.11.0AE TBI1	If Client-Side Authentication is used, the user MUST have verified / input the DSK	240

CC:009F.01.00.11.0AF 10 seconds timeouts MUST be observed with a 2 seconds tolerance. In this case, a node MUST time out between 8 and 12 seconds.

CC:009F.01.00.11.0B0 30 seconds timeouts MUST be observed with a 5 seconds tolerance. In this case, a node MUST time out between 25 and 35 seconds.

CC:009F.01.00.11.0B1 240 seconds timeouts MUST be observed with a 10 seconds tolerance. In this case, a node MUST time out between 230 and 250 seconds.

3.6.7 Security 2 Key Exchange commands

3.6.7.1 Security 2 KEX Get Command

This command is used by an including node to query the joining node for supported KEX Schemes and ECDH profiles as well as which network keys the joining node intends to request.

CC:009F.01.04.11.001 This command MUST be ignored if Learn Mode is disabled.

CC:009F.01.04.11.002 The KEX Report Command MUST be returned in response to this command if Learn Mode is enabled.

CC:009F.01.04.11.003 This command MUST NOT be issued via multicast addressing.

CC:009F.01.04.11.004 A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY_2							
Command = KEX_GET							

3.6.7.2 Security 2 KEX Report Command

This command is used for two purposes during the key exchange:

1) This command is used by a joining node to advertise the network keys which it intends to request from the including node. The including node subsequently grants keys which may be exchanged once a temporary secure channel has been established.
2) After establishment of the temporary secure channel, the including node uses this command to confirm the set of keys that the joining node intends to request.

CC:009F.01.05.11.003 A receiving node MUST ignore this command if Add Node mode is not enabled.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY_2							
Command = KEX_REPORT							
Reserved						Request CSA	Echo
Supported KEX Schemes							
Supported ECDH Profiles							
Requested Keys							

Reserved

CC:009F.01.05.11.004 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Echo (1 bit)

CC:009F.01.05.11.005 If this flag is set to '1', the fields of this command MUST be a copy of the KEX Report received prior to the establishment of the temporary secure channel. All fields described in this section MUST be included in the echo. For the purpose of echoing only, reserved set bits and the reserved field MUST

NOT be ignored or set to zero, but MUST be echoed back as received. If future versions of this Command Class append fields to the KEX Report, those fields MUST be omitted from the echo.

CC:009F.01.05.11.006 If this flag is set to '1', the command MUST be sent securely via the temporary secure channel, i.e. encrypted using the `TempKeyCCM` and `TempPersonalizationString`.

CC:009F.01.05.11.007 The including node MUST NOT set this flag to '0' when sending this command and MUST ignore this command if this flag is set to '1' when received.

CC:009F.01.05.11.008 The including node MUST return a KEX Set Command in response to this command if the "Echo" flag is set to '0' and is performing S2 Bootstrapping.

CC:009F.01.05.11.009 The joining node MUST NOT set this flag to '1' when sending this command and MUST ignore this command if this flag is set to '0' when received.

Request CSA (1 bit)

If this flag is set, the joining node is requesting the use of Client-side Authentication (CSA).

CC:009F.01.05.11.019 This flag MUST be set to 0 if none of the S2 Authenticated and S2 Access Control Security Classes are requested

CC:009F.01.05.11.01A This flag MUST be set to 0 if the sending node has a DSK label printed on itself.

CC:009F.01.05.11.01B This flag MUST be set to 1 by a node only if it has been OTA firmware upgraded to support S2 and requests S2 Authenticated or S2 Access Control Security Class.

Supported KEX Schemes (1 byte)

This field is used to advertise the KEX Schemes supported by the node.

CC:009F.01.05.11.00B The field MUST be treated as a bitmask and MUST comply with the format indicated in Table 17:

Table 17, Supported KEX Schemes

Bit	KEX Scheme	Description
0, 2..7	Reserved	Reserved
1	KEX Scheme 1	Indicates if Scheme 1 is supported (Security 2 Class Keys 0..2)

All other bits are reserved and MUST be set to zero by a sending node.

CC:009F.01.05.11.00E If the KEX scheme is supported the corresponding bit MUST be set to '1'.
If the KEX scheme is not supported the corresponding bit MUST be set to '0'.

CC:009F.01.05.11.00F A node supported the Security 2 Command Class MUST support KEX Scheme 1.

Supported ECDH Profiles (1 byte)

This field is used to advertise the supported ECDH Profiles by the joining node

CC:009F.01.05.11.010 The field MUST be treated as a bitmask and MUST comply with the format indicated in Table 18:

Table 18, Supported ECDH Profiles

Bit	ECDH Profile	Description	Public Key length
0	Curve25519	Indicates support for Curve25519 [27]	32 Bytes

All other bits are reserved and MUST be set to zero by a sending node.

CC:009F.01.05.11.013 If the ECDH Profile is supported the corresponding bit MUST be set to '1'
If the ECDH Profile is not supported the corresponding bit MUST be set to '0'

CC:009F.01.05.11.014 Curve25519 MUST be supported by a node supporting the Security 2 Command Class.

Requested Keys (1 byte)

This field is used by a joining node to advertise the keys that the manufacturer finds most appropriate for the actual type of product.

CC:009F.01.05.13.001 The joining node MAY request a subset of the keys defined in Table 19.

CC:009F.01.05.11.015 The field MUST be treated as a bitmask and MUST comply with the format indicated in Table 19:

Table 19, Requested Keys

Security Level (1 highest - 4 lowest)	Bit	KEX Scheme	Key	Indicates support for
1	2	KEX Scheme 1	S2.2	S2 Access Control Class
2	1	KEX Scheme 1	S2.1	S2 Authenticated Class
3	0	KEX Scheme 1	S2.0	S2 Unauthenticated Class
4	7	KEX Scheme 1	S0	S0 Secure legacy devices

All other bits are reserved and MUST be set to zero by a sending node. Reserved bits MUST be ignored by a receiving node.

CC:009F.01.05.11.017 If the Key is requested the corresponding bit MUST be set to '1'.
If the Key is not requested the corresponding bit MUST be set to '0'.

CC:009F.01.05.11.018 A node supporting the Security 2 Command Class MUST support at least one Key. A node may support multiple keys.

3.6.7.3 Security 2 KEX Set Command

This command is used for two purposes:

1) During initial key exchange this command is used by an including node to grant network keys to a joining node. The joining node subsequently requests the granted keys once a temporary secure channel has been established.
CC:009F.01.06.11.001 The including node MUST send the command non-securely.

2) After establishment of the temporary secure channel, the joining node issues this command to the including node to securely state its intention to request the keys that were granted previously.
CC:009F.01.06.11.002 The joining node MUST send the command securely via the temporary secure channel, i.e. encapsulated using the `TempKeyCCM` and `TempPersonalizationString`.

CC:009F.01.06.11.003 This command MUST be ignored if Learn mode and Add Node mode are both disabled.
CC:009F.01.06.11.004 The including node MUST return the Security 2 KEX Report Command in response to this command unless it is to be ignored.

CC:009F.01.06.11.005 This command MUST NOT be issued via multicast addressing.

CC:009F.01.06.11.006 A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY_2							
Command = KEX_SET							
Reserved						Request CSA	Echo
Selected KEX Scheme							
Selected ECDH Profile							
Granted Keys							

Reserved

CC:009F.01.06.11.007 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Echo (1 bit)

CC:009F.01.06.11.008 If this flag is set to ‘1’, the fields of this command MUST be an identical copy of the KEX Set Command received prior to the establishment of the temporary secure channel.

CC:009F.01.06.11.009 The joining node MUST set this flag to ‘1’.

CC:009F.01.06.11.00A The including node MUST abort the S2 bootstrapping if it receives a KEX Set Command with this flag set to ‘0’.

CC:009F.01.06.11.00B If the “Echo” flag is set to ‘1’ and the Add Node mode is enabled, the including node MUST return a KEX Report Command with the “Echo” flat set to ‘1’ in response to this command.

CC:009F.01.06.11.00D The including node MUST set this flag to ‘0’.

CC:009F.01.06.11.00E The joining node MUST abort the S2 bootstrapping if it receives a KEX Set Command with this flag set to ‘1’.

Request CSA (1 bit)

If this flag is set to ‘1’, the including node is permitting the use of Client-side Authentication (CSA).

Selected KEX Scheme (1 byte)

CC:009F.01.06.11.010 This field is used to specify the Scheme that the including node is granting to the joining node. Exactly one bit MUST be set to ‘1’. Reserved bits MUST be examined for the purpose of verifying that exactly one bit is set. Several bits set MUST trigger an error as indicated in 3.6.6.4.1.

For field format, refer to Table 17.

Selected ECDH Profile (1 byte)

CC:009F.01.06.11.011 This field specifies the ECDH Profile selected by the including node for ECDH Key Exchange. The field MUST carry exactly one of the bits listed in Table 18. Reserved bits MUST be examined for the purpose of verifying that exactly one bit is set.

For field format, refer to Table 18.

Granted Keys (1 byte)

CC:009F.01.06.11.012 If the Echo field is set to '0', this field MUST specify the keys which the including node is granting to the joining node.

CC:009F.01.06.11.013 The value of this field MUST be the same set or a subset of the Requested Keys field advertised in the KEX Report by the joining node during initial S2 bootstrapping.

CC:009F.01.06.11.014 If the Echo field is set to '1', this field MUST advertise the keys which the including node granted to the joining node in the previous KEX Set Command sent by the including node to the joining node.

For field format, refer to Table 19.

CC:009F.01.06.11.015 A joining node MUST NOT issue Network Key Get requests for keys that are not specified in this field.

CC:009F.01.06.11.016 An including node MUST return a Security 2 KEX Fail Command and abort S2 bootstrapping if it subsequently receives a Network Key Get request for keys that are not specified in this field.

3.6.7.4 Security 2 KEX Fail Command

This command is used to advertise an error condition to the other party of am S2 bootstrapping process.

The interrupt points that may trigger the transmission of this command are defined in section 3.6.6.3.

CC:009F.01.07.11.001 This command MUST be ignored if Learn mode and Add Node mode are both disabled.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY_2							
Command = KEX_FAIL							
KEX Fail Type							

KEX Fail Type (1 byte)

CC:009F.01.07.11.002 This field MUST advertise one of the types defined in Table 20

Table 20, Security 2 KEX Fail::KEX Fail Type

Value	KEX Fail Type Identifier	Description
0x01	KEX_FAIL_KEX_KEY	Key failure indicating that no match exists between requested/granted keys in the network.
0x02	KEX_FAIL_KEX_SCHEME	Scheme failure indicating that no scheme is supported by controller or joining node specified an invalid scheme.
0x03	KEX_FAIL_KEX_CURVES	Curve failure indicating that no curve is supported by controller or joining node specified an invalid curve.
0x05	KEX_FAIL_DECRYPT	Node failed to decrypt received frame.
0x06	KEX_FAIL_CANCEL	User has cancelled the S2 bootstrapping.
0x07	KEX_FAIL_AUTH	The Echo KEX Set/Report frame did not match the earlier exchanged frame. A command is received using the wrong Security level.
0x08	KEX_FAIL_KEY_GET	The joining node has requested a key, which was not granted by the including node at an earlier stage.
0x09	KEX_FAIL_KEY_VERIFY	Including node failed to decrypt and hence verify the received frame encrypted with exchanged key.
0x0A	KEX_FAIL_KEY_REPORT	The including node has transmitted a frame containing a different key than what is currently being exchanged.

3.6.7.5 Security 2 Public Key Report Command

This command is used by both the including and the joining node to establish the Elliptic Curve Shared Secret. This is needed to establish the temporary secure channel that enables transfer of all other keys.

CC:009F.01.08.11.001 This command MUST be ignored if Learn mode and Add Node mode are both disabled.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY_2							
Command = PUBLIC_KEY_REPORT							
Reserved							Including Node
ECDH Public Key 1 (MSB)							
…							
ECDH Public Key N (LSB)							

Reserved

CC:009F.01.08.11.002 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Including Node (1 bit)

The Including Node flag advertises if the sending node is the including node or the joining node.

CC:009F.01.08.11.003 When sent by the including node this flag MUST be set to '1'.
CC:009F.01.08.11.004 The joining node MUST abort S2 bootstrapping if this flag is set to '0' in a received command.

CC:009F.01.08.11.005 When sent by the joining node this flag MUST be set to '0'.
CC:009F.01.08.11.006 The including node MUST abort S2 bootstrapping if this flag is set to '1' in a received command.

ECDH Public Key (N bytes)

Device Specific ECDH Public Key of the sending node.

CC:009F.01.08.11.007 The public key MUST be unique for each node. The length of this field is determined by the chosen ECDH profile. Refer to Table 18.

CC:009F.01.08.11.008 If Controller-Side Authentication is used, the DSK bytes 1..2 MUST be obfuscated with zeros (0x00) by the joining node
If Client-Side Authentication (CSA) is used, the DSK bytes 1..4 MUST be obfuscated with zeros (0x00) by the including node.

CC:009F.01.08.11.00B If no authentication is used (S2 Unauthenticated and/or S0 classes are granted only), both joining and including nodes ECDH Public Keys MUST be transmitted in their integrality

3.6.7.6 Security 2 Network Key Get Command

CC:009F.01.09.11.001 This command is used by a joining node to request one key from the including node. One instance of this command MUST be sent for each key that was granted by the including node.

CC:009F.01.09.11.002 The command MUST be sent security encapsulated using the `TempKeyCCM` and `TempPersonalizationString`.

CC:009F.01.09.11.003 This command MUST be ignored unless the Add Node mode is enabled.

CC:009F.01.09.11.004 The Security 2 Network Key Report Command MUST be returned in response to this command unless this command is ignored.

CC:009F.01.09.11.005 This command MUST NOT be issued via multicast addressing.

CC:009F.01.09.11.006 A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY_2							
Command = SECURITY_2_NETWORK_KEY_GET							
Requested Key							

Requested Key (1 byte)

This field is used to request a network key.

CC:009F.01.09.11.007 Only one key MUST be requested at a time, i.e. only 1 bit MUST be set to '1'. This field MUST be encoded according to Table 19.

3.6.7.7 Security 2 Network Key Report Command

This command is used by an including node to transfer one key to the joining node.

CC:009F.01.0A.11.001 The command MUST be sent security encapsulated using the `TempKeyCCM` and `TempPersonalizationString`.

CC:009F.01.0A.11.002 This command MUST be ignored unless the Learn mode is enabled.

CC:009F.01.0A.11.003 The joining node MUST store the received network key in non-volatile memory.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY_2							
Command = SECURITY_2_NETWORK_KEY_REPORT							
Granted Key							
Network Key byte 1							
…							
Network Key byte 16							

Granted Key (1 byte)

CC:009F.01.0A.11.004 This field is used to indicate which Network Key is carried in the command and MUST be encoded according to Table 19

Network Key (16 bytes)

This field carries the granted Network Key.

3.6.7.8 Security 2 Network Key Verify Command

This command is used by a joining node to verify a newly exchanged key with the including node.

CC:009F.01.0B.11.007 This command MUST be sent security encapsulated using the Key that was just exchanged and MUST be encrypted using the derived `KeyCCM` and `PersonalizationString`.

CC:009F.01.0B.11.004 The Security 2 Transfer End Command MUST be returned in response to this command unless it is to be ignored.

CC:009F.01.0B.11.005 The joining node MUST NOT consider the actual key exchange to be complete until a Security 2 Transfer End command has been received from the including node.

CC:009F.01.0B.11.006 This command MUST be ignored if Learn mode and Add Node mode are both disabled.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY_2							
Command = SECURITY_2_NETWORK_KEY_VERIFY							

3.6.7.9 Security 2 Transfer End Command

This command is used by the including node to complete the verification of each individual key exchange while the joining node uses this command to complete the S2 bootstrapping process after all granted keys have been successfully exchanged.

CC:009F.01.0C.11.001 The joining node MUST send this command after all granted keys have been verified.

CC:009F.01.0C.11.002 This command MUST be ignored if Learn mode and Add Node mode are both disabled.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY_2							
Command = SECURITY_2_TRANSFER_END							
Reserved						Key Verified	Key Request Complete

Reserved

CC:009F.01.0C.11.003 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Key Verified (1 bit)

CC:009F.01.0C.11.004 The including node MUST set this flag to '1' if it has successfully verified the Key Verify Command using the newly exchanged network key.

CC:009F.01.0C.11.005 This flag MUST be set to '0' in all other cases.

CC:009F.01.0C.11.009 If this field is set to 0, a receiving node MUST abort S2 bootstrapping and consider that S2 bootstrapping process failed.

Key Request Complete (1 bit)

CC:009F.01.0C.11.006 The joining node MUST set this flag to '1' if it has completed exchanging all granted keys.

CC:009F.01.0C.11.007 This flag MUST be set to '0' in all other cases.

CC:009F.01.0C.11.008 The including node MUST consider S2 bootstrapping to be successfully completed if it receives this command with this flag set to '1' and all keys have been exchanged.

3.6.8 Discovery of Security capabilities commands

A controlling node may discover the Command Class capabilities of a securely included node.

CC:009F.01.00.13.015 The advertised capabilities MAY depend on the actual security level used to request capabilities. Likewise, the advertised capabilities at a given security level may depend on the S2 bootstrapping security level.

3.6.8.1 Security 2 Commands Supported Get Command

This command is used to query the command classes that a joining node supports via secure communication.

CC:009F.01.0D.11.001 Security 2 encryption MUST be used when transmitting this command.

CC:009F.01.0D.11.002 The Security 2 Commands Supported Report Command MUST be returned in response to this command. The response MUST be sent encrypted, using the same Security 2 Class key and encapsulation as was used for this command.

CC:009F.01.0D.11.003 A node receiving this command on its highest granted Security Class MUST respond with the Security 2 Commands Supported Report Command containing all supported secure command classes.

CC:009F.01.0D.11.004 A node receiving this command on any other Security Class than its highest granted Security Class MUST respond with the Security 2 Commands Supported Report Command containing no command classes.

CC:009F.01.0D.11.007 A node receiving the Security Commands Supported Get of the (non S2) Security 0 Command Class (using S0 key and S0 Message Encapsulation Command), MUST respond with the Security 0 Commands Supported Report containing no command classes.

CC:009F.01.0D.11.006 This command MUST NOT be issued via multicast addressing.
A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY_2							
Command = SECURITY_2_COMMANDS_SUPPORTED_GET							

3.6.8.2 Security 2 Commands Supported Report Command

This command is used to advertise the commands that a joining node supports via secure communication. Security 2 encryption MUST be used when transmitting this command.

CC:009F.01.0E.11.001

CC:009F.01.0E.11.002 Transport Service segmentation MUST be used if the command is longer than the available payload length of a single Z-Wave MAC frame.

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SECURITY_2							
Command = SECURITY_2_COMMANDS_SUPPORTED_REPORT							
Command Class 1							
…							
Command Class N							

CC:009F.01.0E.11.003 This command MUST NOT advertise command classes that the joining node can control in other nodes.

CC:009F.01.0E.12.001 Network management user interfaces SHOULD discover control capabilities of a node via the Association Group Information (AGI) Command Class.

As per section 3.6.6, a node may be assigned up to four keys for the S2 Unauthenticated, S2 Authenticated, S2 Access Control and S0 Classes, respectively.

For instance, a light dimmer may accept non-securely transmitted commands if it is not S2 bootstrapped, while the same dimmer will require secure communication for commands if it is S2 bootstrapped. This allows a Security 2 enabled light dimmer to be compatible with first-generation non-secure Z-Wave networks while it will operate fully secure when included in a S2 network at a later time.

In another example, a door lock may be designed to work only with the S0 network key or the S2 Access Control Class key. Inclusion in a network without security bootstrapping will not allow operation of the lock and if bootstrapped with the S2 Access Control Class, S0 encrypted commands will also be ignored.

CC:009F.01.0E.11.00F The Security 0 and Security 2 Command Class MUST NOT be advertised in this command
The Transport Service Command Class MUST NOT be advertised in this command.

CC:009F.01.0E.13.002 A sending node MAY terminate the list of supported command classes with the COMMAND_CLASS_MARK command class identifier.

CC:009F.01.0E.11.007 A receiving node MUST stop parsing the list of supported command classes if it detects the COMMAND_CLASS_MARK command class identifier in the Security 2 Commands Supported Report.

Command Class (N bytes)

This field advertises the command classes that the node supports.

CC:009F.01.0E.11.008 A normal Command Class identifier MUST be one byte long in the range (0x20 – 0xEE).
An extended Command Class identifier MUST be two bytes long where the first byte is in the range (0xF1 – 0xFF), while the second byte is in the range 0x00 – 0xFF.

CC:009F.01.0E.13.003 A joining node MAY advertise extended command classes.
CC:009F.01.0E.11.009 An including node MUST accept extended command classes.

3.7 Supervision Command Class, version 1

The Supervision Command Class allows a sending node to request application-level delivery confirmation from a receiving node. The delivery confirmation includes relevant application-level status information in the confirmation message.

3.7.1 Terminology

The controlling application initiates an operation by sending a Supervision encapsulated command to a supporting node. The supporting node returns an immediate confirmation for the reception while advertising its ability to perform the requested operation. The confirmation may advertise the application status <Working>, <Success>, <No Support> or <Fail> depending on the condition of the supporting node. One or more application status updates may follow later to report problems, delays or the completion of the requested operation.

A door lock application is used as example in the following. Other uses may apply.

A magnetic lock may return the <Success> indication immediately since it takes only a few milliseconds to apply power to the electromagnet. In that case, no status updates are needed later.

An electro-mechanical lock may need time to complete its movement. A <Working> indication is issued along with the expected duration, e.g. 5 seconds. Ideally, the movement is completed and a <Success> indication is returned 5 seconds later. A motor defect or something jamming the movement may prevent the lock from completing the requested operation. A <Fail> indication is returned in that case. If the motor runs slower than anticipated, another <Working> indication may be issued when e.g. 75% of the expected duration has elapsed; reporting the new expected duration. If it runs faster, a <Success> indication may be issued before it was expected. Both cases allow a GUI progress bar to respond instantly rather than waiting for a polling response.

3.7.2 Compatibility considerations

This command class is used as an integrated part of the Security 2 Command Class but may also be used for non-secure applications.

CC:006C.01.00.23.002 The Supervision Command Class MAY be used for solitary commands such as Set, Remove and unsolicited Report commands.

CC:006C.01.00.21.003 The Supervision Command Class MUST NOT be used for session-like command flows such as Get↔Report command exchanges or firmware update.

CC:006C.01.00.23.001 The Supervision Get Command MAY carry multiple commands grouped with the Multi Command encapsulation command.

3.7.2.1 Node Information Frame (NIF)

CC:006C.01.00.21.001 A supporting node MUST always advertise the Supervision Command Class in its NIF and Multi Channel Capability Report, regardless of the inclusion status and security bootstrapping outcome.

3.7.2.2 Encapsulated commands

Supervision Command Class is intended for encapsulating solitary commands; it is to say commands that do not require any response to be returned.

It exists Set type Commands that may optionally require to return a status or optional command such as Notification Report or handshake mechanism. When receiving such a Set type command in a Supervision Get Command:

CC:006C.01.00.21.002
- The Supervision Report MUST be returned by a receiving node.

CC:006C.01.00.23.003
CC:006C.01.00.22.001
- Additional commands MAY be returned. These commands SHOULD be returned before the Supervision Report Command

CC:006C.01.00.51.001 A sending node MUST NOT use handshake mechanisms for Set Commands when using Supervision encapsulation.

CC:006C.01.00.23.004 It is OPTIONAL for a receiving node to return an answer to a Set type Command which requires a response to be returned if received with Supervision encapsulation.

3.7.3 Supervision Get Command

This command is used to initiate the execution of a command and to request the immediate and future status of the process being initiated.

CC:006C.01.01.11.001 The Supervision Report Command MUST be returned in response to this command unless it is to be ignored.

CC:006C.01.01.11.002 The <SUCCESS> status MUST be advertised in the Supervision Report Command if the operation is completed immediately.

CC:006C.01.01.11.003 If the requested operation is accepted but cannot be completed immediately, the <WORKING> status MUST be advertised in the Supervision Report Command along with the expected duration for the operation. The “Status Updates” field MUST be consulted to determine if status updates are to be advertised in the future.

CC:006C.01.01.13.003 This command MAY be issued via multicast addressing.

CC:006C.01.01.11.004 A receiving node MUST NOT ignore the encapsulated command if this command is received via multicast addressing.

CC:006C.01.01.11.005 A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">Command Class = COMMAND_CLASS_SUPERVISION</td></tr>
<tr><td colspan="8">Command = SUPERVISION_GET</td></tr>
<tr><td>Status Updates</td><td colspan="2">Reserved</td><td colspan="5">Session ID</td></tr>
<tr><td colspan="8">Encapsulated Command Length</td></tr>
<tr><td colspan="8">Encapsulated Command 1</td></tr>
<tr><td colspan="8">...</td></tr>
<tr><td colspan="8">Encapsulated Command N</td></tr>
</table>

Reserved

CC:006C.01.01.11.006 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Status Updates (1 bit)

This flag is used to allow a receiving node to advertise application status updates in future Supervision Report Commands.
As an example, this flag must be set to allow a supporting node to immediately advertise the <WORKING> status in a Supervision Report Command and subsequently advertise the <SUCCESS> status in another Supervision Report Command once the operation has been completed.

CC:006C.01.01.11.007 The value of this field MUST comply with Table 21.

Table 21, Supervision Get :: Status Updates

Value	Required behaviour
'0'	Only return a report now
'1'	Return a report now and more later if needed

Session ID (6 bits)

The same command may be received multiple times, e.g. due to retransmissions.

CC:006C.01.01.11.008 A sending node MUST increment this field each time a new unique Supervision Get Command is issued.

CC:006C.01.01.13.004 A sending node MAY use the same Session ID for a multicast and singlecast follow-up carrying the same encapsulated command. A sending node MAY also use the same Session ID for all destinations of singlecast follow-up commands.

CC:006C.01.01.11.00D A receiving node MUST ignore duplicate singlecast commands having the same Session ID and MUST NOT return a Supervision Report to the command, if the command was received without Security encapsulation.

CC:006C.01.01.11.00A A receiving node MUST abort an active operation in favor of a new command with a new Session ID if that new command affects the same resources as the active operation.

CC:006C.01.01.13.001 A receiving node MAY abort an active operation in favor of a new command with a new Session ID even if the new command does not affects the same resources as the active operation, e.g. if the node does not have resources to handle multi-session state management.

Encapsulated Command Length (1 Byte)

This field is used to specify the length of the encapsulated command.

CC:006C.01.01.11.00B The value MUST specify the number of bytes in the Encapsulated Command field.

Encapsulated Command (N bytes)

This field is used to carry an encapsulated command.

CC:006C.01.01.11.00C The length of this field MUST be in the range 1..255 bytes.

CC:006C.01.01.13.002 The field MAY carry a Multi Command Encapsulated Command which contains multiple commands.

3.7.4 Supervision Report Command

This command is used to advertise the status of one or more command process(es).

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SUPERVISION							
Command = SUPERVISION_REPORT							
More Status Updates	Reserved	Session ID					
Status							
Duration							

Reserved

CC:006C.01.02.11.001 This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

More Status Updates

This field is used to advertise if more Supervision Reports follow for the actual Session ID.

CC:006C.01.02.11.002 The value of this field MUST comply with Table 22.

Table 22, Supervision Report :: More Status Updates

Value	Required behaviour
'0'	This is the last report
'1'	More reports follow according to the advertised duration

Session ID (1 Byte)

CC:006C.01.02.11.003 This field MUST carry the same value as the Session ID field of the Supervision Get Command which initiated this session.

Status (8 bit)

This field is used advertise the current status of the command process.

CC:006C.01.02.11.013 This field MUST reflect the actual application status of the received encapsulated command.

CC:006C.01.02.11.004 If a Multi Command encapsulated group of commands is being supervised, an error indication (such as <No Support> or <Fail>) MUST be issued if just one of the Multi Command encapsulated commands triggers an error condition.

CC:006C.01.02.11.005 The <Success> indication MUST signify that all commands carried in a Multi Command encapsulation commands have completed successfully.

CC:006C.01.02.11.006 The value of this field MUST comply with Table 23.

Table 23, Supervision Report :: Status identifiers

Value	Identifier	Description
0x00	NO_SUPPORT	The command is not supported by the receiver. CC:006C.01.02.11.007 A zero Duration value MUST be advertised. CC:006C.01.02.11.008 This identifier MUST be advertised if one or more Multi Command encapsulated commands are not supported.
0x01	WORKING	The command was accepted by the receiver and processing has started. CC:006C.01.02.11.009 A non-zero Duration value MUST be advertised. CC:006C.01.02.11.00A If processing is completed instantly, the receiver MUST skip advertising the WORKING status and return the SUCCESS status instead. CC:006C.01.02.11.00B If Status Updates was set to 1 in the Get Command, this command MUST be followed by another Supervision Report when the duration has elapsed. The new status MUST be one of WORKING, FAIL or SUCCESS. CC:006C.01.02.13.001 The duration MAY be cut short by sending the new Supervision Report before the original predicted duration has elapsed. CC:006C.01.02.11.00C If used for Multi Command encapsulated commands, the advertised Duration value MUST represent the slowest of the commands.

Value	Identifier	Description
0x02	FAIL	The command was accepted by the receiver but processed with a resulting application status which differs from the supervised command requested Examples include but are not limited to: The node may not be ready to perform the requested operation (e.g. a door lock cannot lock if the door is open) or the command processing reached an unexpected situation (e.g. a door lock being jammed while working) or CC:006C.01.02.11.00D there is an application specific limitation (e.g. an irrigation controller which only accepts one open valve at a time). CC:006C.01.02.11.00E A zero Duration value MUST be advertised. This status identifier MUST be advertised if the processing of one or more Multi Command encapsulated commands failed.
0xFF	SUCCESS	The requested command has been completed and the application status is as the supervised command requested. CC:006C.01.02.12.001 A controlling application SHOULD NOT verify the application status, e.g. by sending a Get Command, after receiving this status identifier. CC:006C.01.02.11.00F A zero Duration value MUST be advertised. CC:006C.01.02.11.010 This identifier MUST be advertised if the processing of all Multi Command encapsulated commands was successful.

CC:006C.01.02.11.011 All other values are reserved. Reserved values MUST NOT be used by a sending node and MUST be ignored by a receiving node.

Duration (8 bits)

CC:006C.01.02.11.012 The Duration field MUST advertise the time needed to complete the current operation. The encoding of the Duration field MUST be according to Table 24.

Table 24, Supervision Report::Duration

Duration	Description
0x00	0 seconds. (Already at the Target Value.)
0x01-0x7F	1 second (0x01) to 127 seconds (0x7F) in 1 second resolution.
0x80-0xFD	1 minute (0x80) to 126 minutes (0xFD) in 1 minute resolution.
0xFE	Unknown duration
0xFF	Reserved

3.7.5 Examples and use-cases

3.7.5.1 Set Type commands

CC:006C.01.00.11.002 A supporting node MUST return the Supervision status of solitary Set type commands for all its supported Command Classes.

CC:006C.01.00.11.003 For actuator control Set commands with a corresponding Get type command to read back the value(s), this means that the response codes MUST be used as follows:

- SUCCESS: The application understood the command and completed the requested operation. The values specified in the Set Command would be returned if the supporting node would return an answer to the corresponding Get Command. The only exception are special values (e.g. value to set to the last non-zero level) representing another value.

- WORKING: The application understood the command and started performing the requested operation, but the controller needs to wait before the target value is reached. The supporting node would not return the values specified in the Set Command yet if it returned an answer to the corresponding Get Command

- FAIL: The application understood the command and cannot perform the requested operation (e.g. invalid value or mechanical failure). The supporting node will not return the values specified in the Set Command if it returned an answer to the corresponding Get Command.

The Door Lock Operation Set Command is an example of an actuator control Set command with a corresponding Get Command.

CC:006C.01.00.11.004 For actuator control Start and Stop commands or commands without a corresponding Get type read back, this means that the response codes MUST be used as follows:

- SUCCESS: The application understood the command and completed the requested operation; i.e. successfully started or stopped the requested operation.

- FAIL: The application did not understand the command or could not perform the requested operation (if applicable)

The Multilevel Switch Start Level Change Command, Multilevel Switch Stop Level Change Command and the Powerlevel Test Node Set are examples of actuator control Start/Stop commands.

CC:006C.01.00.11.005 For configuration Set commands, this means that the response codes MUST be used as follows:

- SUCCESS: The application understood the command and accepted all the parameter(s) and value(s). The values specified in the Set Command would be returned if the supporting node would return an answer to the corresponding Get Command.

- FAIL: The application had one or more error while parsing or applying the parameters (e.g. the command was partially or completely ignored due to invalid values)

The Door Lock Configuration Set Command, Configuration Set and Association Set Command are examples of configuration commands.

3.7.5.2 Powerlevel Test Node Set Command

CC:006C.01.00.12.001 The Powerlevel Test Node Set Command SHOULD be considered as a Start/Stop command or command without a corresponding Get Command. The corresponding Get Command (Powerlevel Test Node Get Command) is used to query subsequent results and Supervision cannot replace the need for a controller to issue a subsequent Get Command.

CC:006C.01.00.12.002 A receiving node SHOULD return SUCCESS when starting the Powerlevel Test Node test.

3.7.5.3 Report/Notification Type Commands

CC:006C.01.00.13.003 A supporting node MAY use Supervision encapsulation for issuing an unsolicited Report/Notification type Command. For example, a sleeping node issuing a sensor reading receives a delivery confirmation immediately and can return to sleep as soon as the Supervision Report is received.

CC:006C.01.00.11.006 A node MUST return a Supervision status of a Report/Notification type command for any Command Classes, regardless whether it is supported, controlled or neither.

CC:006C.01.00.12.003 A controlling node SHOULD return a SUCCESS or NO_SUPPORT status when receiving such commands.

Remove Type commands

CC:006C.01.00.11.007 A supporting node MUST return the Supervision status of solitary Remove type commands for all its supported Command Classes. The remove type commands must be treated as Set type of commands.

For Remove command, this means that the response codes MUST be used as follows:

- SUCCESS: The application understood the command and removed the specified parameter(s) and values. The values specified in the Remove Command would not be returned if the supporting node would return an answer to the corresponding Get Command.
- FAIL: The application had one or more error while applying the command (e.g. the command was partially or completely ignored due to invalid values)

The Association Remove Command and Schedule Remove Command are examples of remove commands.

Supervision Command Class, version 2

The Supervision Command Class allows a sending node to request application-level delivery confirmation from a receiving node. The delivery confirmation includes relevant application-level status information in the confirmation message.

Compatibility considerations

CC:006C.02.00.21.001 Compatibility consideration requirements from version 1 MUST also be observed by a version 2 supporting node.

CC:006C.02.00.21.002 Supervision Command Class, version 2 is backwards compatible with Supervision Command Class, version 1. Fields and commands not described in this version MUST remain unchanged from version 1.

The Supervision Report Command has been extended to enable expedited message delivery between Wake Up Destination and Wake Up Node (refer to the Wake Up Command Class). It leverages sleeping nodes using Supervision encapsulation to tell them to initiate a Wake Up Period.

Supervision Report Command

This command is used to advertise the status of one or more command process(es).

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_SUPERVISION							
Command = SUPERVISION_REPORT							
More Status Updates	Wake Up Request	Session ID					
Status							
Duration							

All fields not described below remain unchanged from version 1

Wake Up Request (1 bit)

CC:006C.02.02.11.001 This field is used to indicate to the receiving node that it MUST start a Wake Up Period.

CC:006C.02.02.11.002 If the receiving node does not support the Wake Up On Demand functionality, this field MUST be ignored.

The Wake Up On Demand functionality is part of the Wake Up Command Class, version 3.

CC:006C.02.02.11.003 A node supporting the Wake Up On Demand functionality MUST return a Wake Up Notification Command if the Wake Up destination node issued a this command with the Wake Up Request bit set to 1.

CC:006C.02.02.11.004 A node supporting the Wake Up On Demand functionality MUST ignore the Wake Up Request field if the Supervision Report is not issued by the Wake Up Destination.

Wake Up on Demand functionality

The Wake Up on Demand functionality is shown in Figure 20. This functionality allows the controller to issue important commands to the sleeping device before its next Wake Up Period.

CC:006C.02.02.13.001 When the controlling node receives a Supervision encapsulated frame from the sleeping node, it MAY speed up the delivery of some important commands by setting the *Wake Up Request* bit to 1, if the sleeping node supports this functionality.

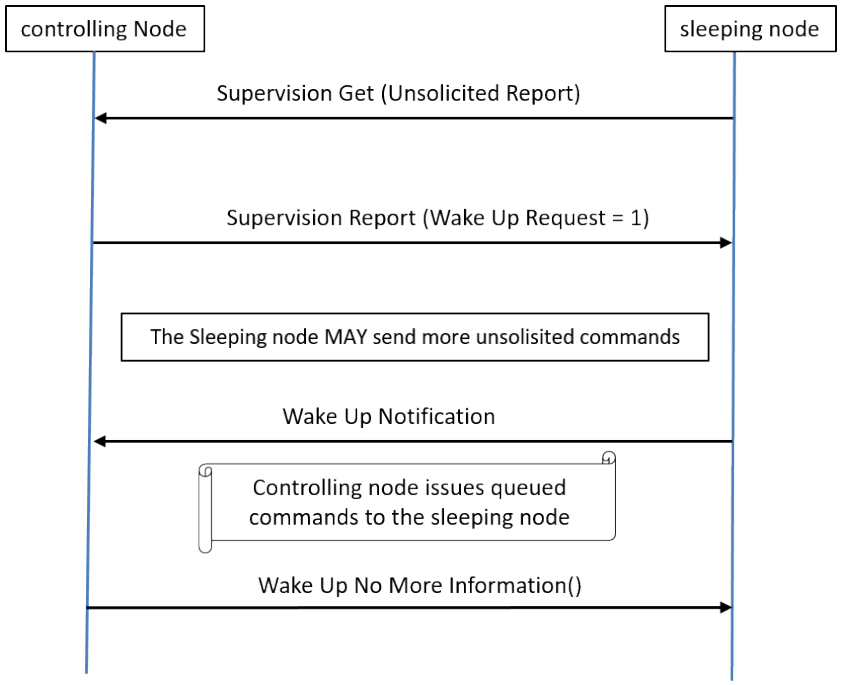

Figure 20, Wake Up on Demand functionality

3.8 Transport Service Command Class, version 1 [OBSOLETED]

THIS COMMAND CLASS HAS BEEN OBSOLETED

New implementations MUST use the Transport Service Command Class version 2.

The Transport Service Command Class Version 2 redefines the frame formats used by the command class.

3.9 Transport Service Command Class, version 2

The Transport Service Command Class supports the transfer of datagrams larger than the Z-Wave frame.

The Transport Service Command Class, version 2 is defined by [16].

The following sections provide additional requirements and frame flows.

3.9.1 Compatibility considerations

A node supporting the Transport Service Command Class, version 2:

CC:0055.02.00.21.001
- MUST NOT send Transport Service segments with the Payload field longer than 39 bytes.

CC:0055.02.00.21.002
- MUST accept datagrams up to 117 bytes long (3 segments of 39 bytes each).

CC:0055.02.00.23.001
- MAY accept larger datagrams.

3.9.1.1 Node Information Frame (NIF)

A node supporting the Transport Service Command Class, version 2:

CC:0055.02.00.21.003
- MUST always advertise this Command Class in its NIF, regardless of the inclusion status and security bootstrapping outcome.

CC:0055.02.00.21.004
- MUST NOT advertise this Command Class in its S0/S2 Supported Command Class list or in the Multi Channel End Point capabilities.

3.9.2 Example Frame flows

CC:0055.02.00.11.001 A supporting node MUST comply with the following frame flows.

3.9.2.1 As things should always work – the default case

- Node A initiates a 117-byte frame transmission to Node B
  - Node A sends FirstSegment(datagram size = 117, Session ID = 10, Payload = bytes 1..39)
  - Node B receives the FirstSegment with valid Transport Service FCS (16-bit checksum) and valid MPDU FCS (8 or 16 bits checksum, depending on transmission speed)
    - Node B creates a tracking list for the datagram; bytes 1..39 are marked as received
    - Node B starts segment rx timer
  - Node A sends SubsequentSegment(datagram size = 117, datagram offset = 39, Session ID = 10, Payload = bytes 40..78)
  - Node B receives the SubsequentSegment with valid Transport Service and MPDU FCS
    - Node B updates the tracking list for the datagram; bytes 40..78 are marked as received
    - Node B (re-)starts segment rx timer
  - Node A sends SubsequentSegment(datagram size = 117, datagram offset = 78, Session ID = 10, Payload = bytes 79..117)
    - Node A starts a segment_complete tx timer
  - Node B receives SubsequentSegment with valid Transport Service and MPDU FCS
    - Node B updates the tracking list for the datagram; bytes 79..117 are marked as received, indicating that this was the last segment
    - Node B checks the tracking list for missing segments; none found
  - Node B sends SegmentComplete(Session ID = 10)
  - Node A receives SegmentComplete(Session ID = 10) with valid MPDU FCS (8 or 16 bits checksum, depending on transmission speed)

3.9.2.2 Losing first segment of a long message

- Node A initiates a 117-byte frame transmission to Node B:
  - Node A sends FirstSegment(datagram size = 117, Session ID = 10, Payload = bytes 1..39).

- Node B receives FirstSegment invalid Transport Service or MPDU FCS (the command is ignored).
- Node A sends SubsequentSegment(datagram offset = 39)
- Node B receives the SubsequentSegment correctly (valid Transport Service and MPDU FCS)
- Node B sends SegmentWait(Pending segments=0) because no session is open.
- Node A waits and restarts the transmission from the FirstSegment.

3.9.2.3 Losing subsequent segment

- Node A initiates a 117-byte frame transmission to Node B
  - Node A sends FirstSegment(datagram size = 117, Session ID = 10, Payload = bytes 1..39)
  - Node B receives the FirstSegment correctly
    - Node B creates a tracking list for the datagram; bytes 1..39 are marked as received
    - Node B starts segment rx timer
  - Node A sends SubsequentSegment(datagram offset = 39)
  - Node B receives SubsequentSegment with invalid Transport Service or MPDU FCS. (the command is ignored)
  - Node A sends SubsequentSegment(datagram offset = 78)
    - Node A starts segment_complete tx timer
  - Node B receives SubsequentSegment correctly
    - Node B updates the tracking list for the datagram, indicating that this was the last segment
    - Node B checks tracking list for the datagram; bytes 40..78 are missing
  - Node B sends SegmentRequest(datagram offset = 39)
  - Node A receives SegmentRequest(datagram offset = 39) correctly
  - Node A send SubsequentSegment(datagram offset = 39)
  - Node B receives SubsequentSegment(datagram offset = 39) correctly
    - Node B updates the tracking list for the datagram
    - Node B checks the tracking list for missing segments; none found
    - Node B clears segment rx timer
  - Node B sends SegmentComplete(Session ID = 10)
  - Node A receives SegmentComplete(Session ID = 10) correctly

3.9.2.4 Losing last segment

- Node A initiates a 117-byte frame transmission to Node B
  - Node A sends FirstSegment(datagram size = 117, Session ID = 10, Payload = bytes 1..39)
  - Node B receives FirstSegment
    - Node B creates a tracking list for the datagram; bytes 1..39 are marked as received
    - Node B starts segment rx timer
  - Node A sends SubsequentSegment(datagram offset = 39)
  - Node B receives SubsequentSegment correctly
    - Node B updates the tracking list for the datagram; bytes 40..78 are marked as received
    - Node B (re-)starts segment rx timer
  - Node A sends (the last) SubsequentSegment(datagram offset = 78)
    - Node A starts segment_complete tx timer
  - Node B receives SubsequentSegment with invalid Transport Service or MPDU FCS. (the command is ignored)
  - Node B segment rx timer times out.
    - Node B checks tracking list for the datagram; bytes 79..117 are missing
  - Node B sends SegmentRequest(datagram offset = 78)
    - Node B starts a segment rx timer to wait for the SubsequentSegment frame
    - If the segment rx timer times out, Node B bails out: discard all received segments and return to idle (e.g. sender may be down or sleeping)
  - Node A receives SegmentRequest(datagram offset = 78)
  - Node A sends SubsequentSegment(datagram offset = 78)
  - Node B receives SubsequentSegment(datagram offset = 78)
    - Node B updates the tracking list for the datagram; bytes 79..117 are marked as received
    - Node B checks tracking list for missing segments; none found
  - Node B sends SegmentComplete(Session ID = 10)
  - Node A receives SegmentComplete(Session ID = 10) correctly
    - Node A stops the segment_complete tx timer

3.9.2.5 Losing SegmentComplete

- Node A initiates a 117-byte frame transmission to Node B
  - Node A sends FirstSegment(datagram size = 117, Session ID = 10, Payload = bytes 1..39)
  - Node B receives FirstSegment correctly
    - Node B creates tracking list for the datagram; bytes 1..39 are marked as received
    - Node B starts a segment rx timer
  - Node A sends SubsequentSegment(datagram offset = 39)
  - Node B receives SubsequentSegment(datagram offset = 39) correctly
    - Node B updates the tracking list for the datagram; bytes 40..78 are marked as received
    - Node B (re-)starts segment rx timer
  - Node A sends SubsequentSegment(datagram offset = 78)
    - Node A starts a segment_complete timer
  - Node B receives SubsequentSegment(datagram offset = 78) correctly.
    - Node B updates the tracking list for the datagram; bytes 79..117 are marked as received, indicating that this was the last segment
    - Node B checks tracking list for missing segments; none found
  - Node B sends SegmentComplete(Session ID = 10)
  - Node A receives SegmentComplete with an invalid MPDU FCS. (the command is ignored)
  - Node A segment_complete tx timer times out
    - Node A sends SubsequentSegment(datagram offset = 78) one more time
      - Node A starts a segment_complete timer again.
      - If the segment_complete timer times out, Node A bails out: return “Error” callback to calling application
  - Node B receives SubsequentSegment(datagram offset = 78) correctly
    - Node B updates the tracking list for the datagram; bytes 79..117 are marked as received, indicating that this was the last segment
    - Node B checks tracking list for missing segments; none found
  - Node B sends SegmentComplete(Session ID = 10) once more
  - Node A receives SegmentComplete(Session ID = 10) correctly
  - Node A stops the segment_complete timer

References

[1] Silicon Labs , SDS10242, Software Design Spec., Z-Wave Device Class Specification.
[2] IETF RFC 4861, Neighbor Discovery for IP version 6 (IPv6), http://tools.ietf.org/pdf/rfc4861.pdf
[3] IETF RFC 3122, Extensions to IPv6 Neighbor Discovery for Inverse Discovery Specification, http://tools.ietf.org/pdf/rfc3122.pdf
[4] IETF RFC 2119, Key words for use in RFCs to Indicate Requirement Levels, http://tools.ietf.org/pdf/rfc2119.pdf
[5] IETF RFC 2460, Internet Protocol, Version 6 (IPv6) Specification, http://tools.ietf.org/pdf/rfc2460.pdf
[6] IETF RFC 4291, IP Version6 Addressing Architecture, http://tools.ietf.org/pdf/rfc4291.pdf
[7] Silicon Labs , SDS11846, Z-Wave Plus Role Types Specification.
[8] Silicon Labs , SDS11847, Z-Wave Plus Device Types Specification.
[9] Silicon Labs, SDS14224, Z-Wave Plus v2 Device Type Specification

[10] Graphical UI elements, http://en.wikipedia.org/wiki/Graphical_user_interface_elements
[11] Silicon Labs , SDS13425. Z-Wave Plus Assigned Manufacturer IDs.
[12] Silicon Labs , SDS13548, List of defined Z-Wave Command Classes
[13] Silicon Labs , SDS13781, Z-Wave Application Command Class Specification
[14] Silicon Labs , SDS13782, Z-Wave Management Command Class Specification
[15] Silicon Labs , SDS13784, Z-Wave Network-Protocol Command Class Specification
[16] ITU-T Recommendation G.9959 (01/2015), Short range narrow-band digital radiocommunication transceivers – PHY, MAC, SAR and LLC layer specifications.
[17] B. Barak, S. Halevi: A model and architecture for pseudo-random generation with applications to /dev/random. IACR eprint 2005/029.
[18] Federal Information Processing Standards Publication 197, November 26, 2001: Advanced Encryption Standard (AES).
[19] S. Matyas, C. Meyer, J. Oseas: Generating strong one-way functions with cryptographic algorithm. IBM Technical Disclosure Bulletin, 27(1985), pp. 5658-5659.
[20] NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation - Methods and Techniques, 2001.
[21] N. Ferguson, B. Schneier, T. Kohno: Cryptography Engineering, John Wiley & Sons, ISBN: 9780470474242, 2010. Chapter 14.
[22] R. Moskowitz, Ed., draft-moskowitz-hip-dex2 work in progress
[23] NIST, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Special Publication 800-38B.
[24] NIST, Recommendation for Block Cipher Modes of Operation: Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, Special Publication 800-38C
[25] NIST, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, Special Publication SP800-90A.
[26] Whiting, D., "Counter with CBC-MAC (CCM)", RFC 3610, September 2003.
[27] D.J. Bernstein, A state-of-the-art Diffie-Hellmann function, http://cr.yp.to/ecdh.html
[28] Silicon Labs , SDS13937, Node Provisioning QR Code Format (S2, Smart Start)