SPsec302 – CAN FD Mapping

This document defines the CAN FD specific mapping of SPsec functions and methods, compatible with CANopen FD and J1939 FD.

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 1.40 of 8-JANUARY-2026, 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).

Contents

1 General Settings

This section summarizes the basic requirements to operate SPsec on CAN FD.

1.1 Physical Layer and CAN Address Identifiers

SPsec has no specific requirements towards the bitrates used. If CANopen FD or J1939 FD are used, it is recommended to use the settings as defined by the corresponding documents.

The CAN IDs used by the SPsec control plane are using 29 bit CAN IDs with bit 25 set. This ensures that there is no collision with any defined CANopen FD and J1939 FD messages. Therefore, by evaluating the CAN ID, it can easily be determined, if a CAN FD frame contains content for the control plane or data plane.

1.2 Payload sizes

The maximum payload size of data plane messages must be reduced to make room for the SPsec security information, the security stamp. The CANopen FD or J1939 FD implementation using SPsec functionality must support setting such a maximum payload.

The two biggest CAN FD frame sizes are 48 or 64 bytes. As the last 10 bytes of the CAN FD data field are used for the SPsec Security Stamp, the maximum payload size is 56 bytes. For ease of implementation, a maximum payload size of 48 bytes (or smaller) may be used.

See section 2.9 for the definition of the Security Stamp.

1.3 Timestamp Synchronization

In the CAN FD mapping of SPsec the sync value is a shared timestamp of 64 bits. See section 2.11 for its definition.

Upon start up, each Participant synchronizes its time using the service as defined in 5.2 and 6.5. After that, participants rely on the Sync message as defined in 6.6.

1.4 Use of Nonces

Cryptographic methods often require a nonce (number used once) of a specific or minimum size. Using a nonce with the same key and value again may already compromise security.

The nonce for the Security Stamp is built in the following way:

- The lowest bits of the nonce are filled with the current 64 bit timestamp.
- To ensure that the nonce differs, if the same timestamp is coincidentally used by two different participants, the next 16 bit are set to the CAN ID of the message to protect. This will be different for different participants.
- 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 Participant ID is a unique number assigned to each Participant used for addressing it. When and how these are assigned is application specific. The assignment can happen in several ways:

- Hard coded, if purpose and place in network is pre-known.
- Set by an individual 1:1 secure maintenance and configuration session, just before the node is connected to a network.
- Set dynamically on power up, based on dynamically assigned node IDs.

The Participant ID is unsigned, 7 bit, range is 1 to 127. If mapped to an 8 bit value, bit 7 indicates source (bit 7 = 0) or destination (bit 7 = 1) addressing.

By default, the following Participant IDs are recommended to be used as indicated.

- 01h: Time sync producer
- 7Eh: Default Participant ID to use, if none is set.
- 7Fh: Default Participant ID for diagnostic or configuration devices.

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 “302-[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

8 bits, unsigned, range of 1 to 127, write-only (activated on power cycle).

2.3.5.2 61h: Secure Heartbeat Timing

8 bits, unsigned, read-write. Optionally read-only if only one timing is supported.

The following values are defined:

0: Heartbeat disabled
1: 8s cycle time
2: 4s cycle time
3: 2s cycle time
4: 1s cycle time
5: 500ms cycle time
6: 250ms cycle time
80h-8Fh: Manufacturer specific cycle time
other: reserved

The default timeout for monitoring heartbeats is 2.5 times the cycle time. For example, 2.5s for a 1s cycle time.

2.3.5.3 62h: Secure Heartbeat Monitoring

32bits as array of 4 bytes (unsigned values of 8 bits), read-write. Optionally read-only if not configurable.

A value of "00h 00h 00h 00h" indicates that this device does not consume any heartbeats.

A value of "FFh FFh FFh FFh" indicates that this device does consume all heartbeats.

For any other value, each byte indicates the Participant ID whose heartbeat should be monitored. Up to 4 Participants ID can be defined. A byte of '00h' indicates no monitoring. Example: "04h 37h 00h 08h" defines that the heartbeats of the participants 4, 37h and 8 are monitored by this device.

2.3.5.4 63h: Sync Role Activation

8 bits, unsigned, read-write.

If present, '0' indicates that the sync role is switched off, a '1' indicates that it is switched on.

2.3.5.5 7Bh: CAN FD Bit Rate

16 bits, unsigned, write-only (activated on power cycle).

In the list below, (M) indicates a mandatory value that shall be support by all devices. All other list items are optional.

The lower 8 bits set the nominal bitrate as follows:

0: 1000kbps (M)
1: 800kbps
2: 500kbps (M)
3: 250kbps (M)
A0-FF: manufacturer specific

The upper 8 bits set the data bitrate as follows:

0: 1Mbps (M)
1: 2Mbps (M)

2: 4Mbps
3: 5Mbps (M)
4: 8Mbps
5: 10Mbps
A0-FF: manufacturer specific

2.3.5.6 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.

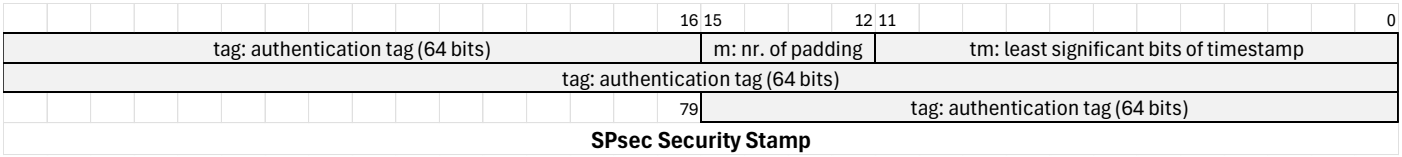
Data type: Unsigned, 8 bits.

2.9 Security Stamp

The Security Stamp is a record of 10 bytes (80 bits), added to the end of the CAN FD data field.

Bit 0 – 11,12 bits: least significant bits of timestamp
Bit 12 – 15, 4 bits: number of padding bytes inserted before
Bit 16 – 79, 64 bits: authentication tag

The maximum data size available per CAN FD frame is therefore 54 bytes. Depending on implementation this might be reduced to 48 bytes, the next smaller frame size available in CAN FD.



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 a concatenation of

- 32 bit CAN ID (bits 29 to 31 set to zero)
- 8 bit data field size
- Data field without padding and Security Stamp (only if encryption is not used)

If the CAN data before the padding and the Security Stamp is encrypted or in plaintext is a global configuration setting that all participants must be set to.

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

The CAN FD mapping uses a free running timestamp of 64 bit, unsigned. It is incrementing every 0.1 milliseconds.

The initial start value shall not be zero. Upon power up, the time sync role initializes the 64 bit timestamp with a random value.

Data type: Unsigned, 64 bit.

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
9: CPMT_SYNC
10: CPMT_HB
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

Key derivation is used to derive Communication and Session keys. The interface is HKDF compatible using one of the following methods:

- SHA256

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 (using 128 bits key size and 64 bits tag size)

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.

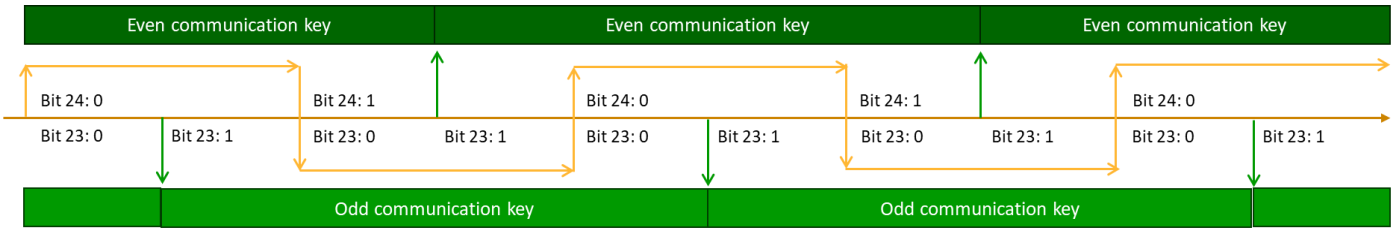
4.4 Seed Key

The seed key must be pre-shared among all SPsec participants in the same network. It is the base for Communication key derivations.

4.5 Odd and Even Communication Key

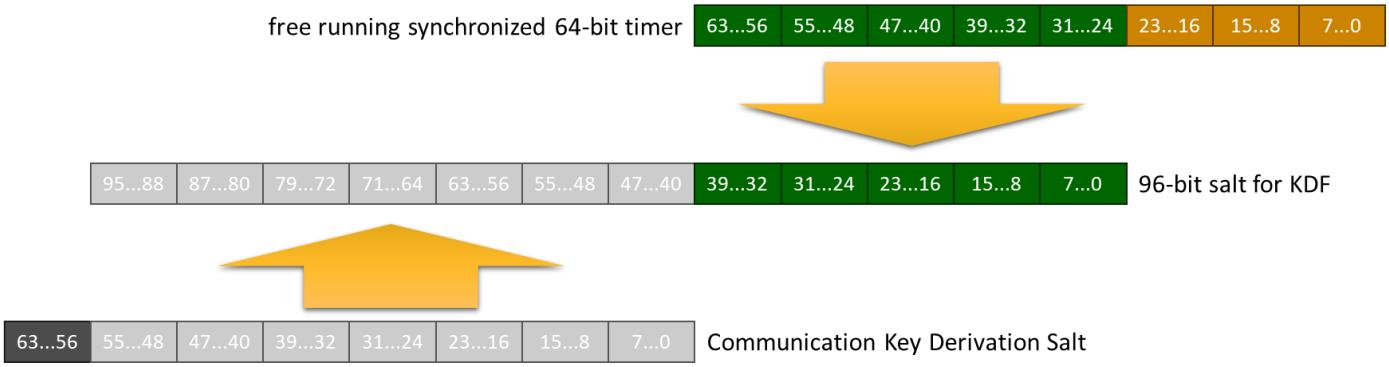
These keys are automatically derived on a fixed time basis using the synchronized 64 bit timestamp. These keys are derived with every 0 to 1 transition of bit 23 of the timestamp. If at that transition bit 24 is zero, then the odd key is generated, else the even key is generated.

For any instance using a communication key, bit 24 of the timestamp is the selector. If bit 24 is zero, the even key is used, else the odd key.



At any time, both even and odd key are valid

The KDF for generating the communication key uses the Seed key as input and generates the salt from the timestamp and the pre-shared salt as shown below.



Generation of salt value for the KDF input

Implementation note: When initializing the two keys for the first time based on a current timestamp, revert to the 2 previous values of bit 23 and 24 to derive the 2 “past” keys.

The specific use case KDF parameters for the key derivation are:

- input key: Seed Key
- salt: concatenation of 64 bit timer and Communication Key Derivation Salt
- salt size: 96 bits

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

On restart, participants require a synchronized timestamp. This is provided by the parameter authentication service as described in section 5.2. This service uses a one-time, one-use temporary key derived from the Seed key.

5 Cryptographic Functions and Modules

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

Nots on CAN FD frame information and data covered by cryptographic functions:

- Where used, encryption and decryption is only applied to the data field. If the data field includes a Security Stamp or authentication tag, these are exempt from encryption/decryption.
- Internal CAN meta information and control bits are never authenticated (like error passive bit).
- The CAN ID is included in the authentication.
- The CAN DLC (Data Length Code, does not directly show the real length of the data, allowed length values have gaps) is never authenticated to simplify implementations. 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 CAN FD, this is used by the Configurator for secure 1:1 sessions with Participants. The protocol definition is such, that the Configurator could establish multiple secure sessions with various Participants at the same time.

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 CAN FD, this is used by every Participant for the initial time value synchronization. When entering the Waiting state, each Participant sends the synchronization request to the Time Sync Role, which in return generates the matching response.

5.3 SPsec Multi-Participant Grouping

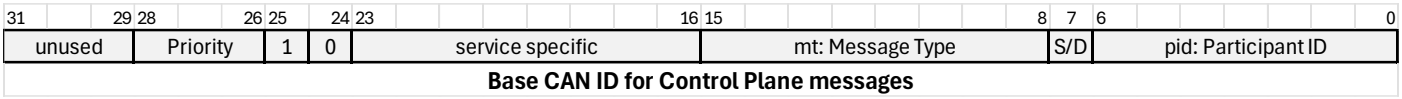
In CAN FD, once a Participant enters the Secure state, all its data plane communication is secured. Depending on configuration, only authenticated or both encrypted and authenticated.

5.4 SPsec Heartbeat

In CAN FD, once a Participant enters the Secure state, it fully participates in the production of its own heartbeat and the consumption of all other heartbeats.

6 External Control Plane Protocol

This section maps the messages of the control plane to CAN FD. Control plane messages use a 29 bits CAN ID as illustrated below.



This format supports both source and destination addressing. Bit 7 determines if the pid value is indicating the source (0) or destination (1) Participant.

The message type values are defined in 2.12.

The values for the message priority (lowest value for highest priority) are:

0: Sync time broadcast
1: Parameter authentication response
2: Parameter authentication request
3: Reserved
4: Secure heartbeat broadcast
5: Reserved
6: Internal control plane
7: SPsec secure session

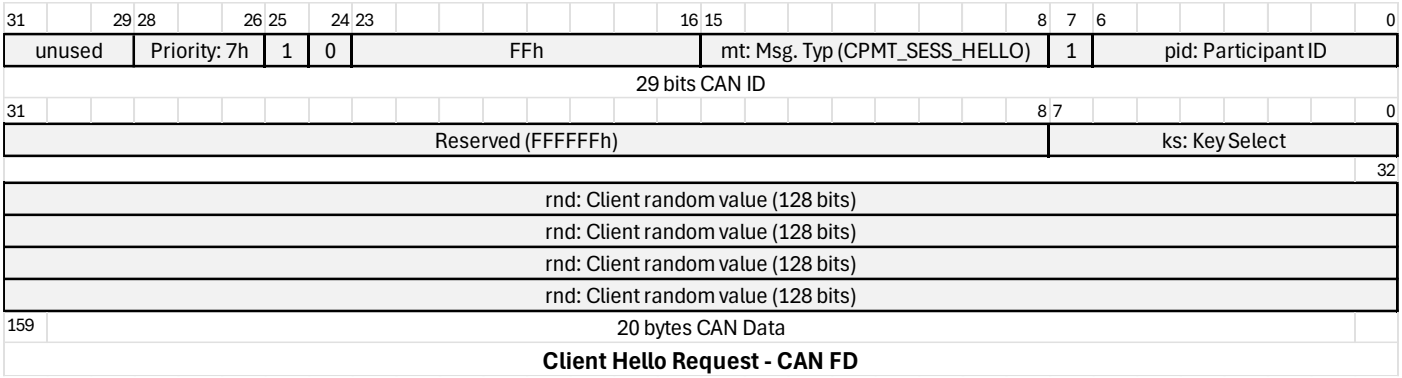
6.1 SPsec Session Start

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.



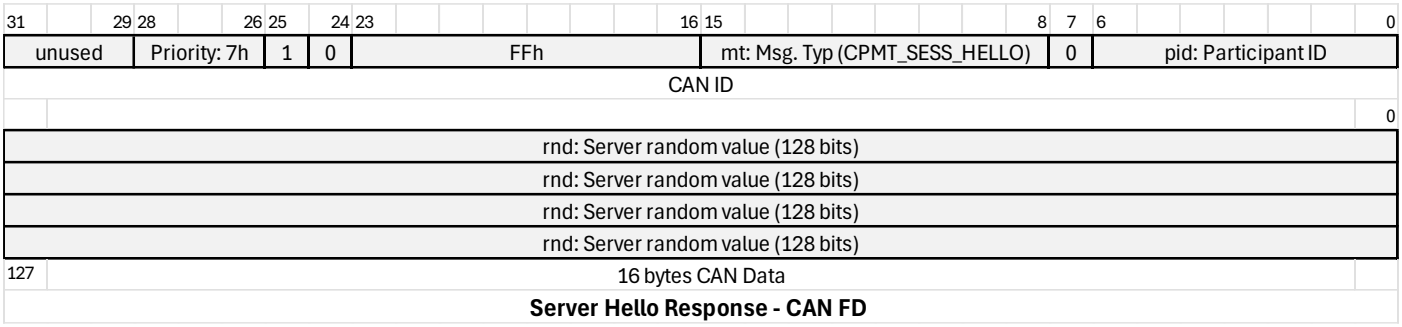
Client Hello Request - CAN FD

This request uses destination addressing, bit 7 is set to one. The pid value indicates the Participant receiving this request.

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



Server Hello Response - CAN FD

This response uses source addressing, bit 7 is set to zero. The pid value indicates the Participant sending this response.

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 Key, the Provisioning key, the Integrator Key or the 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.

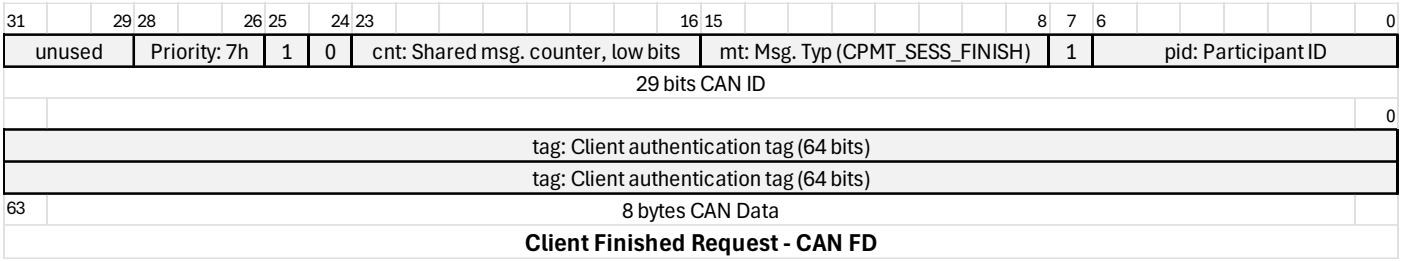
6.1.3 Client Finished Request

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
    - 32 bit CAN ID of the Client Finished Request (bits 29 to 31 set to zero)
- tag (encrypt): generated and returned
- tag (decrypt): used to verify



Client Finished Request - CAN FD

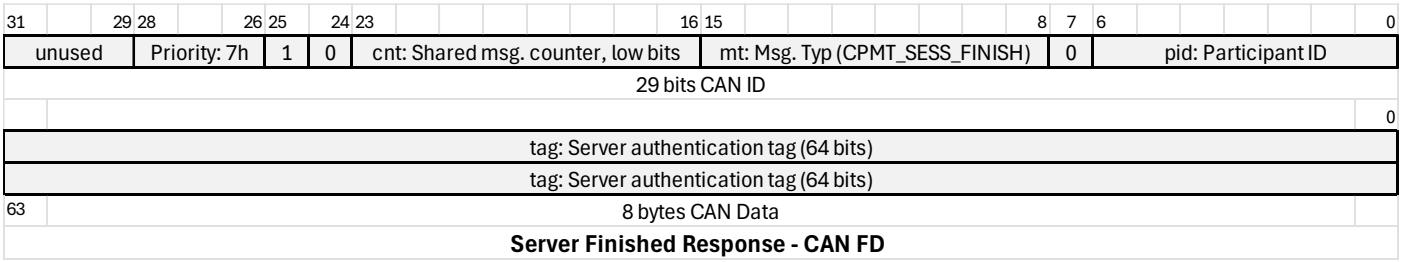
This request uses destination addressing, bit 7 is set to one. The pid value indicates the Participant receiving this request.

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
    - 32 bit CAN ID of the Server Finished Response (bits 29 to 31 set to zero)
- tag (encrypt): generated and returned
- tag (decrypt): used to verify



Server Finished Response - CAN FD

This response uses source addressing, bit 7 is set to zero. The pid value indicates the Participant sending this response.

6.2 SPsec session Register Read Access

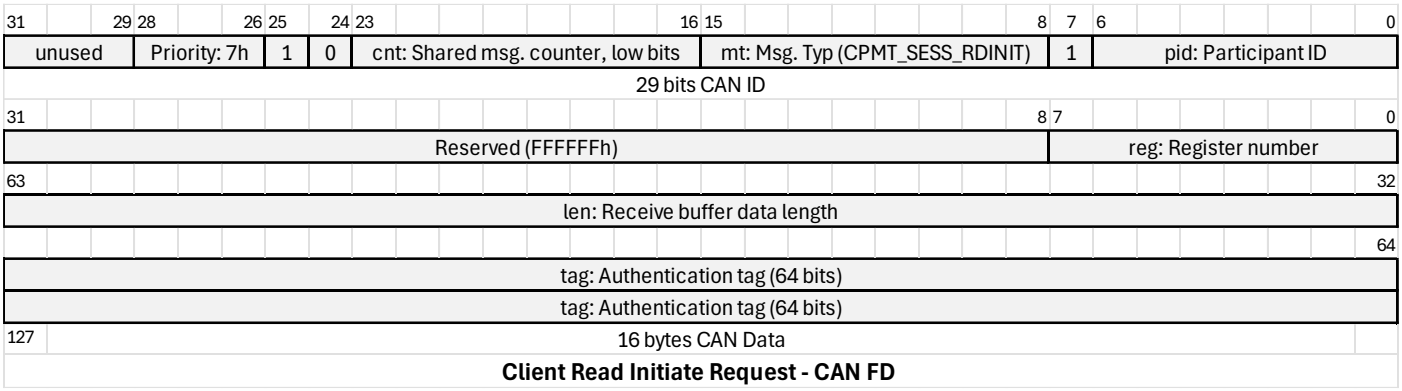
In this communication mode, the data field up to the authentication tag is encrypted.

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: 32 bit CAN ID (bits 29 to 31 set to zero)
- 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.

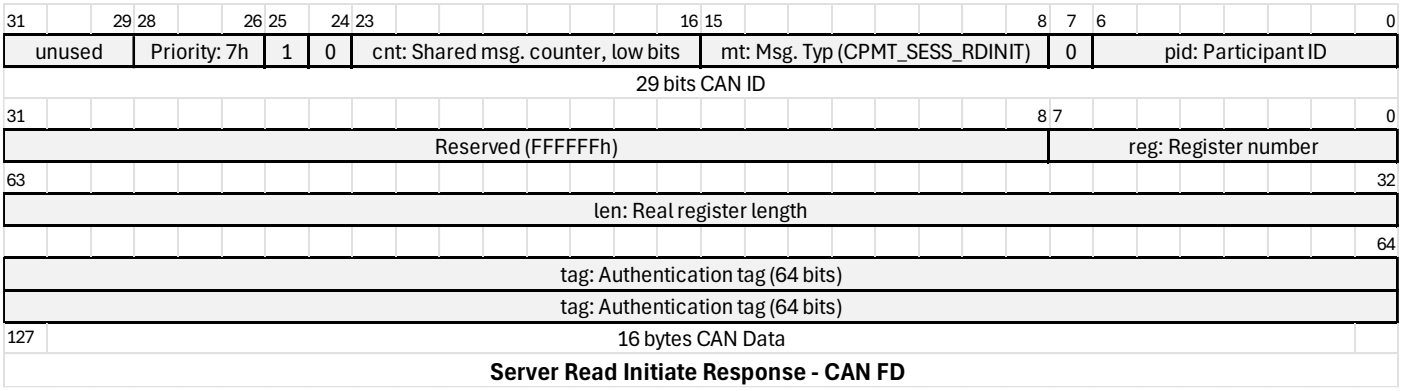


Client Read Initiate Request - CAN FD

This request uses destination addressing, bit 7 is set to one. The pid value indicates the Participant receiving this request.

6.2.2 Server Read Initiate Response

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

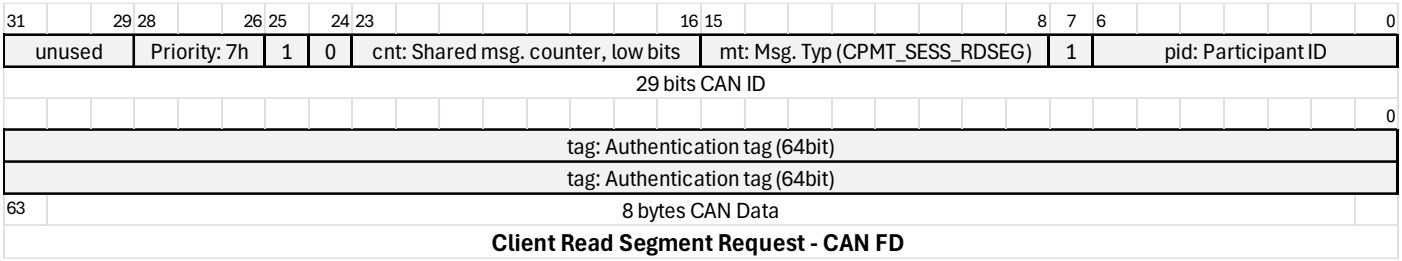


Server Read Initiate Response - CAN FD

This response uses source addressing, bit 7 is set to zero. The pid value indicates the Participant sending this response.

6.2.3 Client Read Segment Request

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

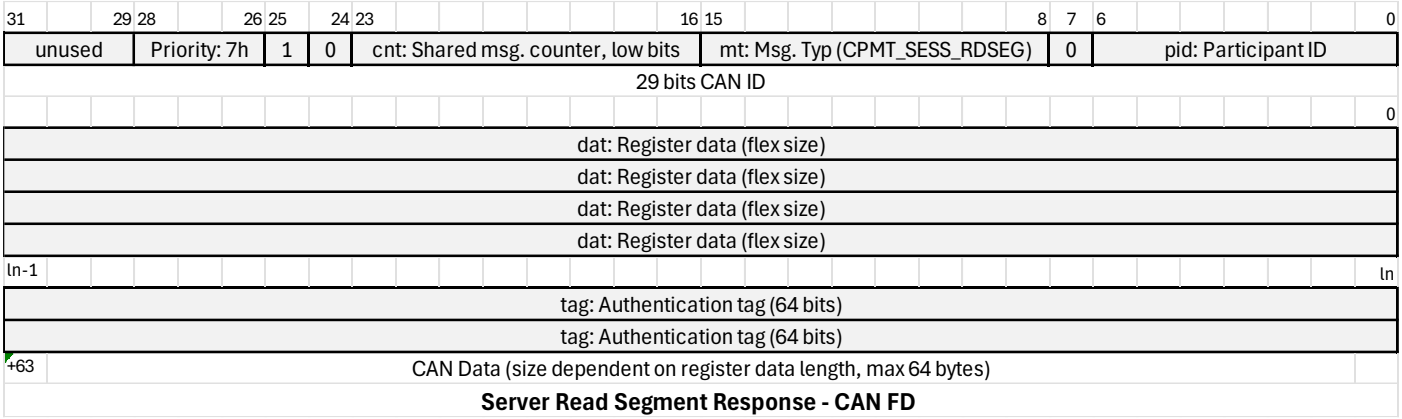


Client Read Segment Request - CAN FD

This request uses destination addressing, bit 7 is set to one. The pid value indicates the Participant receiving this request.

6.2.4 Server Read Segment Response

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



Server Read Segment Response - CAN FD

This response uses source addressing, bit 7 is set to zero. The pid value indicates the Participant sending this response.

If padding is required, the padding is added behind the tag, set to FFh and not taken into account for the AEAD.

6.3 SPsec session Register Write Access

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

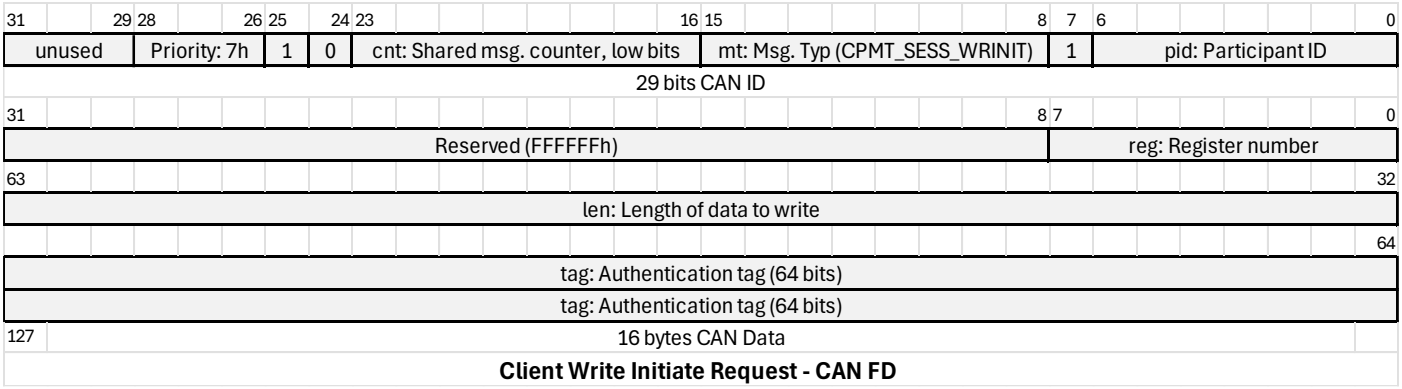
The AEAD parameters for the calculation of the authentication tag are:

- 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: 32 bit CAN ID (bits 29 to 31 set to zero)

- 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.

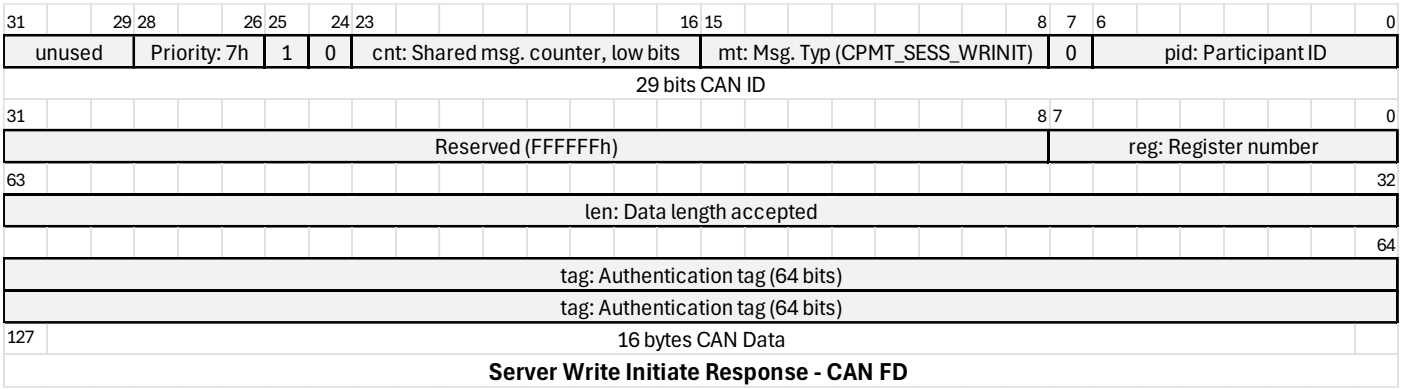


Client Write Initiate Request - CAN FD

This request uses destination addressing, bit 7 is set to one. The pid value indicates the Participant receiving this request.

6.3.2 Server Write Initiate Response

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

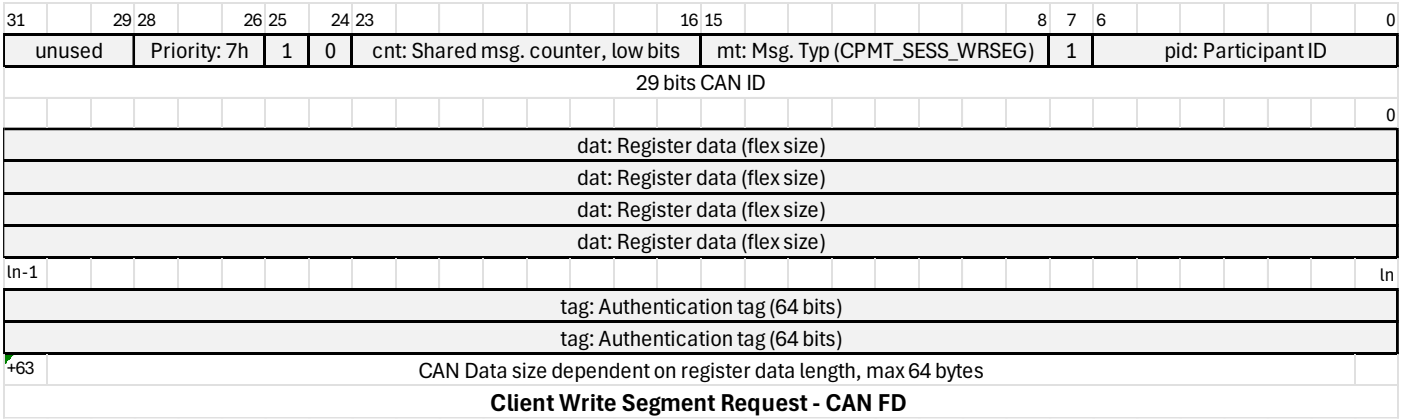


Server Write Initiate Response - CAN FD

This response uses source addressing, bit 7 is set to zero. The pid value indicates the Participant sending this response.

6.3.3 Client Write Segment Request

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



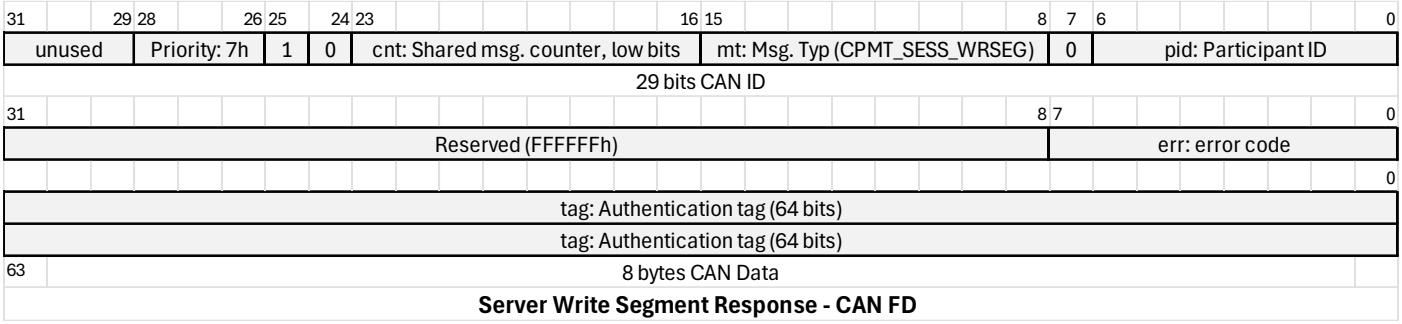
Client Write Segment Request - CAN FD

This request uses destination addressing, bit 7 is set to one. The pid value indicates the Participant receiving this request.

If padding is required, the padding is added behind the tag, set to FFh and not taken into account for the AEAD.

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 - CAN FD

This response uses source addressing, bit 7 is set to zero. The pid value indicates the Participant sending this response.

6.4 SPsec session Termination

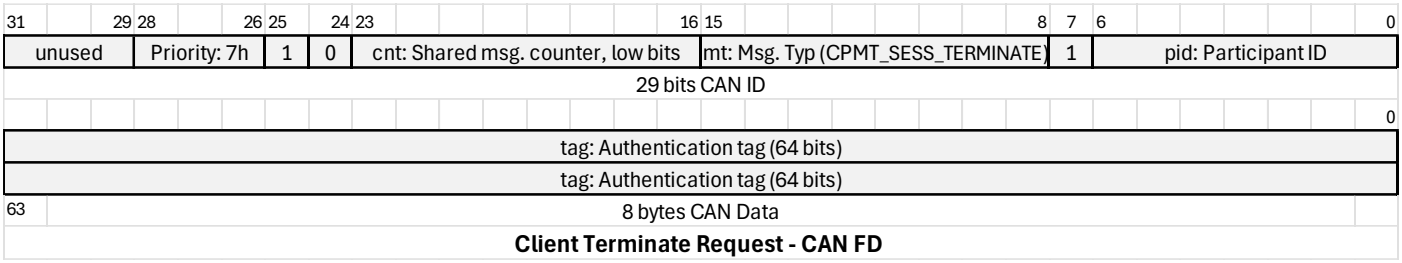
In this communication mode, encryption is not used.

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: 32 bit CAN ID (bits 29 to 31 set to zero)
- 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.

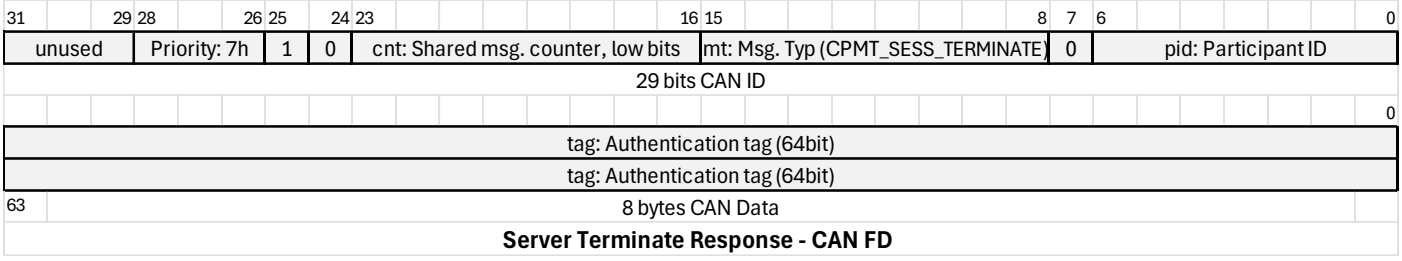


Client Terminate Request - CAN FD

This request uses destination addressing, bit 7 is set to one. The pid value indicates the Participant receiving this request.

6.4.2 Server Terminate Response

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



Server Terminate Response - CAN FD

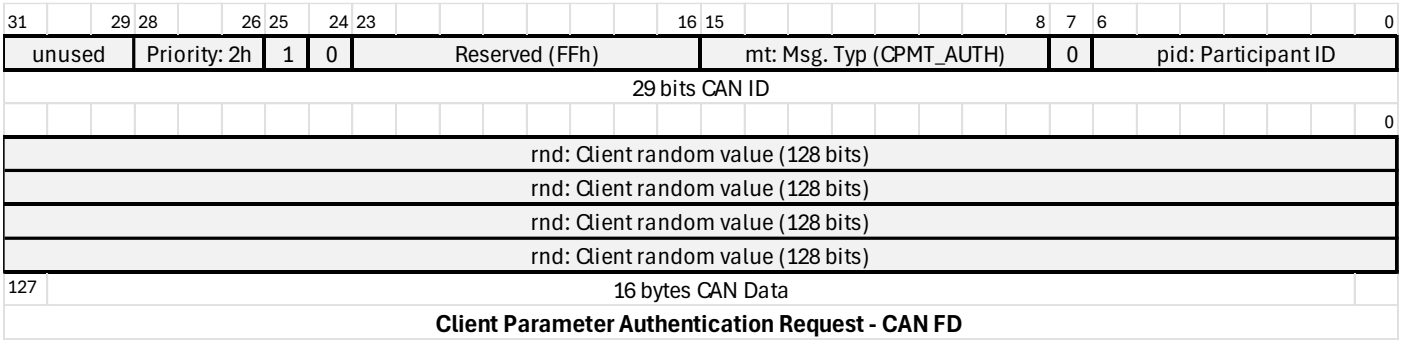
This response uses source addressing, bit 7 is set to zero. The pid value indicates the Participant sending this response.

6.5 SPsec Authenticate Parameter

In this communication mode, encryption is not used.

6.5.1 Client Parameter Authentication Request

This request comes from a Participant and is addressed to the Sync Timer Role.

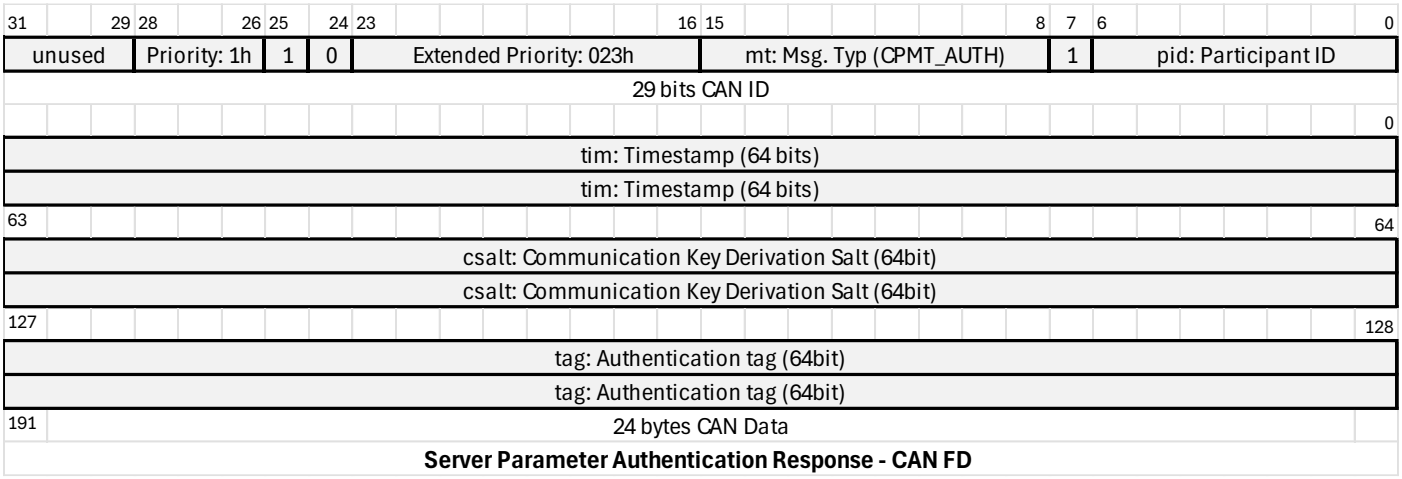


Client Parameter Authentication Request - CAN FD

This request uses source addressing, bit 7 is set to zero. The pid value indicates the Participant sending the request.

6.5.2 Server Parameter Authentication Response

This response comes from the Sync Timer Role and is addressed to the Participant that sent the request.



Server Parameter Authentication Response - CAN FD

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: concatenation of rnd, tim and csalt
- 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)
    - 32 bit CAN ID of the Server Parameter Authentication Response (bits 29 to 31 set to zero)
- tag (encrypt): generated and returned
- tag (decrypt): used to verify

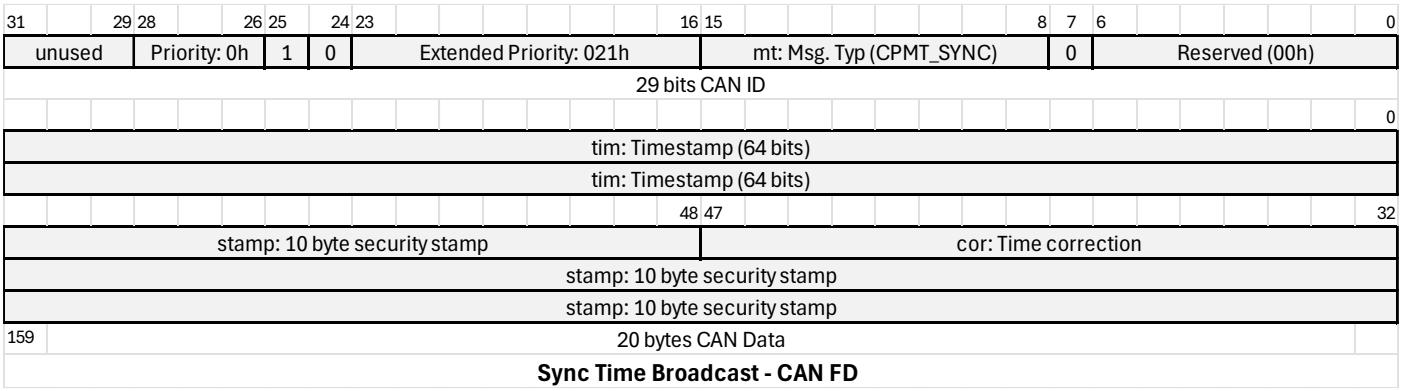
This response uses destination addressing, bit 7 is set to one. The pid value indicates the Participant receiving this response.

NOTE: The Base value uses the highest priority CAN ID value to ensure minimal delay for the timestamp data transmitted.

6.6 SPsec Sync Time

6.6.1 Sync Time Broadcast

This is a broadcast from the Sync Timer Role to all Participants.



Sync Time Broadcast - CAN FD

The time sync server uses a base address of 021h.

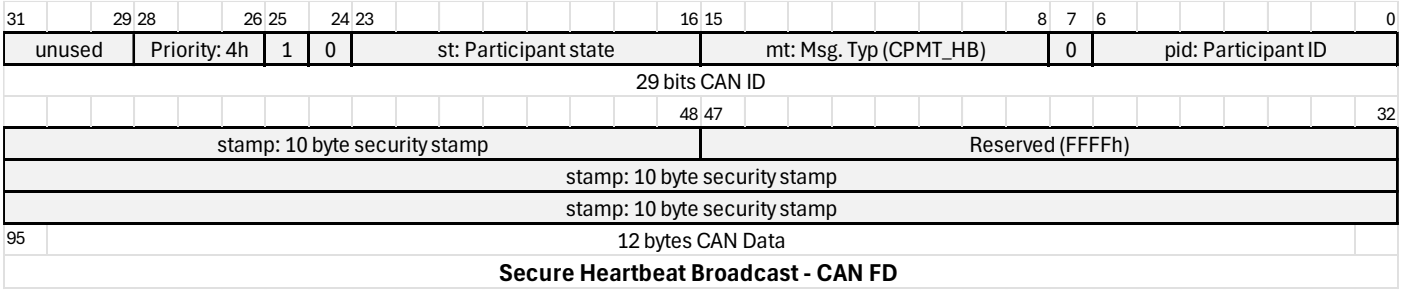
The time correction value is a signed 16bit value and defines a correction offset to the last Sync Time request transmitted by the Sync Role. This is optional, leave zero if not used.

NOTE: The Base value uses the highest priority CAN ID value to ensure minimal delay for the timestamp data transmitted.

6.7 SPsec Secure Heartbeat

6.7.1 Secure Heartbeat Broadcast

This is a broadcast transmitted by all Participants. All Participants monitor the heartbeats from all other Participants.



Secure Heartbeat Broadcast - CAN FD

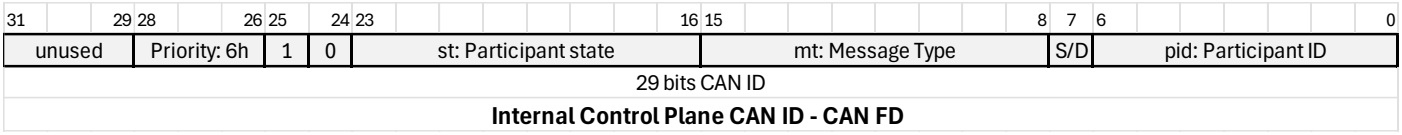
This broadcast uses source addressing, bit 7 is set to zero. The pid value indicates the Participant sending the broadcast.

NOTE: The Base value uses a medium priority CAN ID value to ensure the heartbeats do not use up valuable high-priority communication bandwidth.

7 Optional Internal Control Plane Protocol

The services of this protocol are optional. An application / host can fully operate without any control plane access. These services provide debug and maintenance access.

All transfers from Application / host to the Participant and from Participant to Application use the address / CAN ID as shown below.

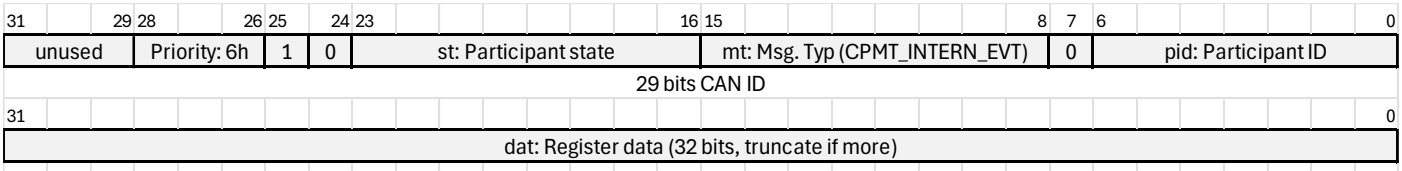


Internal Control Plane CAN ID - CAN FD

Bit 7 is set for messages from application to the SPsec sublayer and cleared for messages from the sublayer to the application.

7.1 Internal SPsec Event

This message reports an event to the host application.



Internal Security Event - CAN FD

The security event uses a fixed data length. If the register content is smaller than 32 bits, the value is copied to the lowest bits and upper bits are filled with zero. If the register content is greater than 32 bits, only the first 32 bits are reported.

Any changes to the following registers are reported:

- 42h: Integrator Key ID
- 43h: Seed Key ID
- 50h: SPsec Status
- 51h: SPsec Last Security Event

Whenever entering the “Waiting” state, the following registers are reported:

- 61h: Participant ID

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 – Time Sync 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
- DLL_ADRID_GUARD: 0xDE03 – Injection of own used Address ID detected
- DLL_DUP_FRAME_IGNORED: 0xDE04 – Ignored duplicated frame