SPsec301 – Generic Mapping

This document defines a generic mapping of SPsec functions and methods to communication systems capable of exchanging data blocks of various sizes.

The SPsec specifications are divided into the following documents:

- **SPsec101** – Small-Packet Network Security Concept
  Introduction to the security concept of SPsec.
- **SPsec102** – Small-Packet Network Security Glossary
  Terminology and references used by SPsec.
- **SPsec201** – Small-Packet Network Security Generic Specification
  Network independent specification of the SPsec technology and methods.
- **SPsec301** – Small-Packet Network Security Generic Mapping
  Mapping SPsec to generic serial point to point communication, including Modbus, CANopen.
- **SPsec302** – Small-Packet Network Security CAN FD Mapping
  Mapping SPsec to CAN FD, supporting CANopen FD and J1939 FD.

Version 1.01 of 12-NOVEMBER-2025, jointly authored by

www.em-sa.com
Embedded Systems Academy GmbH
Bahnhofstraße 17
30890 Barsinghausen, Germany

ivesk.hs-offenburg.de
Hochschule Offenburg
Badstraße 24
77652 Offenburg, Germany

This project has been funded as part of the Central Innovation Program for SMEs (ZIM) by the Federal Ministry for Economic Affairs and Climate Action (BMWK).

All rights reserved. No part of the contents of this document may be reproduced without the prior written consent of the authors, except for the inclusion of brief quotations in a review.

The authors are not liable for defects or indirect, incidental, special, or consequential damages, including loss of anticipated profits or benefits, arising from the use of this document or warranty breaches, even if advised of such possibilities.

The information presented in this book is believed to be accurate. Responsibility for errors, omission of information, or consequences resulting from the use of this information cannot be assumed by the authors.

Contents

1 General Settings

This section summarizes the basic requirements to operate SPsec on generic communication systems capable of exchanging data blocks.

1.1 Basic Protocol Requirements

This SPsec mapping supports communication systems supporting a client – server communication model with different data sizes. The system must support a method to clearly mark data blocks exchanged as belonging to SPsec communication to differentiate from the remaining communication.

- Custom serial protocols should have some meta information regarding content, to mark SPsec communication.
- Object Dictionary based fieldbus systems like CANopen can link Object Dictionary entries of type DOMAIN to SPsec.
- Register based fieldbus systems like Modbus can reserve a variable-length transfer register for SPsec communication.

1.2 Payload sizes

The SPsec generic implementation requires a minimum payload size of 22 bytes for handling the secure session on the control plane. This allows SPsec registers to have a maximum size of 12 bytes.

The Security Stamp defined in section 2.9 used for the data plane has a size of 10 bytes. Therefore, the payload size must be at least 10 bytes longer than the biggest data block exchanged securely.

The maximum payload size is of the generic mapping is 250 bytes.

1.3 Timestamp Synchronization

Not used with generic mapping.

1.4 Use of Nonces

Cryptographic methods often require a nonce (number used once) of a specific or minimum size.

A shared counter of 32 bits is the basis for the nonce.

If more bits are needed to reach a certain nonce size, these are filled with the lowest bytes of the pre-shared salt of the key used.

2 Constants, Data Types and Parameters

2.1 Participant ID

The generic mapping only implements secure client-server communication. Therefore, a Participant ID is not required.

2.2 SPsec Participant Status

Unsigned, 8 bit:

- Bit 0-3: Status as defined in SPsec Generic Specification.
- Bit 4-6: Reserved (0)
- Bit 7: Alert – set on FSA transition “Security Event”

2.3 Parameter Register Number

Unless otherwise specified, all values are unsigned.

2.3.1 Registers 20h to 2Fh: Configurator session Keys and Seed Key

When setting / writing keys it is important that the key, the pre-shared salt and the key ID match to each other as they are a data set. To ensure the integrity of key data sets, the configurator shall first invalidate the Key ID (set to all FFh), then write the key and the pre-shared salt and only write the new Key ID last.

2.3.1.1 21h: Provisioning Key

256 bits, array of bytes. Read-only if set (not all FFh). If not set, can be written once based on a Zero Key session.

2.3.1.2 22h: Integrator Key

256 bits, array of bytes. Read-only if set (not all FFh). If not set, can be written once based on a Provisioning Key session.

2.3.1.3 23h: Seed Key

256 bits, array of bytes. Read-write and can be written by sessions based on Provisioning or Integrator key.

2.3.2 Registers 30h to 3Fh: Key specific pre-shared salt

2.3.2.1 31h: Provisioning Key Salt

64 bits, array of bytes. Same access type as Provision Key.

2.3.2.2 32h: Integrator Key Salt

64 bits, array of bytes. Same access type as Integrator Key.

2.3.2.3 33h: Seed Key Salt

64 bits, array of bytes. Same access type as Seed Key.

2.3.3 Registers 40h to 4Fh: Key ID

2.3.3.1 41h: Provisioning Key ID

32 bits, unsigned. Same access type as Provision Key.

2.3.3.2 42h: Integrator Key ID

32 bits, unsigned. Same access type as Integrator Key.

2.3.3.3 43h: Seed Key ID

32 bits, unsigned. Same access type as Seed Key.

2.3.4 Registers 50h-5Fh: SPsec Information

2.3.4.1 50h: SPsec Status

8 bits, unsigned, read-only, contains the current SPsec status information as defined in SPsec 201.

2.3.4.2 51h: SPsec Last Security Event

16 bits, unsigned, read-only, contains the last security event occurred as defined in 8.2. Default value is zero (NO_SEC_EVENT).

2.3.4.3 58h: SPsec Core Version Information

String, read-only, contains a version string of the SPsec201 document version used for implementation.

2.3.4.4 59h: SPsec Mapping Version Information

String, read-only, contains “301-[V]” for implementations based on this document. Replace [V] with document version number from first page.

2.3.5 Registers 60h-7Fh: SPsec Configuration

2.3.5.1 60h: Participant ID

Not used with generic mapping.

2.3.5.2 61h: Secure Heartbeat Timing

Not used with generic mapping.

2.3.5.3 62h: Secure Heartbeat Monitoring

Not used with generic mapping.

2.3.5.4 63h: Sync Role Activation

Not used with generic mapping.

2.3.5.5 7Fh: Manufacturer Reset to Default

32 bits, unsigned, write-only if session is based on the Integrator Key.

Writing the value of 1D04E5E1h activates the mechanism to restore the device to the manufacturer default settings on the next power cycle. All keys besides the Provisioning key will be erased.

2.3.6 Registers 80h-8Fh: SPsec Device Information

2.3.6.1 81h: Device Identification

String, read-only.

2.3.6.2 82h: MCU Serial Number

128 bits, unsigned, read-only.

2.3.7 Registers 90h-9Fh: Code Updates

2.3.7.1 90h: Code Update Capabilities

32 bits, unsigned, read-only.

If bit 0 is set, the device offers code update capabilities.

Bits 1 to 23 are reserved, bits 24 to 31 are manufacturer specific.

2.3.7.2 91h: Public Authentication Key

Read-only, tbd.

2.3.7.3 92h: Code Update File

Write-only, tbd.

2.4 Keys

Keys are 256 bits in size. If a cryptographic function uses a smaller key size, then only the least significant bits corresponding to that key size are used.

Data type: Unsigned, 256 bits.

2.5 Pre-shared Salt

Pre-shared salt is generated with each key and is 64 bits in size.

Data type: Unsigned, 64 bits.

2.6 Key ID

The Key ID is an unsigned value of 32 bits.

Data type: Unsigned, 32 bits.

2.7 Key Selector

As defined in SPsec Generic Specification.

Data type: Unsigned, 8 bits.

2.8 Security Event Codes

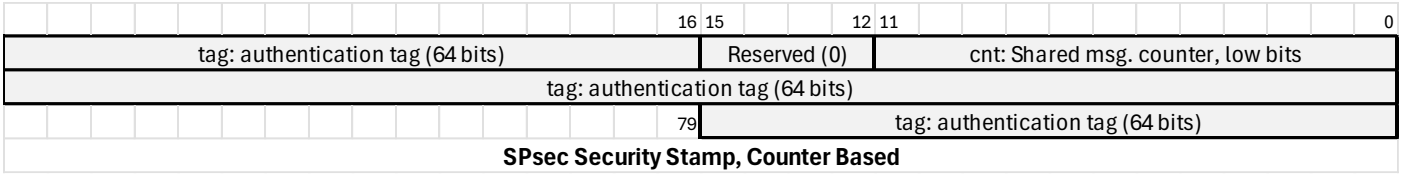
As defined in SPsec Generic Specification.

Data type: Unsigned, 8 bits.

2.9 Security Stamp

The Security Stamp is a record of 10 bytes (80 bits).

Bit 0 – 11,12 bits: least significant bits of shared counter
Bit 12 – 15, 4 bits: reserved (set to 0)
Bit 16 – 79, 64 bits: authentication tag



The authentication tag comes from the AEAD interface described in section 3.4. Inputs to the AEAD interface include a nonce and associated data.

The nonce is a record as defined in 1.4.

The associated data is the data field without Security Stamp (only if encryption is not used)

2.10 Uniqueness Value: Shared Message Counter

The Configurator SPsec sessions use a synchronized shared message counter. It is 32 bits, unsigned. To start with a non-deterministic value, the counter is initialized with random values from Hello messages. See section 6.1.4 for details.

The counter is incremented BEFORE every message sent by either Client or Server.

In the protocols a truncated version of the least significant 8 bits is exchanged to confirm synchronization.

Data type: Unsigned, 32 bit.

2.11 Uniqueness Value: Global Synchronized Timestamp

Not used with generic mapping.

2.12 Control Plane Message Types

Values defined as constants, data type: Unsigned, 8 bit.

0: CPMT_SESS_HELLO
1: CPMT_SESS_FINISH
2: CPMT_SESS_RDINIT
3: CPMT_SESS_RDSEG
4: CPMT_SESS_WRINIT
5: CPMT_SESS_WRSEG
6: CPMT_SESS_TERMINATE
7: Reserved
8: CPMT_AUTH_RD
9: CPMT_AUTH_WR
10: Reserved
11: Reserved
12: CPMT_INTERN_EVT

3 Cryptographic Primitives

3.1 True Random Number Generator

A true random number generator is required to achieve the highest security level. If this is not available and software based random numbers are used, the maximum SPsec security level reachable is limited.

3.2 Secure Key Storage

If secure key storage is not available (protecting keys even when physical chip data extraction services are used), then the maximum SPsec security level reachable is limited.

3.3 KDF – Key Derivation Function

The interface is HKDF compatible using one of the following methods:

- SHA256
- ASCON-HASH

For HKDF the parameters are

- Input key: use case specific
- Input key size: 256 bits
- Salt: use case specific
- Salt size: flexible
- Output key size: 256 bits

The output key inherits the pre-shared salt from the input key.

3.4 AEAD – Authenticated Encryption with Associated Data

This interface uses one of the following methods with tag sizes of 64 bits:

- AES-GCM (using 256 bits key size)
- ChaCha20-Poly1305 (using 256 bits key size)
- ASCON-128

All devices in one system must be pre-configured to use the same methods.

If the nonce available is not as long as required by the AEAD method, remaining/unused bytes are set to the pre-shared salt.

4 Key Derivations

This chapter defines the input keys and salt values used for the KDF.

4.1 Zero Key

The Zero key is a key with all values set to zero, the session based on it is therefore unsecure. It is used to derive a session key when starting an unsecure configuration session.

4.2 Provisioning Key

This is a device unique key. It is used to derive a session key when starting a configuration session.

4.3 Integrator Key

Depending on use case, this is a device specific or shared key. It can only be written once with a secure session based on the provisioning Key. It is used to derive a session key when starting a configuration session.

4.4 Seed Key

Not used with generic mapping.

4.5 Odd and Even Communication Key

Not used with generic mapping.

4.6 Session Key

The SPsec session key is based on the Provisioning, Integrator, Seed or Zero key. The salt used is a concatenation of both random values from Client and Server.

4.7 Parameter Authentication Key

The temporary key required by this service is derived from the Seed key.

5 Cryptographic Functions and Modules

This chapter provides an overview of how the functions are mapped generically. For a detailed definition of the values exchanged, see the next chapter.

Notes on data covered by cryptographic functions:

- Where used, encryption and decryption is only applied to the data field. The Security Stamp or authentication tag are exempt from encryption/decryption.
- The entire data field without any meta data is used when calculating the authentication code.
- Data field size is not authenticated. This is not a security risk, because
    - If the data field is shorter than expected, then authentication will fail.
    - If the data field is longer than expected, then either additional bytes are ignored and not passed on to higher layers or authentication will fail.

5.1 SPsec Session

In generic mapping, this is used by the Configurator (Client) for secure 1:1 sessions with one Participant (Server).

During the Hello phase, the communication partners exchange the key selector and random values of 128 bits. These are used to derive a session key. In the Finish phase, the communication partners exchange 64 bit authentication tags generated based on the previous communication and the session key generated.

5.2 SPsec Parameter Authentication

In generic mapping, this can be used by the Configurator (Client) to read a single authenticated parameter from the Participant (Server). This is primarily used to read an authenticated identification entry.

5.3 SPsec Multi-Participant Grouping

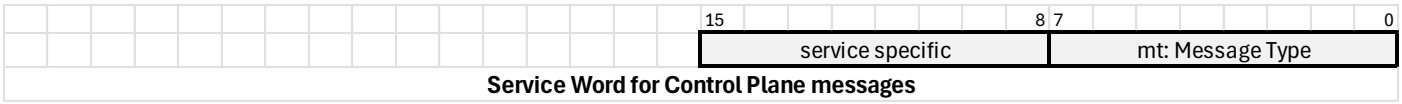
Not used with generic mapping.

5.4 SPsec Heartbeat

Not used with generic mapping.

6 External Control Plane Protocol

This section maps the messages of the control plane for generic mapping. Each data block exchanged containing external control plane data starts with a service word of 16 bits.



The message type values are defined in 2.12.

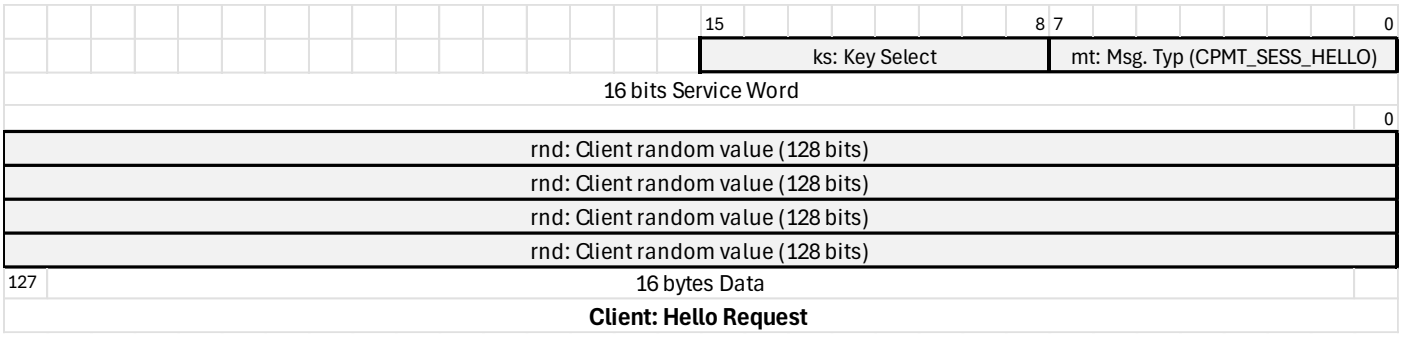
6.1 SPsec Session Start

The timeouts used are:

- Session response timeout (on Client): 100ms
- Overall session timeout (on Server): 10 seconds

6.1.1 Client Hello Request

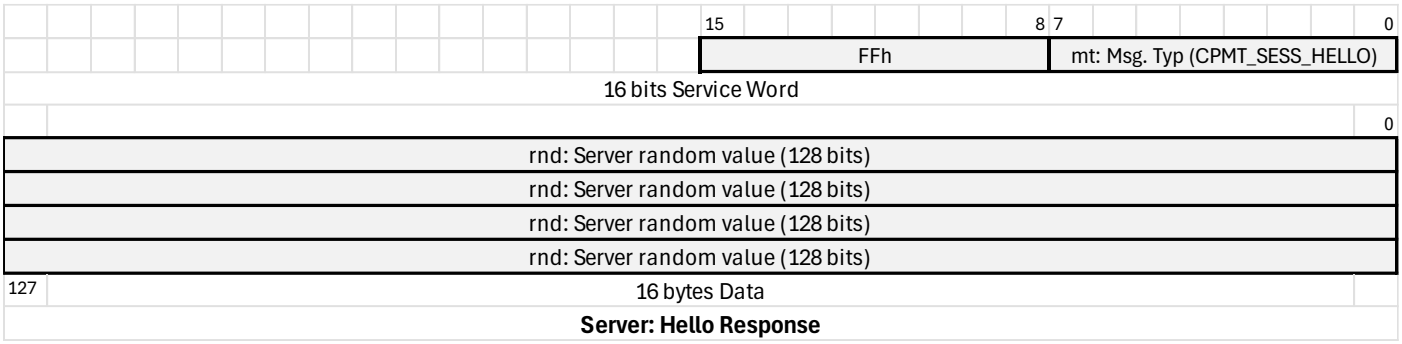
This request comes from the Configurator and is addressed to a Particpant.



The key select values are defined in the SPsec Generic Specification document. This service supports connections based on the Zero, Provisioning, Integrator or Seed key.

6.1.2 Server Hello Response

This response comes from the Participant that received a request and it is addressed to the Configurator.



Both the Client and Server derive a session key based on the information exchanged in the Hello messages. The use case specific KDF parameters for the derivation of the session key are:

- input key: specified by ks from the Hello Request (limited to Zero, Provisioning, Integrator or Seed key)
- salt: concatenated rnd values of Client Hello Request and Server Hello Response
- salt size: 256 bits

The global shared 32bit message counter cnt is initialized with 32bit truncated value of both rnd values, XORed. This ensures a non-deterministic initial value.

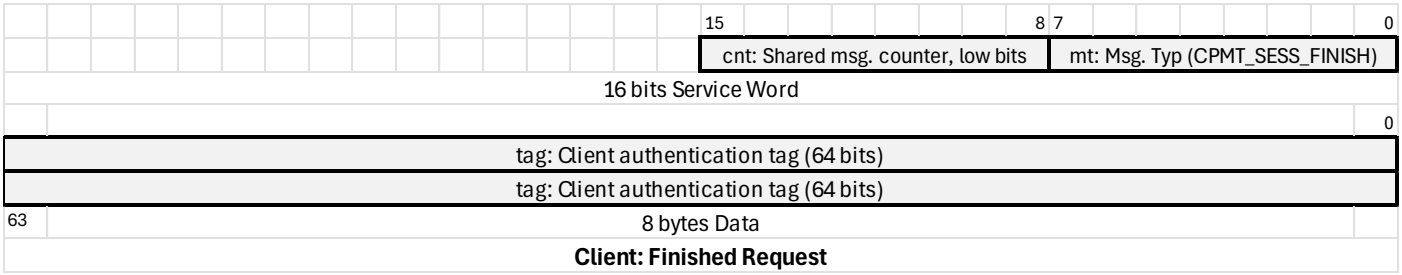
6.1.3 Client Finished Request

This request comes from the Configurator and is addressed to a Participant.

The AEAD parameters for the calculation of the authentication tag are:

- key: the session key derived for this session
- nonce: the shared message counter as described in 2.10
- plaintext (encrypt): none
- cyphertext (decrypt): none
- associated data: concatenation of
    - data field from the Client Hello Request

- data field from the Server Hello Response
- tag (encrypt): generated and returned
- tag (decrypt): used to verify

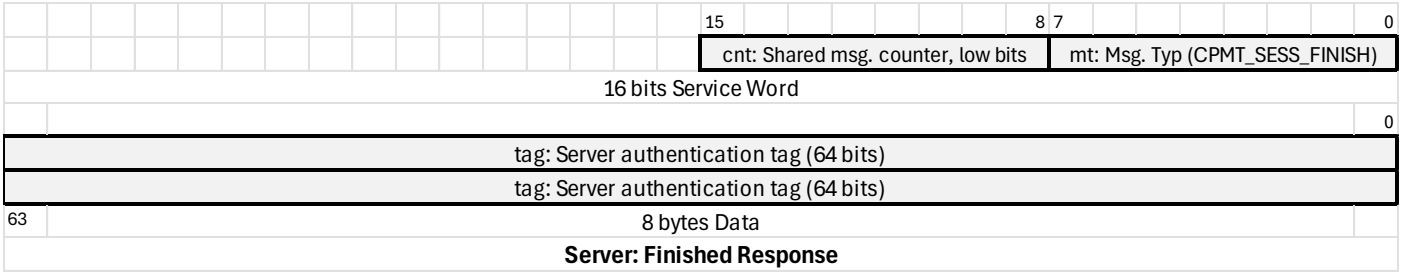


6.1.4 Server Finished Response

This response comes from the Participant that received a request and it is addressed to the Configurator.

The AEAD parameters for the calculation of the authentication tag are:

- key: the session key derived for this session
- nonce: the shared message counter as described in 2.10.
- plaintext (encrypt): none
- cyphertext (decrypt): none
- associated data: concatenation of
  - data field from the Client Hello Request
  - data field from the Server Hello Response
  - data field from the Client Finished Response
- tag (encrypt): generated and returned
- tag (decrypt): used to verify



6.2 SPsec session Register Read Access

In this communication mode, the data field up to the authentication tag is encrypted.

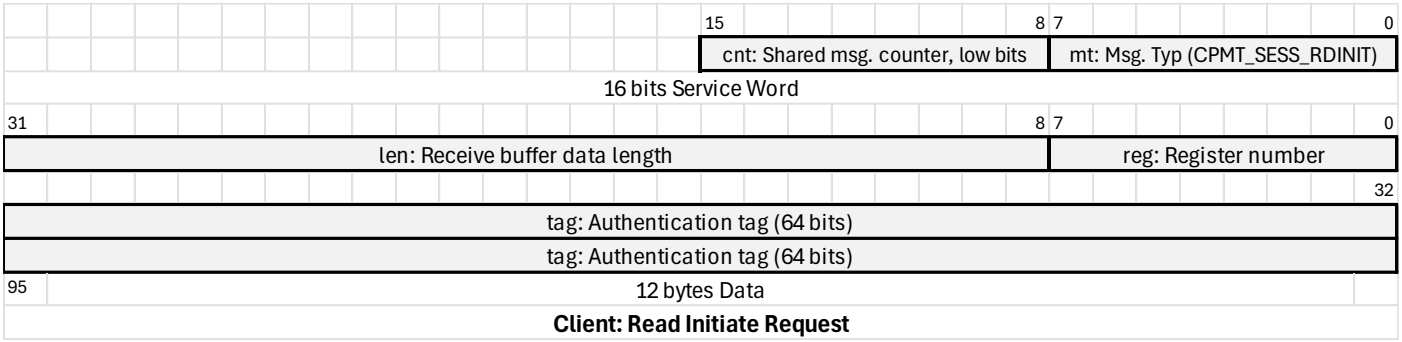
The AEAD parameters for the calculation of the authentication tag are:

- key: the session key derived for this session
- nonce: the shared message counter as described in 2.10.
- plaintext (encrypt): the data field up to the authentication tag
- cyphertext (decrypt): the data field up to the authentication tag
- associated data: none

- tag (encrypt): generated and returned
- tag (decrypt): used to verify

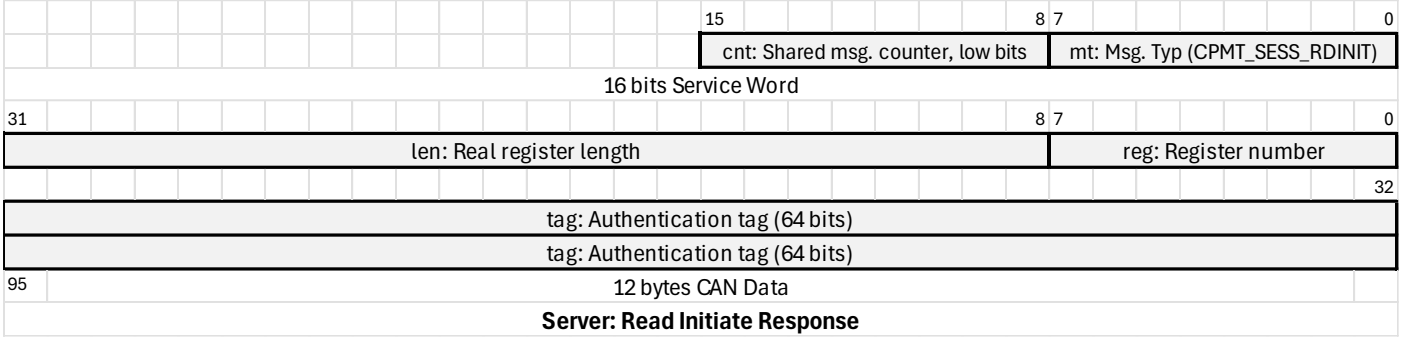
6.2.1 Client Read Initiate Request

This request comes from the Configurator and is addressed to a Particpant.



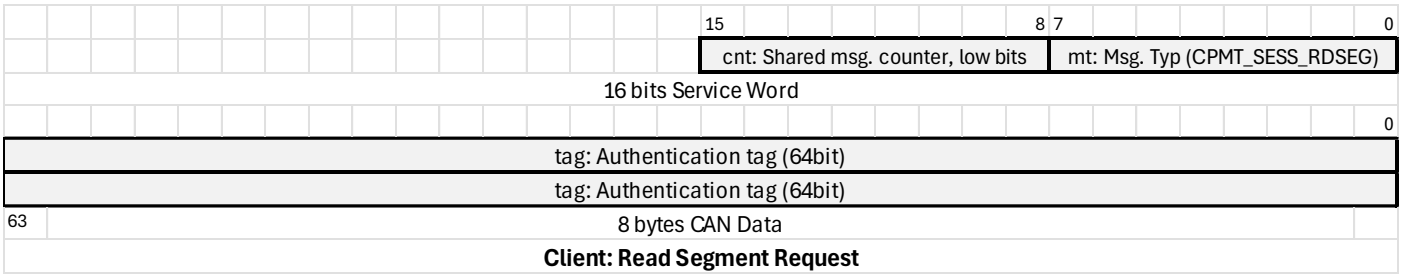
6.2.2 Server Read Initiate Response

This response comes from the Participant that received a request and it is addressed to the Configurator.



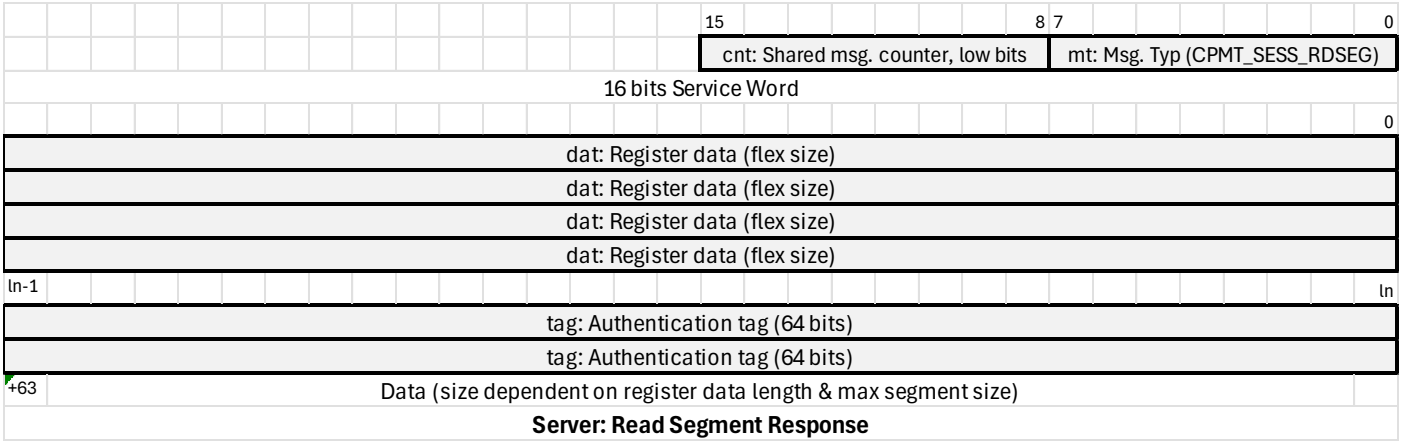
6.2.3 Client Read Segment Request

This request comes from the Configurator and is addressed to a Particpant.



6.2.4 Server Read Segment Response

This response comes from the Participant that received a request and it is addressed to the Configurator.



6.3 SPsec session Register Write Access

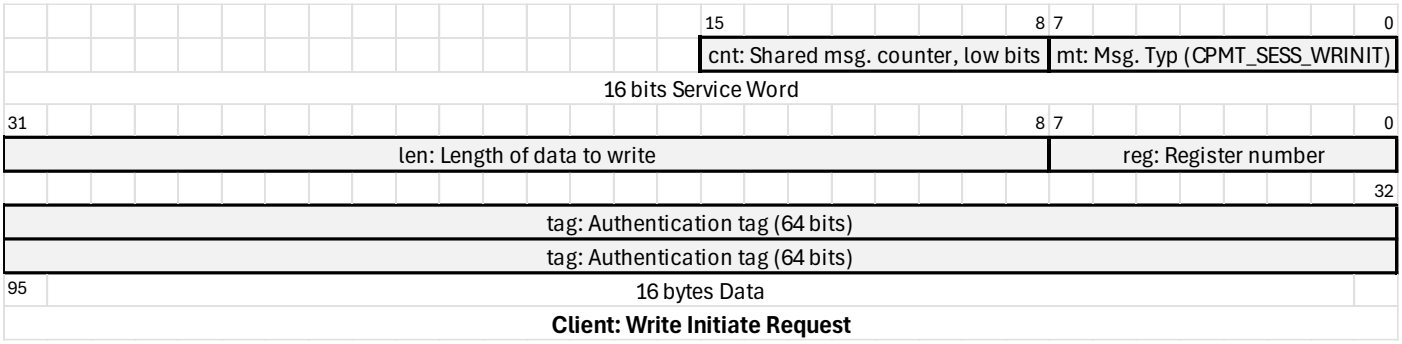
In this communication mode, the data field up to the authentication tag is encrypted.

The AEAD parameters for the calculation of the authentication tag are:

- key: the session key derived for this session
- nonce: the shared message counter as described in 2.10.
- plaintext (encrypt): the data field up to the authentication tag
- cyphertext (decrypt): the data field up to the authentication tag
- associated data: none
- tag (encrypt): generated and returned
- tag (decrypt): used to verify

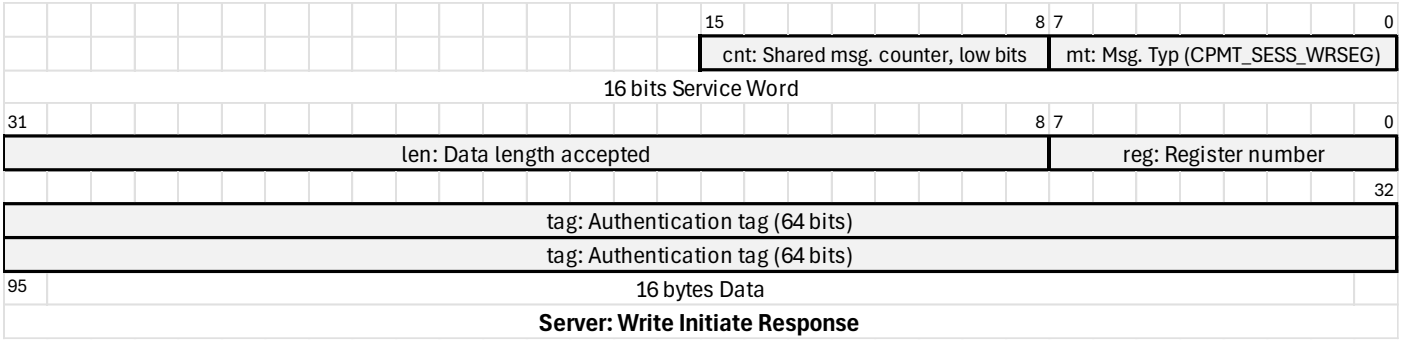
6.3.1 Client Write Initiate Request

This request comes from the Configurator and is addressed to a Particpant.



6.3.2 Server Write Initiate Response

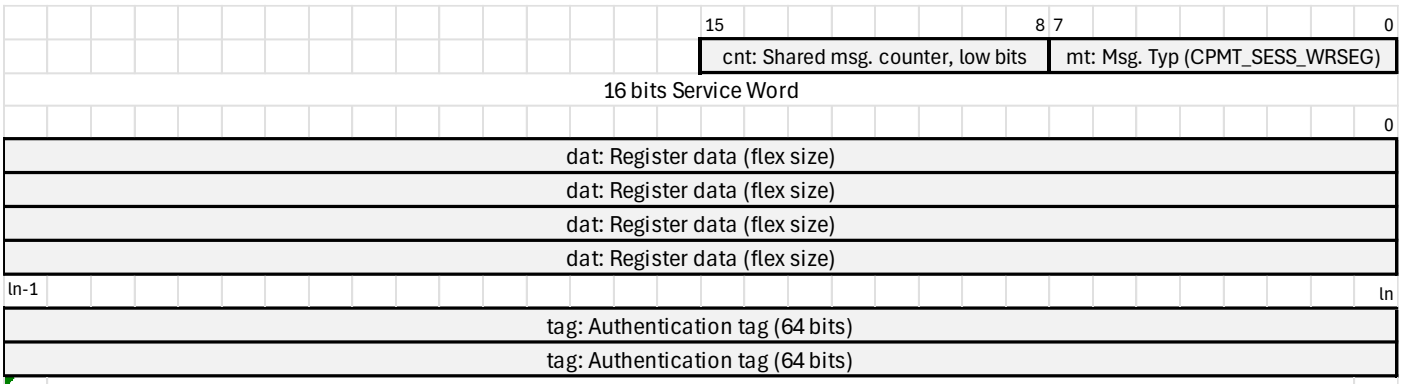
This response comes from the Participant that received a request and it is addressed to the Configurator.



Server: Write Initiate Response

6.3.3 Client Write Segment Request

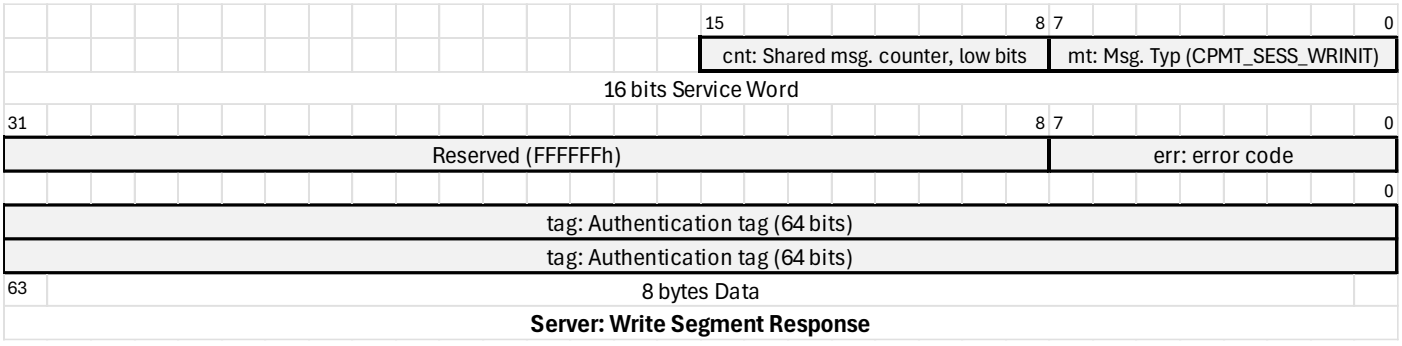
This request comes from the Configurator and is addressed to a Particpant.



Client: Write Segment Request

6.3.4 Server Write Segment Response

This response comes from the Participant that received a request and it is addressed to the Configurator.



Server: Write Segment Response

6.4 SPsec session Termination

In this communication mode, encryption is not used.

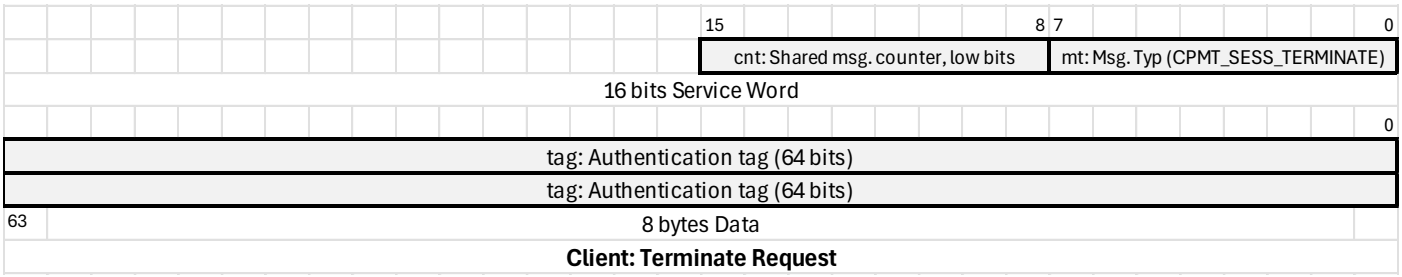
The AEAD parameters for the calculation of the authentication tag are:

- key: the session key derived for this session
- nonce: the shared message counter as described in 2.10.
- plaintext (encrypt): none
- cyphertext (decrypt): none
- associated data: none

- tag (encrypt): generated and returned
- tag (decrypt): used to verify

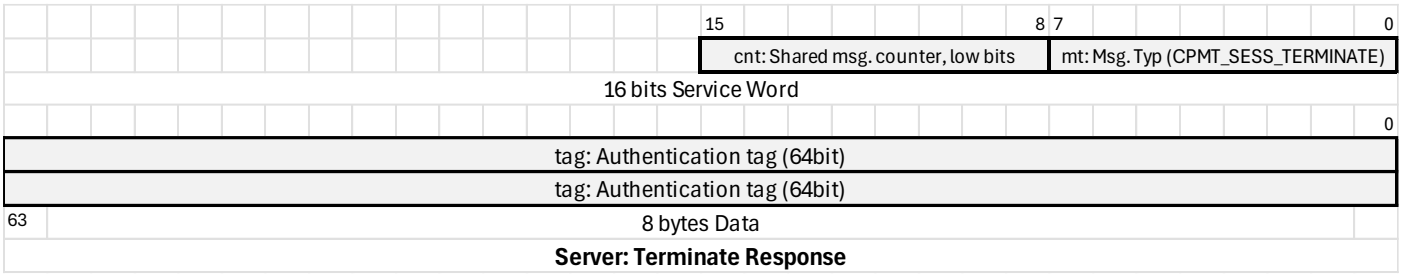
6.4.1 Client Terminate Request

This request comes from the Configurator and is addressed to a Particpant.



6.4.2 Server Terminate Response

This response comes from the Participant that received a request and it is addressed at the Configurator.



6.5 SPsec Authenticate Parameter

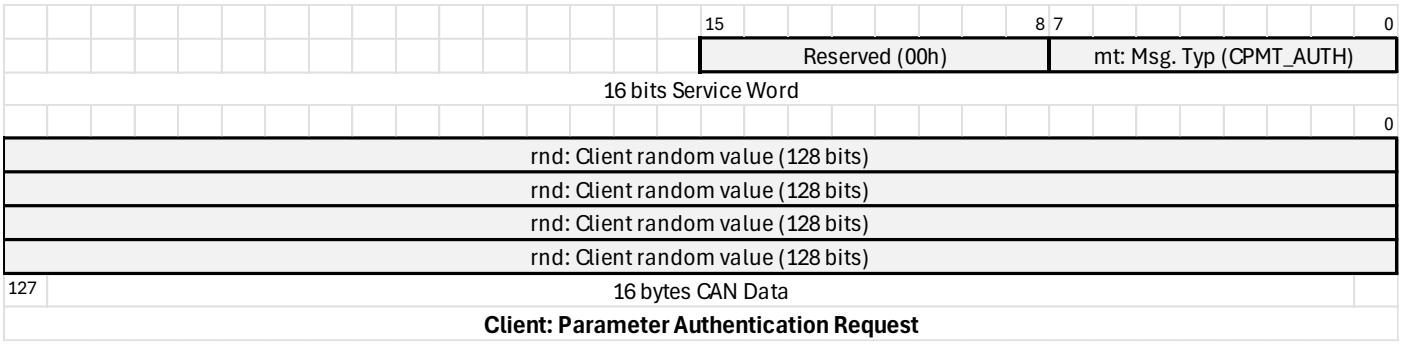
In this communication mode, encryption is not used.

6.5.1 Client Parameter Authentication Request

This request allows the client to read or write a single authenticated parameter without the need to establish a secure session.

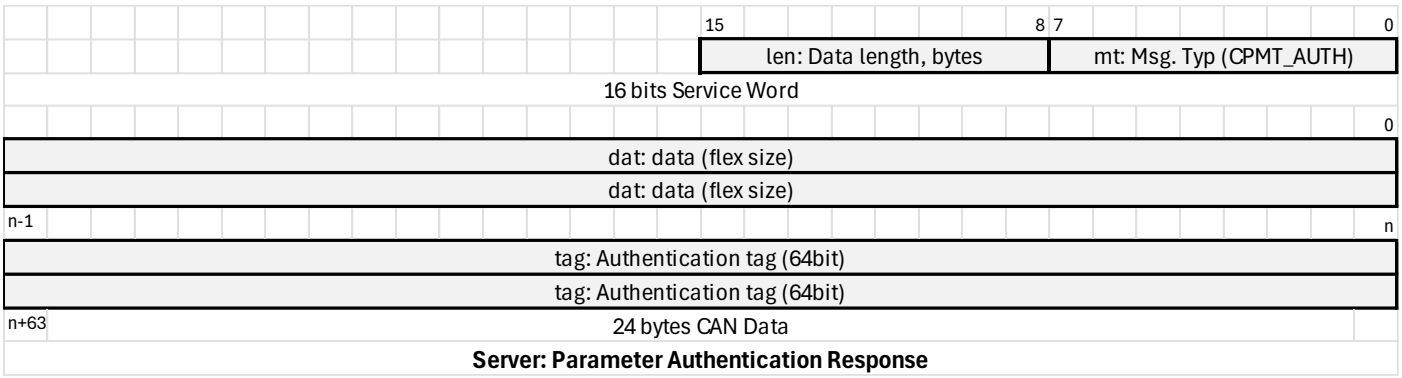
To initiate a parameter read access, the client first writes the parameter authentication request to the server's object and then reads the parameter authentication response from it.

To initiate a parameter write access, the client first reads the parameter authentication request from the server's object and then writes the parameter authentication response to it.



6.5.2 Server Parameter Authentication Response

This response from the server provides the single parameter including authentication.



Both the Client and Server derive the parameter authentication key based on the information from the request and the response. The use case specific KDF parameters for the key derivation are:

- input key: Seed Key
- salt: rnd value concatenated with pre-shared salt of Seed key
- salt size: 192 bits

The AEAD parameters for the calculation of the authentication tag of the response are:

- key: the parameter authentication key derived for this session
- nonce: array of zeros, length of nonce required by cryptographic function (zero nonce acceptable, as session key is only used once)
- plaintext (encrypt): none
- cyphertext (decrypt): none
- associated data: concatenation of
    - data field from the Client Parameter Authentication Request
    - data field from the Server Parameter Authentication Response (without tag)
- tag (encrypt): generated and returned
- tag (decrypt): used to verify

6.6 SPsec Sync Time

Not used with generic mapping.

6.7 SPsec Secure Heartbeat

Not used with generic mapping.

7 Optional Internal Control Plane Protocol

Not used with generic mapping.

8 Status and Event Indications and Handling

This section defines additional indications and recommended procedures not defined in the generic SPsec 201 document.

8.1 LED Security Status Indication

If the device provides LED indicators, it is recommended to also implement the SPsec security state indicators defined in the generic SPsec 201 document.

8.2 Security Status and Event Reporting

This section defines the 16 bits security event codes used in reporting events. On occurrence, they are written to the SPsec Last Security Event register (51h).

Default value

- NO_SEC_EVENT: 0x0000 – No security event

Group 5Exxh – Secure Session Events

- SESS_HELLO_KEY_NOT_FOUND: 0x5E01 – Hello, key requested not available
- SESS_FINISH_AUTH_FAILURE: 0x5E02 – Finished, authentication failure
- SESS_RESPONSE_TIMEOUT: 0x5E03 – Server response timeout
- SESS_TIMEOUT: 0x5E04 – Overall session timeout
- SESS_KEY_AUTH_FAILURE: 0x5E05 – Authentication failure during session

Group FExxh – Parameter Authentication Events

- SNC_REQ_AUTH_FAILURE: 0xFE00 – Authentication failure on parameter authentication
- SNC_REQ_TIMEOUT: 0xFE01 – Timeout waiting for parameter authentication response
- SNC_REFR_AUTH_FAILURE: 0xFE0E – Authentication failure of sync time service
- SNC_REFR_TIMEOUT: 0xFE0F – Timeout of sync time service

Group EExxh – Secure Data Plane Events

- SDP_AUTH_FAILURE: 0xEE00 – Authentication failure
- SDP_HB_LOSS: 0xEFxx – Participant heartbeat loss, xx indicates Participant ID of lost device

Optional, Group DExxh – Data Link Layer Events

- DLL_RX_OVERRUN: 0xDE01 – Receive buffer overrun
- DLL_TX_OVERRUN: 0xDE02 – Transmit buffer overrun

9 System Specific Implementation Notes

9.1 Generic Serial

Tbd.

9.2 CANopen and CANopen FD

In CANopen and CANopen FD all SPsec functionality is directly connected to object dictionary entries communicated by CANopen (U)SDO transfers. No further protocols, messages or CAN frames required.

SPsec registers may be mirrored to CANopen object dictionaries as required by the application. This supports the following scenarios:

- Selected SPsec registers can be read via (U)SDO without requiring a secure connection, like a public key ID.
- Selected object dictionary entries can be accessed securely through a SPsec secure session.

Using the authenticated parameter access, object dictionary entries supporting this access can be directly accessed securely, without requiring a SPsec secure session.

9.2.1 SPsec Secure Session

In CANopen and CANopen FD all communication between client and server uses SDO or USDO communication as defined by the CANopen specifications. For each client-server communication, the SDO/USDO server shall provide one SPsec Secure Session object dictionary entry of type DOMAIN. The CANopen access type is read/write. A single SPsec request-response transfer consists of the following steps:

- (U)SDO Client writes request to object SPsec Secure Session
  (depending on length can be expedited or segmented transfer)
- (U)SDO Client shall give the server processing time of at least 5 milliseconds before reading the response
- (U)SDO Client reads response from object SPsec Secure Session

If a CANopen node supports multiple (U)SDO clients, then it shall provide one separate SPsec Secure Session object for each client.

As (U)SDO transfers already handle segmentation, it is recommended to not duplicate segmentation handling in SPsec. In such an implementation all read and write transfers of the SPsec secure session are a combination of one read or write init transfer followed by a single data segment exchange.

9.2.2 Parameter Authentication

There can be multiple objects of type DOMAIN supporting the parameter authentication request. The CANopen access type is read/write.

As an example, one object could hold a copy of the 1018h device identification record, allowing a client to read the entire device authentication with one (U)SDO access, authenticated. The steps involved in this transfer are:

- (U)SDO Client writes parameter authentication request to the object containing the requested parameter.
- (U)SDO Client reads parameter authentication response from the object containing the requested parameter.

9.3 Modbus

Tbd.