Elektrobit

EB tresos® AutoCore Generic 8 Security Extensions documentation

product release 8.8.3

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

https://www.elektrobit.com/support

Legal disclaimer

Table of Contents

1. Overview of EB tresos AutoCore Generic 8 Security Extensions documentation

Welcome to the EB tresos AutoCore Generic 8 Security Extensions (ACG8 SECEXT) product documentation.

This document provides:

- Chapter 2, “Supported features“: list of features supported by the ACG8 SECEXT
- Chapter 3, “ACG8 SECEXT release notes“: release notes for the ACG8 SECEXT modules
- Chapter 4, “ACG8 SECEXT user guide“: background information and instructions
- Chapter 5, “ACG8 SECEXT module references“: information about configuration parameters and the application programming interface

2. Supported features

2.1. Supported IPsec features

The IPsec protocol suite provides authentication on the IP layer supporting the following main features:

- **Support of transport mode**: Support secure communication between two IP interfaces on two different ECUs.
- **Management of Security Associations**: Support configuration of settings (called Security Associations) for a specific local and remote IP address. Based on these settings the applicable authentication mechanisms are determined.
- **Support of security database policy**: Determine if received/transmitted data shall be secured, blocked or transmitted unsecured based on connection settings (e.g. protocol, port, target IP address). In case of blocking, diagnostic events shall be recorded. This also allows to implement packet filtering functionality.
- **Support of authentication header according to IETF RFC 4302**: Support pure authentication of IP frames without encryption. This reduces CPU load compared to encryption.
- **Support of extended sequence numbers according to IETF RFC 4304**: Support the use of extended (64-bit) sequence numbers. This option permits the transmission of very large volumes of data at high speeds over an IPsec Security Association, without re-keying to avoid sequence number space exhaustion.
- **Support of Internet Key Exchange protocol version 2 using pre-shared key (PSK)**: Support of IKEv2 using pre-configured keys.
- **Support of key exchange based on IKEv2 using certificates**: Exchange keys using the Internet Key Exchange protocol version 2 based on X.509v3 certificates and digital signatures (IETF RFC 7427).
- **Support of IKEv2 fragmentation according to IETF RFC 7383**: Allow the reception of more than one certificate without the need of using IP reassembly.
- **Support of regularly exchanging new keys**: Increase security by regularly exchanging new keys and seamless switching to these new keys.
- **API functions to manage key exchange via diagnostic commands**: API functions are provided to start key exchange, stop key exchange, lock the configuration, and get the status and error information that can be triggered via diagnostic messages.
- **Report diagnostic events**: Report diagnostic events if IP secured communication is not possible.
- **Sending of DELETE information message when shutting down communication**: Notify communication partners about the shutdown by sending a respective information message.
- **Dead peer detection**: Indicate a communication problem if no authenticated messages are received anymore.

2.2. Supported TLS features

The Transport Layer Security 1.2 is implemented according to IETF RFC 5246 and IETF RFC 4279 and supports the following cipher suites:

- TLS_PSK_WITH_NULL_SHA256
- TLS_PSK_WITH_AES_128_GCM_SHA256

2.3. Supported KeyM features

The KeyM module consists of two submodules: the crypto key submodule and the certificate submodule.

2.3.1. Supported crypto key submodule features

- **Key storage class `KEYM_STORAGE_IN_RAM`:**

  The crypto key submodule manages the RAM keys that it provides to the certificate submodule. This is done only internally. No API is supported.

- **Key storage class `KEYM_STORAGE_IN_NVM`:**

  The crypto key submodule manages the `NvM` keys that it provides to the certificate submodule. This is done only internally. No API is supported.

2.3.2. Supported certificate submodule features

- **Supported APIs:**
  - KeyM_Init()
  - KeyM_Deinit()
  - KeyM_GetVersionInfo()
  - KeyM_ServiceCertificate()
  - KeyM_SetCertificate()
  - KeyM_VerifyCertificate()
  - KeyM_GetCertificate()
  - KeyM_CertGetStatus()

- KeyM_CertElementGet()
- KeyM_CertElementGetFirst()
- KeyM_CertElementGetNext()

- **Key storage classes (`KeyMCryptoKeyStorage`):**
  - KEYM_STORAGE_IN_NVM: The certificate is stored to the `NvM`.
  - KEYM_STORAGE_IN_RAM: The certificate is stored to RAM.
- **X.509 certificate parser/verifier:**
  - Public key (`subjectPublicKey`): It is stored into the key element `CRYPTO_KE_SIGNATURE_KEY` (numerical key element ID: 1) of the `Csm` key of the corresponding `Csm` SIGNATUREVERIFY job (`KeyMCertCsmSignatureVerifyJobRef`), for use in verification.
  - Certificate validation step (`KeyMCertCertificateElementRuleRef`): During certificate parsing, the certificate is validated according to the configured rules. The comparison of parsed certificate elements is supported for integer arrays (`KeyMCertificateElementConditionArray`) and for integer constants (`KeyMCertificateElementConditionPrimitive`).
  - Signature (`signatureValue`): It is passed to the `Csm` SIGNATUREVERIFY job for verification.
  - Issuer/subject (`issuer and subject`): During certificate verification, when the certificate chain is checked, the issuer is compared to the subject of the upper certificate.

3. ACG8 SECEXT release notes

3.1. Overview

This chapter provides the ACG8 SECEXT product specific release notes. General release notes that are applicable to all products are provided in the EB tresos AutoCore Generic documentation. Refer to the general release notes in addition to the product release notes documented here.

3.2. Scope of the release

3.2.1. Configuration tool

Your release of EB tresos AutoCore is compatible with the release of the EB tresos Studio configuration tool:

- EB tresos Studio: 28.1.0 b210701-0227

3.2.2. AUTOSAR modules

The following table lists the AUTOSAR modules that are part of this ACG8 SECEXT release.

Module name	AUTOSAR version and revision	SWS version and revision	Module version	Supplier
KeyM	4.3.0 []	4.4.0 [0000]	1.2.11	Elektrobit Automotive GmbH
Tls	4.0.3 []	4.0.3 [0000]	1.0.8	Elektrobit Automotive GmbH

Table 3.1. Hardware-Independent Modules specified by the AUTOSAR standard

3.2.3. EB (Elektrobit) modules

The following table lists all modules which are part of this release but are not specified by the AUTOSAR standard. These modules include tooling developed by EB or they may hold files shared by all other modules.

Module name	Module version	Supplier
No EB modules available		

Table 3.2. Modules not specified by the AUTOSAR standard

3.2.4. MCAL modules and EB tresos AutoCore OS

For information about MCAL modules and OS, refer to the respective documentation, which is available as PDF at `$TRESOS_BASE/doc/3.0_EB_tresos_AutoCore_OS` and `$TRESOS_BASE/doc/5.0_MCAL_modules`[1]. It is also available in the online help in EB tresos Studio. Browse to the folders `EB tresos AutoCore OS` and `MCAL modules`.

3.3. Module release notes

3.3.1. KeyM module release notes

- AUTOSAR R4.3 Rev 0
- AUTOSAR SWS document version: 4.4.0
- Module version: 1.2.11.B439717
- Supplier: Elektrobit Automotive GmbH

3.3.1.1. Change log

This chapter lists the changes between different versions.

Module version 1.2.11
2021-04-30

- ASCKEYM-267 Fixed known issue: Placing the KeyM plugin in another directory than <tresos>/plugins is not possible

[1]`$TRESOS_BASE` is the location at which you installed EB tresos Studio.

Module version 1.2.7
2021-01-22

- Internal module improvement. This module version update does not affect module functionality.

Module version 1.2.6
2020-12-18

- Internal module improvement. This module version update does not affect module functionality.

Module version 1.2.5
2020-05-22

- Internal module improvement. This module version update does not affect module functionality.

Module version 1.2.4
2020-02-28

- Internal module improvement. This module version update does not affect module functionality.

Module version 1.2.3
2020-02-21

- Internal module improvement. This module version update does not affect module functionality.

Module version 1.2.2
2020-01-24

- Internal module improvement. This module version update does not affect module functionality.

Module version 1.2.1
2019-11-07

- Added improvements and fixes.

Module version 1.2.0
2019-10-25

- Improved X.509 certificate handling.

Module version 1.1.0
2019-09-20

- Module development snapshot of X.509 certificate handling.

3.3.1.2. New features

- No new features have been added since the last release.

3.3.1.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- KeyM

  Description:

  If KeyM_ServiceCerificate() is called with RequestDataLength equal to zero, the status of a certificate can be reset the same way as it is specified for KeyM_SetCertificate() [SWS_KeyM_00141].

- KeyM

  Description:

  If an internal call of Csm_KeyElementSet() fails, the additional Development error KEYM_E_-CSMKEYELEMENTSET_FAILED (0xE0) is set.

- KeyM

  Description:

  If a certificate handling function is being processed, any other API call will be rejected and will return KEYM_E_BUSY. This is done to ensure data consistency.

3.3.1.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- No DET error in KeyM_Init() and KeyM_GetVersionInfo()

  Description:

  The KeyM does not trigger the DET error KEYM_E_UNINIT in API functions KeyM_Init() and KeyM_-GetVersionInfo().

  Rationale:

  - SWS_KeyM_00144

    This requirement is not complete and does not consider that KeyM_Init has to be excluded.

  Requirements:

  SWS_KeyM_00144

- Return of KEYM_E_BUSY for 'Certificate submodule'

  Description:

  The fully supported KeyM 'Certificate submodule' APIs KeyM_ServiceCertificate, KeyM_SetCertificate, KeyM_GetCertificate, KeyM_VerifyCertificate and KeyM_CertGetStatus additionally return KEYM_E_-BUSY if KeyM is currently busy with other operations.

  Rationale:

  - In order to ensure data consistency, the KeyM module can execute only one API request at a time.

  Requirements:

  SWS_KeyM_00056 SWS_KeyM_00057 SWS_KeyM_00058 SWS_KeyM_00060 SWS_KeyM_00066

- Limited implementation of 'Crypto key submodule'

  Description:

  KeyM 'Crypto key submodule' APIs do not provide any functionality, except DET checks, and always return E_NOT_OK only.

  Rationale:

  - The support of the full feature set of the 'Crypto key submodule' is not in the scope of the current KeyM module. But providing related APIs and/or related configurable APIs and processing possible DET checks is needed by a KeyM's implementation in order to avoid missing references during linkage.

  Requirements:

  SWS_KeyM_00003, SWS_KeyM_00004, SWS_KeyM_00005, SWS_KeyM_00006, SWS_KeyM_00007, SWS_KeyM_00008, SWS_KeyM_00009, SWS_KeyM_00010, SWS_KeyM_00011, SWS_KeyM_00012, SWS_KeyM_00013, SWS_KeyM_00014, SWS_KeyM_00015, SWS_KeyM_00016, SWS_KeyM_00017, SWS_KeyM_00018, SWS_KeyM_00019, SWS_KeyM_00020, SWS_KeyM_00050, SWS_KeyM_00085, SWS_KeyM_00086, SWS_KeyM_00087, SWS_KeyM_00088, SWS_KeyM_00051, SWS_KeyM_00089, SWS_KeyM_00090, SWS_KeyM_00052, SWS_KeyM_00091, SWS_KeyM_00154, SWS_KeyM_00155, SWS_KeyM_00092, SWS_KeyM_00098, SWS_KeyM_00099, SWS_KeyM_00100, SWS_KeyM_00101, SWS_KeyM_00102, SWS_KeyM_00094, SWS_KeyM_00095, SWS_KeyM_00156, SWS_KeyM_00053, SWS_KeyM_00103, SWS_KeyM_00104, SWS_KeyM_00105, SWS_KeyM_00106, SWS_KeyM_00054, SWS_KeyM_00107, SWS_KeyM_00108, SWS_KeyM_00109, SWS_KeyM_00055, SWS_KeyM_00302

- No implementation of 'Crypto key submodule' expected interfaces

  Description:

  The KeyM 'Crypto key submodule' expected interfaces as specified by the KeyM SWS are not implemented.

  Rationale:

  - The support of the full feature set of the 'Crypto key submodule' is not in the scope of the current KeyM module.

  Requirements:

  SWS_KeyM_00067, SWS_KeyM_00068, SWS_KeyM_00069, SWS_KeyM_00097, SWS_KeyM_00096, SWS_KeyM_00093, SWS_KeyM_00070, SWS_KeyM_00071, SWS_KeyM_00077, SWS_KeyM_00150, SWS_KeyM_00079, SWS_KeyM_00080, SWS_KeyM_00081, SWS_KeyM_00151

- Limited implementation of 'Certificate submodule'

  Description:

  KeyM 'Certificate submodule' APIs KeyM_VerifyCertificates and KeyM_VerifyCertificateChain do not provide any functionality, except DET checks, and always return E_NOT_OK only.

  Rationale:

  - The support of the full feature set of the 'Certificate submodule' is not in the scope of the current KeyM module. But providing related APIs and/or related configurable APIs and processing possible DET checks is needed by a KeyM's implementation in order to avoid missing references during linkage.

  Requirements:

  SWS_KeyM_00059, SWS_KeyM_00118, SWS_KeyM_00119, SWS_KeyM_00123, SWS_KeyM_00139, SWS_KeyM_00061, SWS_KeyM_00124, SWS_KeyM_00125, SWS_KeyM_00126, SWS_KeyM_00028, SWS_KeyM_00153

- No implementation of 'Certificate submodule' expected interfaces

  Description:

  The KeyM 'Certificate submodule' key handler interface, KeyM_KH_ServiceCertificate, as specified by the KeyM SWS is not implemented.

  Rationale:

  - The support of the full feature set of the 'Certificate submodule' is not in the scope of the current KeyM module.

  Requirements:

  SWS_KeyM_00072

- Limited implementation of 'Certificate submodule' KeyM_ServiceCertificate() API

  Description:

  The KeyM 'Certificate submodule' KeyM_ServiceCertificate() API only provides the services KEYM_-SERVICE_CERT_SET_ROOT, KEYM_SERVICE_CERT_UPDATE_ROOT, KEYM_SERVICE_CERT_-SET_INTERMEDIATE and KEYM_SERVICE_CERT_UPDATE_INTERMEDIATE. Additionally 'Crypto key submodule' sessions are not considered.

  Rationale:

  - The support of the full feature set of the 'Certificate submodule' KeyM_ServiceCertificate() API is not in the scope of the current KeyM module.

  Requirements:

  SWS_KeyM_00113, SWS_KeyM_00114

- No implementation of RTE support

  Description:

  The KeyM RTE support as specified by the KeyM SWS is not implemented.

  Rationale:

  - RTE support is not in the scope of the current KeyM module.

  Requirements:

  SWS_KeyM_00082, SWS_KeyM_00159, SWS_KeyM_00083, SWS_KeyM_00084, SWS_KeyM_91005, SWS_KeyM_00160, SWS_KeyM_00161, SWS_KeyM_00162, SWS_KeyM_00163, SWS_KeyM_00164 SWS_KeyM_91000, SWS_KeyM_91001, SWS_KeyM_91002, SWS_KeyM_91004, SWS_KeyM_91012

- No implementation of post build support

  Description:

  The KeyM post build support as specified by the KeyM SWS is not implemented.

  Rationale:

  - Post build support is not in the scope of the current KeyM module.

Requirements:

ECUC_KeyM_00001

- KeyMCertificateManagerEnabled always enabled

  Description:

  The KeyM configuration parameter 'KeyMCertificateManagerEnabled' default value is changed to 'true' and not editable.

  Rationale:

  - The 'Certificate submodule' is the only scope of the current KeyM module.

  Requirements:

  ECUC_KeyM_00010

- KeyMCertCsmSignatureVerifyKeyRef not visible

  Description:

  The KeyM configuration parameter 'KeyMCertCsmSignatureVerifyKeyRef', that refers to the Csm key associated to the Csm signature verify job, is not visible and can therefore not be configured.

  Rationale:

  - The 'KeyMCertCsmSignatureVerifyKeyRef' is obsolete in the current KeyM module, as the Csm key reference in the job is used directly.

  Requirements:

  ECUC_KeyM_00031

- Custom steps for verification of KeyM certificates

  Description:

  At least, the following verification steps shall be successfully passed in this order for a successful verification of a certificate:

  - The certificates are verified from the top of the hierarchy to the bottom.
  - All certificates involved in the verification shall be available and have been parsed and verified successfully.
  - The subject field of the certificate in the upper hierarchy matches the issuer field of the certificate in the lower hierarchy.

- The signature can be verified with the associated public key of the certificate referenced by KeyMCertUpperHierarchicalCertRef. Signatures are verified by using the KeyMCertCsmSignatureVerifyJobRef of the certificate referenced by KeyMCertUpperHierarchicalCertRef.

  Rationale:

  - The verification steps of KeyM listed in SWS_KeyM_00029 are not supported completely by the current KeyM module.

  Requirements:

  SWS_KeyM_00029

- No implementation of pre-verification of certificates

  Description:

  The KeyM does not support pre-verification of certificates in the background functions KeyM_MainFunction or KeyM_MainBackgroudFunction triggered by KeyM_Init().

  Rationale:

  - A pre-verification would require a notification mechanism for all affected certificates when an upper certificate has been modified. As there is no reference to lower certificates, the search for the certificates that need to be re-verified is extensive. Also, runtime spent on a pre-verification that becomes outdated due to modifications to a certificate shall be avoided. Therefore, verification is only done when requested. And the certificates of the particular chain are locked against modification during verification.

  Requirements:

  SWS_KeyM_00045

- Rejected requirements

  Description:

  Some requirements are informational only, not in the scope of the implementation of KeyM or not implementable.

  Rationale:

  - SWS_KeyM_00025

    This requirement contradicts its following note and the API description of SetCertificate(). (https://jira.autosar.org/browse/AR-87704)

  - SWS_KeyM_00156

This is an integration requirement.

- SWS_KeyM_00158

  This is an integration requirement.

- SWS_KeyM_00033

  Certificate periodity check is not in the scope of the current KeyM module

- SWS_KeyM_00035

  Certificate revocation is not in the scope of the current KeyM module

- SWS_KeyM_00037

  https://jira.autosar.org/browse/AR-87705

- SWS_KeyM_00044

  There is no meaningful possibility to trigger KEYM_E_INIT_FAILED.

- SWS_KeyM_00046

  The support of the full feature set of the 'Crypto key submodule' is not in the scope of the current KeyM module.

- SWS_KeyM_00076

  This is not a requirement related to the KeyM, but to its environment.

- SWS_KeyM_00078

  This is not a requirement related to the KeyM, but to its environment.

- SWS_KeyM_00002

  In the scope of the X.509 certificate parser feature the KeyM 'Certificate submodule' is needed.

- SWS_KeyM_00023

  The public key of the issuer certificate to be used for signature verification will only be stored once, in the Csm key referenced in the Csm job referenced in the certificate to be verified.

- SWS_KeyM_00111

  In the scope of the current KeyM module a differentiation of KeyMCryptoKeyHandlerServiceCertificateEnabled is not necessary. It is always considered as FALSE.

- SWS_KeyM_00136

  In the current KeyM module, the certificate error status is always defined with sufficient details, so that the general KEYM_CERTIFICATE_INVALID status for any (unknown) internal error does not occur.

Requirements:

SWS_KeyM_00025, SWS_KeyM_00156, SWS_KeyM_00158, SWS_KeyM_00033, SWS_KeyM_00035, SWS_KeyM_00037, SWS_KeyM_00044, SWS_KeyM_00046, SWS_KeyM_00076, SWS_KeyM_00078, SWS_KeyM_00002, SWS_KeyM_00023, SWS_KeyM_00111, SWS_KeyM_00136

- Description:

  The KeyM uses different upper limits for KeyMCertificateElementConditionPrimitiveValue and KeyMCertificateElementConditionArrayElementValue.

  Rationale:

  - The specified value for the upper range limits is 18446744073709551615 and requires a 64 bit datatype. This is not supported on all platforms. Further, the endianness for any datatype bigger than 32 bit is not specified.

  Requirements:

  ECUC_KeyM_00053, ECUC_KeyM_00056

- Description:

  KeyM_CertElementGetNext returns a different error code if called for a certificate element that is not configured for iterations.

  Rationale:

  - The specified return value KEYM_E_CERT_INVALID_FORMAT does not exist.

  Requirement:

  SWS_KeyM_00148

3.3.1.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- KeyM

  Description:

  The KeyM module provides support of the key management submodule only to the extent as needed for realizing the certificate submodule, i.e. for managing the storage of a certificate. This means in particular that no key operation is available. All unsupported configuration options are not visible.

- KeyM

  Description:

  The KeyM certificate submodule only supports X.509 v3 certificates.

- KeyM

  Description:

  The API function KeyM_ServiceCertificate only supports the services:

  - KEYM_SERVICE_CERT_SET_ROOT
  - KEYM_SERVICE_CERT_UPDATE_ROOT
  - KEYM_SERVICE_CERT_SET_INTERMEDIATE
  - KEYM_SERVICE_CERT_UPDATE_INTERMEDIATE

- KeyM

  Description:

  The KeyM certificate submodule for X.509 v3 certificates only supports string types:

  - utf8String
  - printableString
  - IA5String

  Rationale:

  These are the most commonly used string types.

- KeyM

  Description:

  The KeyM certificate submodule for X.509 v3 certificates only supports the type 'AlgorithmIdentifier' with an empty component 'parameters' (TLV = 0x05 0x00).

- KeyM

  Description:

  The KeyM crypto key submodule does not support KeyM CryptoKey storage location (KeyMCryptoKeyStorage) KEYM_STORAGE_IN_CSM. Supported storage locations are KEYM_STORAGE_IN_RAM and KEYM_STORAGE_IN_NVM.

  Rationale:

In the KeyM certificate submodule for X.509 v3 certificates, only whole certificates are stored to KeyM crypto keys. There is no usecase for the storage location KEYM_STORAGE_IN_CSM.

- KeyM

  Description:

  The KeyM certificate submodule does not support time stamp check. Thus, there is no check of a certificate's validity, i.e. "not before" and "not after" time periods. There is no interconnection to a time server.

- KeyM

  Description:

  The KeyM certificate submodule supports the validation step of the parsing, i.e. when checking rules configured in KeyMCertificateElementVerification, only for the comparison of KeyElement compare conditions (KeyMCertificateElementCondition) of type KeyMCertificateElementConditionPrimitive (primitive compare value) and KeyMCertificateElementConditionArray (array compare value).

- KeyM

  Description:

  The KeyM certificate submodule supports up to 32 certificate extensions.

  Rationale:

  In order to check for dublicated extensions, i.e. extensions with the same OID, the given extensions need to be stored. The KeyM reserves memory to store a maximum of 32 pointers to parsed certificate extensions.

- KeyM

  Description:

  The KeyM certificate submodule, when calling the configured Csm signature verification job in order to verify a certificate, only supports operation mode CRYPTO_OPERATIONMODE_SINGLECALL.

  Rationale:

  Other operation modes would require several calls to the Csm job.

- KeyM

  Description:

  The KeyM certificate submodule for X.509 v3 certificates supports (i) parsing including check for valid format and configured validations, and (ii) verifying signatures in a certificate chain. It does not provide a full validation check of all conditions specified in RFC5280. These are the RFC5280 validations that are performed additionally to the user configured validations and conditions:

  - The certificate's version is v3.
  - The signature algorithm field contains the same algorithm as the signature field in the sequence tb-sCertificate.
  - The issuer field contains a non-empty distinguished name (DN).
  - The subject field contains a non-empty distinguished name (DN).
  - All extensions marked as critical are on the list of identified extensions and can be processed.
  - A certificate does not include more than one instance of a particular extension.
  - For all not self-signed certificates, the AuthorityKeyIdentifier extension exists and can be parsed.
  - AuthorityKeyIdentifier extension is not marked as critical.
  - For all not self-signed certificates, the SubjectKeyIdentifier extension exists and can be parsed.
- KeyM

  Description:

  The KeyM certificate submodule for X.509 v3 certificates supports the identification of the following certificate extensions as specified in RFC5280:

  - AuthorityKeyIdentifier
  - SubjectKeyIdentifier
  - BasicConstraints
  - KeyUsage
  - ExtendedKeyUsage

  For unknown extensions, we support parsing of a single element and of the subelements of a single SEQUENCE. Other values are ignored and do not raise an error. When an unknown extension which is set as critical is found during parsing, the parsing will fail and an error will be reported as specified in RFC5280.

3.3.1.6. Open-source software

`KeyM` does not use open-source software.

3.3.2. Tls module release notes

- Module version: 1.0.8.B439717
- Supplier: Elektrobit Automotive GmbH

3.3.2.1. Change log

This chapter lists the changes between different versions.

Module version 1.0.8
2021-06-25

- Support for HSM key derivation.

Module version 1.0.7
2021-06-08

- Support for certificate on server side using TLS_ECDHE_ECDSA_WITH_NULL_SHA.
- Support for TcpIpTlsUseSecurityExtensionRecordSizeLimit.
- Support for asynchronous signature generation/verification.
- Support for certificate on server side using TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256.
- Asynchronous key exchange functionality.
- ASCTLS-112 Fixed known issue: TCP receive window continuously decreases.
- ASCTLS-65 Fixed known issue: Re-establishing a previously closed TLS/DTLS connection is not possible.

Module version 1.0.6
2021-03-22

- ASCTLS-70 Fixed known issue: TLS connection cannot be established when using MAC authentication.

Module version 1.0.5
2021-03-05

- ASCTLS-29 Fixed known issue: Application data message is discarded by the upper layer when Tls is configured as client over TCP.
- ASCTLS-53 Fixed known issue: Wrong event sent to SoAd when closing a DTLS UDP connection.
- ASCTLS-60 Fixed known issue: TLS/DTLS client stops the handshake if no ServerKeyExchange message is received.

Module version 1.0.4
2021-02-12

- Report an event when TLS extensions are used during connection establishment.
- Return PHY_ADDR_MISS to SoAd when the first UDP datagram using Tls is sent.
- Retry sending ClientHello when the physical address of the remote host is not known.

Module version 1.0.3
2020-10-23

- Allow receiving HelloVerifyRequest message with DTLS version lower than 1.2.
- ASCTLS-20 Fixed known issue: No DTLS message exchange possible.

Module version 1.0.2
2020-06-19

- Internal module improvement. This module version update does not affect module functionality.

Module version 1.0.1
2020-02-21

- Internal module improvement. This module version update does not affect module functionality.

Module version 1.0.0
2018-08-01

- Initial development version (limited feature set).

3.3.2.2. New features

- Added the support to derive the session keys and perform cryptographic operations from the HSM firmware for ciphersuites TLS_PSK_WITH_NULL_SHA256 and TLS_PSK_WITH_AES_128_GCM_SHA256.
- Added the support to send and receive the recordSizeLimit extension.
- Added the support for ciphersuite using certificates (TLS_ECDHE_ECDSA_WITH_NULL_SHA and TLS_-ECDHE_ECDSA_WITH_AES_128_GCM_SHA256).

3.3.2.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.2.4. Deviations

This module is not part of the AUTOSAR specification.

3.3.2.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- To be defined.

3.3.2.6. Open-source software

Tls does not use open-source software.

4. ACG8 SECEXT user guide

4.1. Overview

This user guide describes the concepts and the configuration of the modules:

- TCP/IP (`TcpIp`)
- Key Manager (`KeyM`)

This user guide is intended for readers who have good knowledge of AUTOSAR and about the purpose of the EB tresos AutoCore Generic 8 Security Extensions modules.

4.2. Background information

This chapter describes the basic concepts of the EB tresos AutoCore Generic 8 Security Extensions.

Additional background information is available in the module-specific user guides:

- Section 4.3, “IPsec user guide” for the `TcpIp` module
- Section 4.4, “KeyM module user guide” for the `KeyM` module

4.2.1. Purpose of the ACG8 SECEXT

ACG8 SECEXT provides extensions to fulfill common security requirements. It provides support for Transport Layer Security (TLS) to encrypt and authenticate layer 4 communication (TCP, UDP), and Internet Protocol security (IPsec) to authenticate layer 3 communication (IP).

ACG8 SECEXT further provides support for a vehicle key and certificate management system.

To be able to use the security functionality provided by the ACG8 SECEXT product within an AUTOSAR environment, the `Csm`, `CryIf`, and `Crypto Driver` modules are required.

4.3. IPsec user guide

4.3.1. Overview

This user guide provides information about the IPsec concept in the AUTOSAR context. Before you read this user guide, read the general concept of communication stacks in AUTOSAR that are described in the EB tresos AutoCore Generic documentation.

- Section 4.3.2, "Background information" describes the basic concept of the IPsec and the interplay with the IKEv2 part.

4.3.2. Background information

IPsec provides interoperable, high quality, cryptographically-based security for IPv4 and IPv6. This includes the following services:

- access control
- connectionless integrity
- data origin authentication
- detection and rejection of replays (a form of partial sequence integrity)
- confidentiality (via encryption)
- limited traffic flow confidentiality

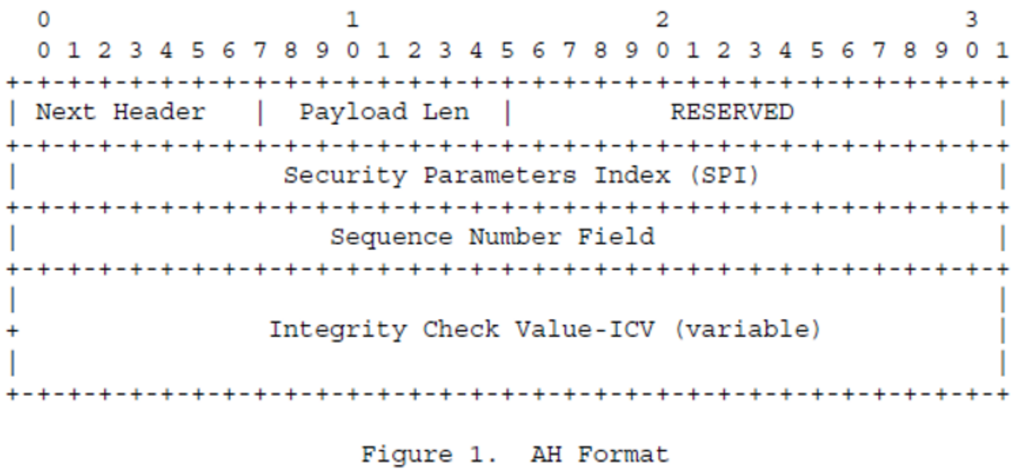
These services are provided at the IP layer, offering protection in a standard fashion for all protocols that may be carried over IP. IPsec ensures minimal firewall functionality. This functionality uses the cryptographically-enforced authentication and integrity provided for all IPsec traffic (access control).

In general, IPsec supports two distinct protocols: the Authentication Header (AH) protocol and the Encapsulating Security Payload (ESP) protocol and two modes of operation: transport mode and tunnel mode. This IPsec implementation uses the AH protocol in transport mode only. The AH protocol provides authenticity, the integrity of the datagram, and protection against replay attacks, but no confidentiality. Transport mode means that the AH header is added immediately after the IP header but in contrast to tunnel mode no new IP header is added.

4.3.2.1. Authentication Header (AH) protocol

The AH protocol (specified in IETF RFC 4302) is a security protocol for IPsec and ensures data integrity and data origin authentication for IP datagrams. Figure 4.1, "Authentication Header format (Source: IETF RFC

4302)" shows the AH format. Furthermore, it protects against replay attacks. It also allows the optional use of an ESN (Extended Sequence Number).



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Payload Len  |          RESERVED             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Security Parameters Index (SPI)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Sequence Number Field                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                Integrity Check Value-ICV (variable)           |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                    Figure 1.  AH Format
```

Figure 4.1. Authentication Header format (Source: IETF RFC 4302)

The AH protocol can be used in transport mode or tunnel mode. The policy (security strategy) is maintained in the Security Policy (SP). Therefore, the SP is a connection-specific selector whether and with which protocol or mode IPsec is to be used. The individual security policies are managed on the systems in the Security Policy Database (SPD).

The Security Association (SA) is a particular instance of a security policy that applies to a SA-defined connection for a limited time (with IKE) or an infinite time (static SAs). That means an SA can be generated statically or automatically (with IKEv2). The SA contains a list of the agreed cryptographic procedures with the valid keys. Since there can be multiple SA IPsec connections simultaneously, the different SAs must be managed somewhere.

This happens in the Security Association Database (SAD). Incoming and outgoing packets carry the index of the SA that applies to them, which refers to the respective entry in the SAD. This pointer is called the Security Parameter Index (SPI).

Generated security information is carried in each IP datagram via the AH, which is added immediately after the IP header. The integrity and authenticity mechanisms are generated via a message authentication code: e.-g. HMAC - Hash Message Authentication Code. The HMAC is formed from the data of the IP header, the AH header, and the actual payload of the packet. The AH header also contains a sequence number counter, an authentication field with the HMAC, and the Security Parameter Index (SPI).

This SPI is the indicator to determine which SA should be used. If IP fragmentation is needed, this is done after the calculation of the authentication data. However, the algorithms that should be used are not part of the standard. The AH can be used in transport and tunnel mode (not supported). In transport mode, the AH is inserted after the IP header and before the next layer protocol. In an IPv6 context, AH is used as an end-to-end payload. Therefore it should be inserted after hop-by-hop, routing, and fragmentation extension headers.

The reception (Rx) behavior slightly differs from the transmission (Tx) side. Incoming data is checked for validity (e.g. length field does not exceed payload length) before being processed. Figure 4.2, "Rx check process" shows how an inbound packet is processed in the implementation to find the correct policy and SA.

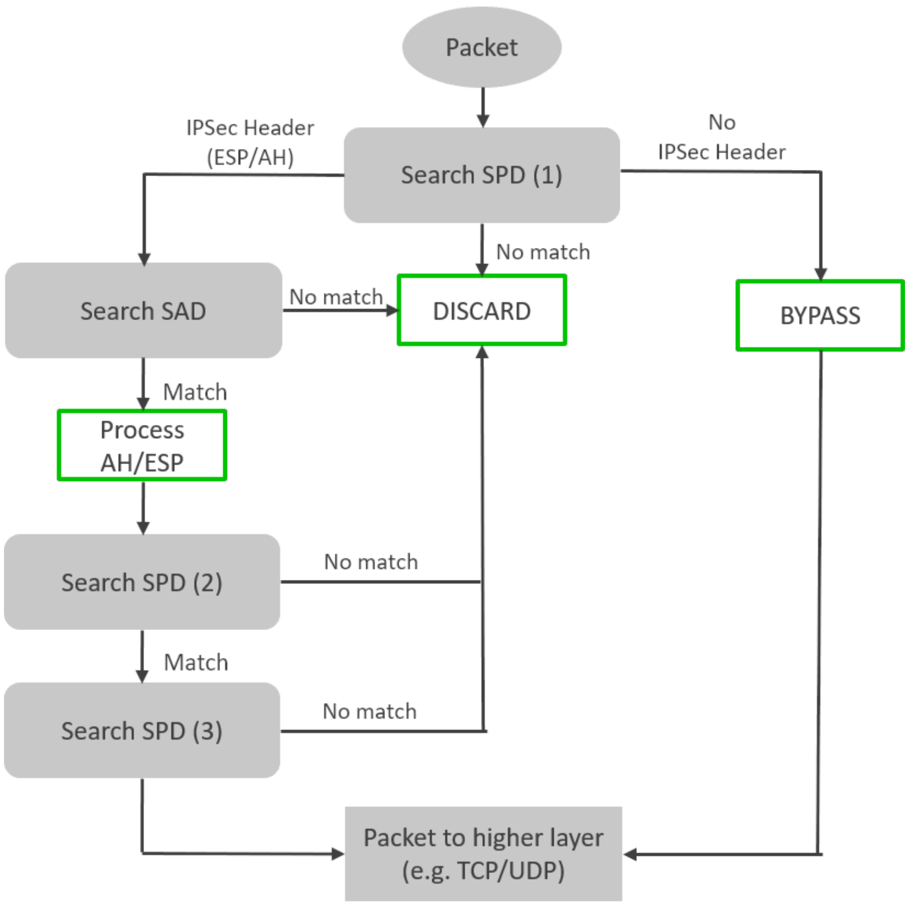


Figure 4.2. Rx check process

The process works as follows:

1. The implementation checks if the IP address is in the configured range and mode (transport). Depending on the behavior, the following decisions can be performed:
   - Discard, if the packet does not match the policies in (1) or the SP was defined as discard;
   - Protected, if the packet matches the (1) and an AH header was found;
   - Bypass, if the packet matches the policy in (1) and the SP was defined as bypass.
2. The implementation searches the SAD for the SPI included in the AH header, to find the belonging security association and verify that the anti-replay window rule was not violated (if it is enabled). If no SA is found, the implementation discards the packet. This is a deviation to IETF RFC 4202, which defines that a new SA should be negotiated if no valid SA matches the packet. As a list of all communication partners is statically configured, discarding the packet is more suitable in this case. For the encryption/verification, also several `Csm` functions need to be called, e.g. `Csm_MacVerify()` for AH.

3. The validity of the upper-layer protocol is checked, see (2) in figure Figure 4.2, “Rx check process”.
4. The destination and the source port of the UDP or TCP header are checked, see (3) in figure Figure 4.2, “Rx check process”. If the protocol is ICMP, then this check is obsolete because ICMP does not have ports.
5. If all checks are passed successfully, the packet is moved to the upper layer, e.g. TCP.

4.3.2.2. IKEv2

The Internet Key Exchange version 2 (IKEv2) is a hybrid protocol that is responsible for establishing security relationships and for negotiating and exchanging authenticated key material. Moreover, it is designed to be independent of connection protocols. That means, it can be used in a variety of key exchange scenarios. It was developed by the Internet Engineering Task Force (IETF).

IKEv2 is the newer version of the outdated IKEv1. It is responsible for creating SAs dynamically on behalf of IPsec. Furthermore, it populates and manages the SAD. The whole IKE communication consists of message pairs (request and response). This message pair is called an exchange. IKEv2 includes following communication types:

- IKE_SA_INIT exchange - creates an IKEv2 SA
- IKE_AUTH exchange - authentication of the IKEv2 SA and generation of the first child SA
- CREATE_CHILD_SA exchange - generation or rekeying of child SAs
- INFORMATIONAL exchange - control flow information/messages

Figure 4.3, “Concept of IPsec and IKEv2” visualizes the interplay of the whole IPsec and IKEv2 approach:

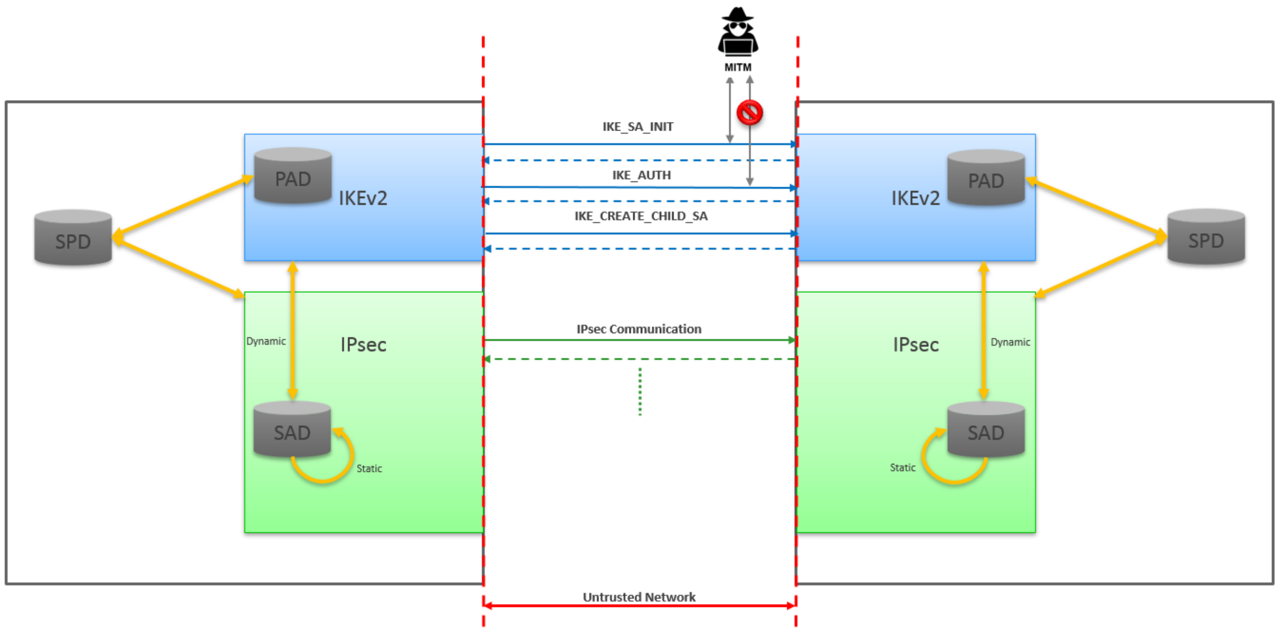


Figure 4.3. Concept of IPsec and IKEv2

4.3.2.3. IKEv2 initial exchange

The communication with IKEv2 always starts with the initial exchange, which includes the IKE_SA_INIT and the IKE_AUTH exchange. The IKE_SA_INIT pair of messages negotiate cryptographic algorithms, exchange nonces and do a Diffie-Hellman exchange for the IKE SA, while the IKE_AUTH message pair authenticate the previous messages, exchange identities, certificates, and establish the first child SA. Parts of the IKE_-AUTH message are encrypted and integrity-protected. The keys for the IKE_AUTH were negotiated via the IKE_SA_INIT exchange. Therefore, the identities are hidden from the attacker/eavesdroppers, and all fields in the messages are authenticated. A man-in-the-middle attacker who cannot complete the IKE_AUTH exchange can nonetheless see the identity of the initiator.

Moreover, the IKE SA and the IPsec SA must have a lifetime (e.g. ike lifetime = 60 min and key lifetime = 20 min) to guarantee a defined security level. Before the lifetime runs out, a rekeying message must be transmitted (specific CREATE_CHILD_SA exchange message).

All messages after the initial exchange are cryptographically protected by using the cryptographic algorithms and keys that are negotiated in the IKE_SA_INIT exchange. For the CREATE_CHILD_SA, IKE_AUTH, or INFORMATIONAL exchanges, the message after the header is encrypted and, including the header, is integrity-protected using the cryptographic algorithms negotiated in the IKE SA. Figure 4.4, “IKE_INIT_SA exchange” shows the IKE_SA_INIT pair of messages. Note: CERTREQ is optional and only used if the participant can handle certificates.

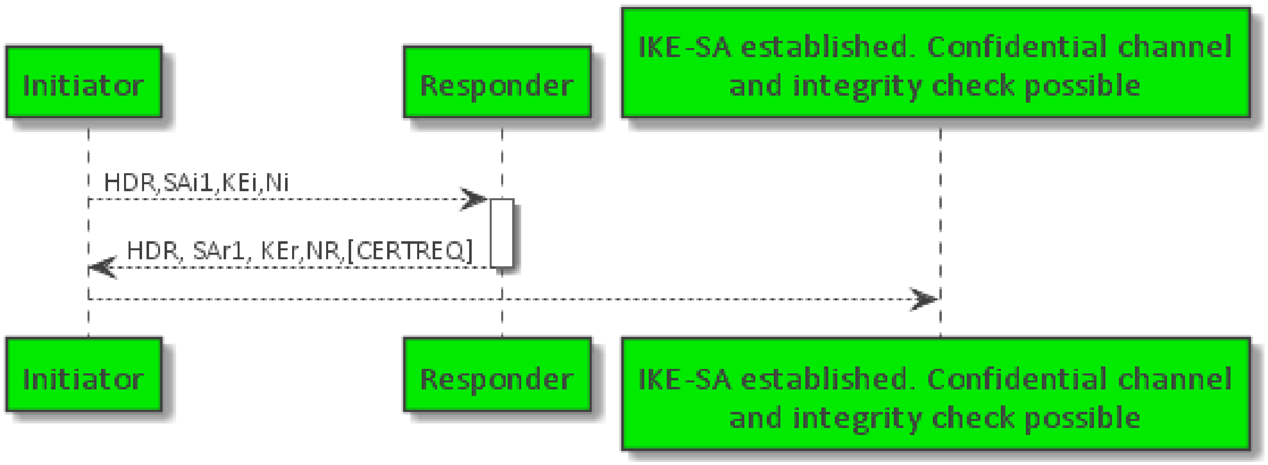

Initiator
Responder
IKE-SA established. Confidential channel and integrity check possible
HDR,SAi1,KEi,Ni
HDR, SAr1, KEr,NR,[CERTREQ]
Initiator
Responder
IKE-SA established. Confidential channel and integrity check possible

Figure 4.4. IKE_INIT_SA exchange

To create an IKE SA, the following four cryptographic algorithms (transformers) must be negotiated in a proposal and the Protocol ID must be set to IKE:

- Encryption algorithm (ENCR)
- Pseudorandom functions (PRF)
- Integrity algorithm (INTEG)

- Diffie-Hellman Group (DH)

After this exchange and the calculation of secrets, the messages are authenticated. The child SA can now be negotiated with the IKE_AUTH exchange. IKE_AUTH is used to authenticate the first two messages, exchange the identity information, and generate the first CHILD_SA (AH). The payloads of the IKE_AUTH are encrypted and authenticated using the keys exchanged in the IKE_SA_INIT exchange. The authentication method can either be *pre-shared key* or *certificates (X.509)*. Figure 4.5, “IKE_AUTH exchange” shows the IKE_AUTH message pair.

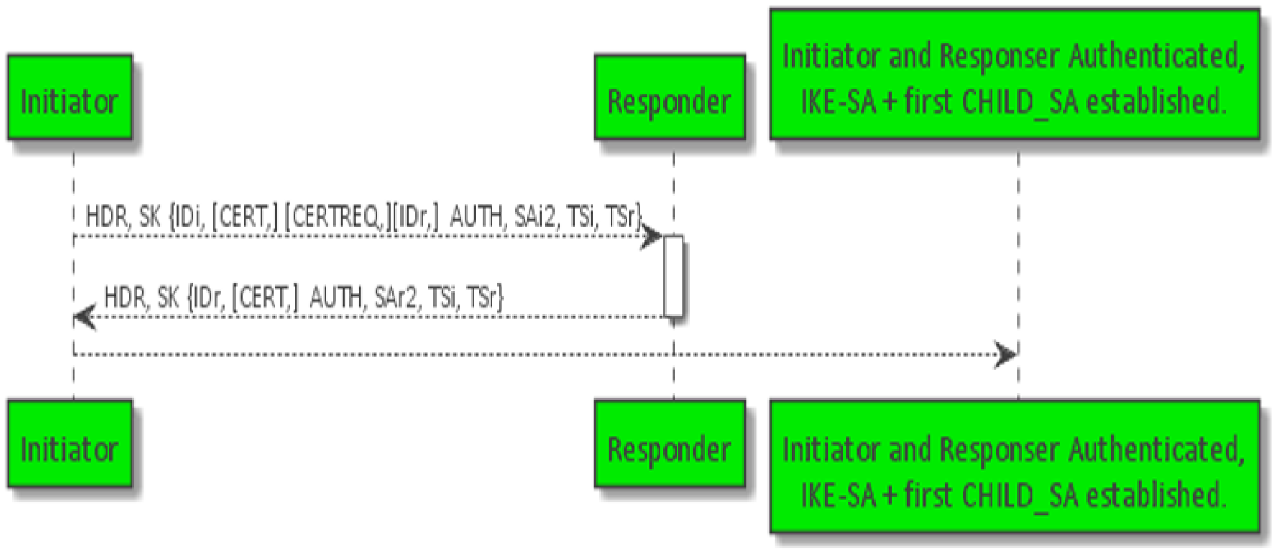

Initiator
Responder
Initiator and Responser Authenticated, IKE-SA + first CHILD_SA established.
HDR, SK {IDi, [CERT,] [CERTREQ,][IDr,] AUTH, SAi2, TSi, TSr}
HDR, SK {IDr, [CERT,] AUTH, SAr2, TSi, TSr}
Initiator
Responder
Initiator and Responser Authenticated, IKE-SA + first CHILD_SA established.

Figure 4.5. IKE_AUTH exchange

A child SA must contain the following parameters for a transform proposal if it is to be an AH SA:

- Integrity algorithm (INTEG)
- Diffie-Hellmann Group (DH) - optional
- Extended Sequence Numbers (ESN)

4.3.2.4. Create child SA exchange

The Create child SA exchange is used to create new child SAs. It is also used for the rekeying of both IKE SA and child SAs. Furthermore, it consists of a request/response pair. An SA is rekeyed by deleting the old (which should be rekeyed) and creating a new SA. In this part, all messages are encrypted.

4.4. KeyM module user guide

4.4.1. Overview

This user guide provides you with `KeyM`-specific information.

- Section 4.4.2, “Background information” explains the concepts of the `KeyM` module.
- Section 4.4.3, “Configuring the KeyM” provides information on how to configure the `KeyM` module.

4.4.2. Background information

4.4.2.1. Purpose of the KeyM module

The `KeyM` module consists of two submodules: the crypto key submodule and the certificate submodule. The crypto key submodule is provided only as needed for the certificate submodule. It manages the crypto keys that are used for storing a certificate. The certificate submodule provides services for the processing of certificates. It allows to define certificate items that can be associated in such a way that they build a certificate chain.

The parsing of certificates identifies certificate fields that are required for validation and verification. In the validation step, performed during parsing, user-configured rules are checked. Parsed certificate items that have been configured for the particular certificate are provided on request.

During the verification of certificates, the signature contained in the certificate is checked. Further, it is checked if all certificates in the corresponding certificate chain are valid certificates, i.e. have no errors during parsing or verification.

For details, see the Specification of Key Manager, AUTOSAR 4.4.0.

4.4.2.2. Comparison of periodic main function and preemptive background function

Several `KeyM` API functions contain an asynchronous part that is not executed directly when the particular function is called. This is mainly used for time-consuming processing. For example, the parsing triggered by the synchronous `KeyM_SetCertificate()` is done asynchronously. The asynchronous part is performed in either the periodic `KeyM_MainFunction()` or in the preemptive `KeyM_MainBackgroundFunction()`, to be configured in EB tresos Studio. The main difference is that the `KeyM_MainBackgroundFunction()` is interrupted as soon as a task is activated. It proceeds where it stopped when the background task is reactivated. The `KeyM_MainFunction()` has a defined set of operations that are performed in every main function cycle.

The following gives a rough overview of the main tasks that are processed by the `KeyM_MainFunction()` during one cycle:

- Call `NvM_ReadBlock()`, `NvM_WriteBlock()` or `NvM_GetErrorStatus()` if certificates are configured to be stored in `NvM`.
- Perform the parsing of a certificate.
- Check the user-configured rules for a certificate.
- Call the `Csm` signature verification job to verify a certificate.

4.4.2.3. Use of certificate element iterations

A particular certificate element can occur more than once in a certificate. In order to tell the certificate parser to handle all of these instances, the particular certificate element must be configured with activated `KeyMCertificateElementHasIteration`. Then the parser tracks all single occurrences and provides them when `KeyM_CertElementGetFirst()` and `KeyM_CertElementGetNext()` are called. If a certificate element is not retrieved by the application, the element does not need to be configured in the `KeyM`.

The configuration of `KeyMCertificateElementHasIteration` can also be used to tell the parser to handle the subelements of a certificate extension. The iterator provided by `KeyM_CertElementGetFirst()` can then be used to retrieve the single elements contained in the certificate extension when calling `KeyM_CertElementGetNext()`. It can also be used to retrieve the subelements of the `IssuerName` and `SubjectName` fields of a certificate.

If a certificate element, that is configured for iterations, is configured for certificate element verification, the rule's processing considers the multiple occurrences or subelements. Thus, the rule is applied to each single instance or subelement. The rule fails if at least one of the condition checks fails.

It is up to the application to ensure that the certificate data is not overwritten when the retrieval of all occurrences or subelements by a particular iterator is not finished yet. Using the iterator after modifying a certificate's content, without resetting it by calling `KeyM_CertElementGetFirst()`, leads to data inconsistencies.

4.4.2.4. Use of NvM storage location

During configuration of a certificate, the reference to a KeyM Crypto Key, `KeyMCryptoKey`, has to be set for storing the certificate. If the certificate shall be stored to the `NvM`, the storage location `KEYM_STORAGE_IN_NVM` has to be configured. The `KeyM` will read all referenced `NvM` blocks during initialization calling `NvM_ReadBlock()`. As long as there is no `NvM` status for each read block, the `KeyM` will reject any incoming request, e.g. `KeyM_GetCertificate`, with return value `KEYM_E_BUSY`. During processing of `KeyM_ServiceCertificate()`, the `KeyM` will store a certificate calling `NvM_WriteBlock()`.

NOTE

Calling `KeyM_SetCertificate()` for a certificate with NvM storage location, `KEYM_STORAGE_IN_NVM`, will result in an error, and `E_NOT_OK` will be returned.

The configuration of the `NvM` optionally allows to define a permanent RAM block, `NvMRamBlockDataAddress`, for a configured block descriptor. The `KeyM` does not make use of it. It holds an internal key data array for storing the certificate data and it destroys the data during `KeyM_Deinit()`.

4.4.3. Configuring the KeyM

Beside general configuration options, the `KeyM` editor provides configuration options for:

- `KeyM` crypto keys
- `KeyM` certificates
- `KeyM` certificate element verifications

4.4.3.1. General configuration

On the **General** tab, you can configure the following parameters:

- `KeyMCertificateChainMaxDepth`: Maximum number of certificates that have to be handled in a certificate chain
- `KeyMKeyCertNameMaxLength`: Maximum length in bytes of a certificate name used for the service interface
- `KeyMMainFunctionPeriod`: Time period of the main function `KeyM_MainFunction()` in seconds

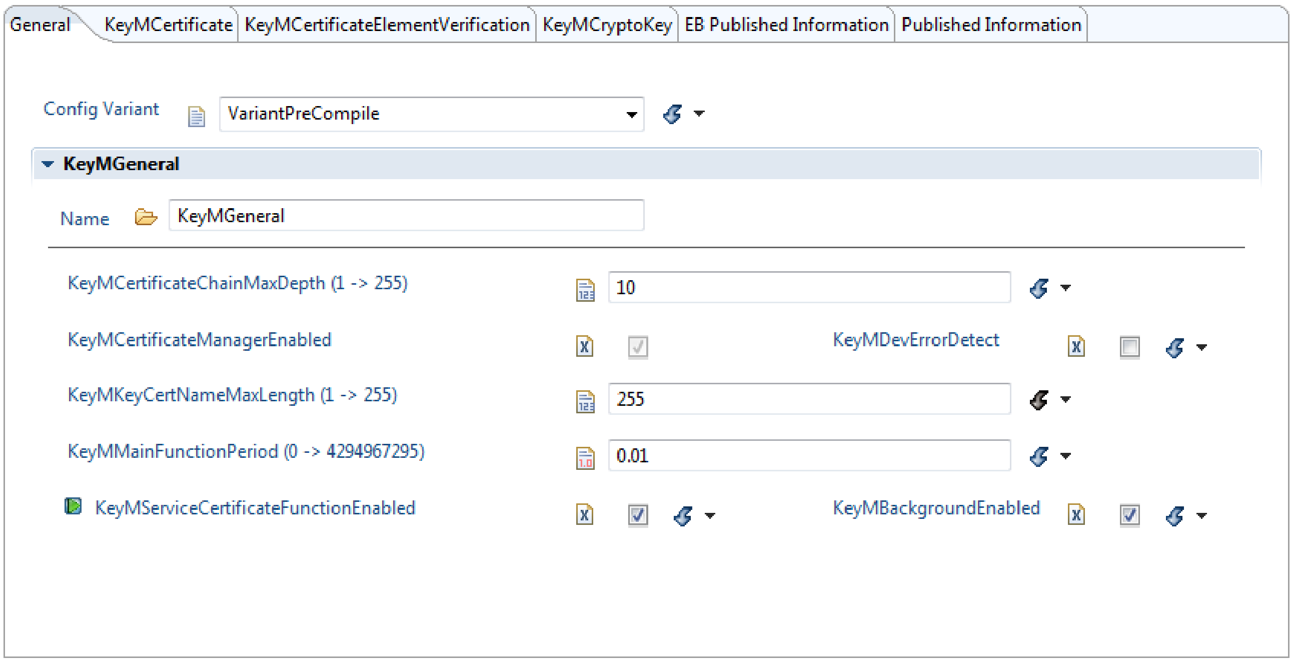
  NOTE

  Consideration of `KeyMMainFunctionPeriod`

  The `KeyMMainFunctionPeriod` parameter is mandatory according to the Specification of Key Manager, AUTOSAR 4.4.0. Its value is not used if the `KeyM_MainBackgroundFunction()` is configured instead of the default `KeyM_MainFunction()`.

- `KeyMBackgroundEnabled`: Enables (TRUE) or disables (FALSE) the `KeyM_MainBackgroundFunction()` instead of the default `KeyM_MainFunction()`.

For the configuration of the other `KeyM` parameters, see the corresponding descriptions in Chapter 5, “ACG8 SECEXT module references“. For a configuration example, see the following picture.



Config Variant: VariantPreCompile

KeyMGeneral

Name: KeyMGeneral

KeyMCertificateChainMaxDepth (1 -> 255): 10

KeyMCertificateManagerEnabled: ✓

KeyMDevErrorDetect: ☐

KeyMKeyCertNameMaxLength (1 -> 255): 255

KeyMMainFunctionPeriod (0 -> 4294967295): 0.01

KeyMServiceCertificateFunctionEnabled: ✓

KeyMBackgroundEnabled: ✓

Figure 4.6. Example for general configuration

4.4.3.2. Configuring a `KeyM` crypto key

On the **KeyMCryptoKey** tab, you configure a `KeyM` crypto key. Specify the following parameters by selecting an entry from a drop-down list box.

- `KeyMCryptoKeyGenerationType`: Identifier whether the `KeyM` crypto key shall be derived from another key or simply stored with `Csm_KeyElementSet()`.

  NOTE

  Limited selection for `KeyMCryptoKeyGenerationType`

  Currently only `KEYM_STORED_KEY` is supported.

- `KeyMCryptoKeyStorage`: Type of storage location to be used for the crypto key. For example, use `KEYM_STORAGE_IN_RAM` if you do not want to store the crypto key in either non-volatile memory (`NvM`) or to a configured `Csm` key.

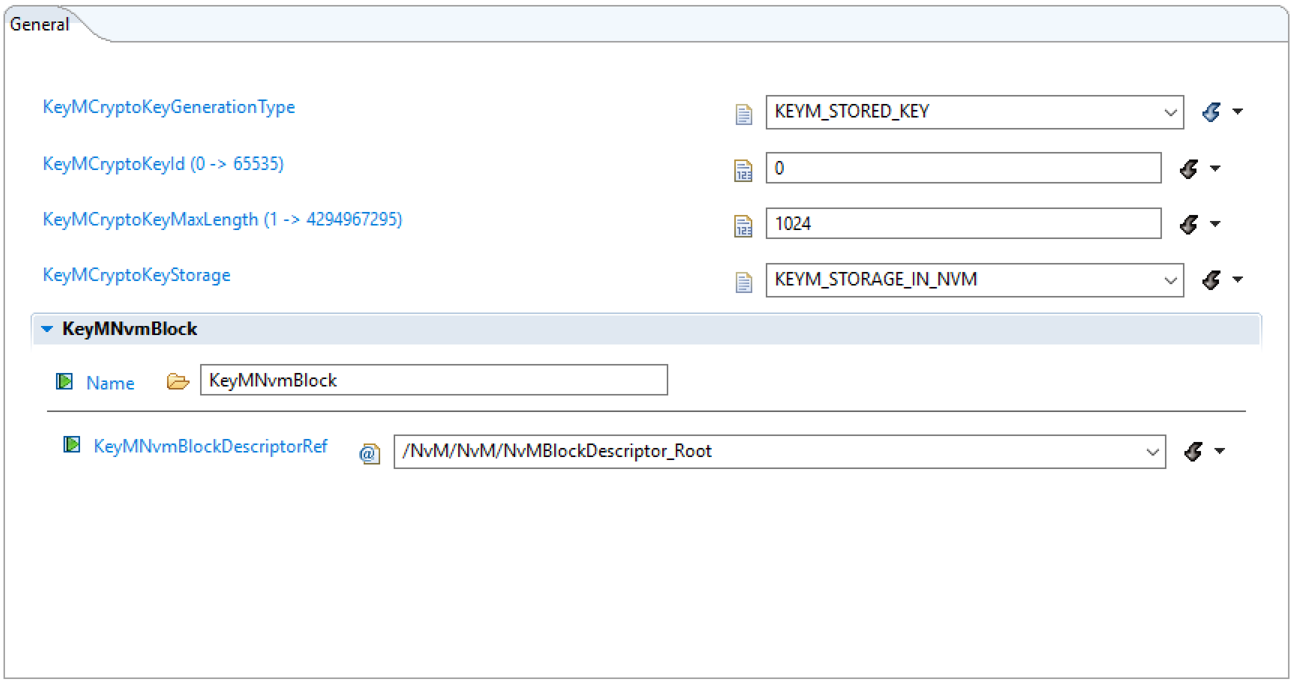
NOTE

Limited selection for `KeyMCryptoKeyStorage`

Currently only `KEYM_STORAGE_IN_RAM` and `KEYM_STORAGE_IN_NVM` are supported.

- `KeyMNvmBlock/KeyMNvmBlockDescriptorRef`: Reference to the `NvM` block that is used to store the crypto key data persistently if the parameter `KeyMCryptoKeyStorage` is set to `KEYM_STORAGE_IN_-NVM`.

For the configuration of the other parameters, see the corresponding descriptions in Chapter 5, “ACG8 SECEXT module references“. For a configuration example, see the following picture.


General

KeyMCryptoKeyGenerationType: KEYM_STORED_KEY
KeyMCryptoKeyId (0 -> 65535): 0
KeyMCryptoKeyMaxLength (1 -> 4294967295): 1024
KeyMCryptoKeyStorage: KEYM_STORAGE_IN_NVM

KeyMNvmBlock
Name: KeyMNvmBlock
KeyMNvmBlockDescriptorRef: /NvM/NvM/NvMBlockDescriptor_Root

Figure 4.7. Configuration example of a specific `KeyM` crypto key

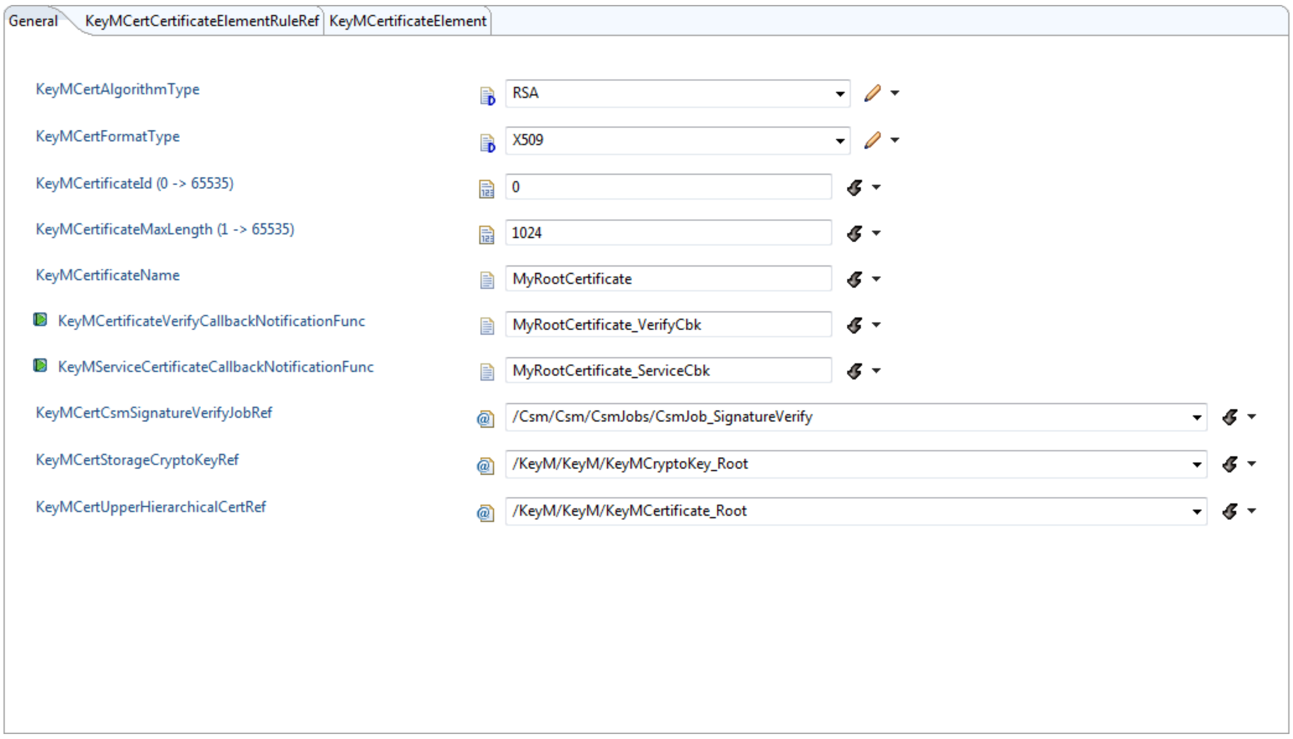
4.4.3.3. Configuring a `KeyM` certificate

On the **KeyMCertificate** tab, you configure a `KeyM` certificate. Specify the following parameters by selecting an entry from a drop-down list box.

- `KeyMCertCsmSignatureVerifyJobRef`: Reference to the `Csm` job that is used to verify the signature.
- `KeyMCertStorageCryptoKeyRef`: Storage location that shall be used for the certificate. It references a specific `KeyM` crypto key.
- `KeyMCertUpperHierarchicalCertRef`: Identifier of the certificate that is the next higher certificate in the hierarchical public key infrastructure (PKI).

- `KeyMCertCertificateElementRuleRef`: A list of certificate element rules that shall be checked during the validation step of a certificate's parsing. The validation of the certificate elements can be done in a `KeyMCertificateElementCondition` and `KeyMCertCertificateElementRule`.

For the configuration of the other parameters, see the corresponding descriptions in Chapter 5, “ACG8 SECEXT module references“. For a configuration example, see the following picture. Here, a root certificate is configured, i.e. the next higher certificate in the chain references itself.



KeyMCertAlgorithmType	RSA
KeyMCertFormatType	X509
KeyMCertificateId (0 -> 65535)	0
KeyMCertificateMaxLength (1 -> 65535)	1024
KeyMCertificateName	MyRootCertificate
KeyMCertificateVerifyCallbackNotificationFunc	MyRootCertificate_VerifyCbk
KeyMServiceCertificateCallbackNotificationFunc	MyRootCertificate_ServiceCbk
KeyMCertCsmSignatureVerifyJobRef	/Csm/Csm/CsmJobs/CsmJob_SignatureVerify
KeyMCertStorageCryptoKeyRef	/KeyM/KeyM/KeyMCryptoKey_Root
KeyMCertUpperHierarchicalCertRef	/KeyM/KeyM/KeyMCertificate_Root

Figure 4.8. Configuration example of a root certificate

4.4.3.4. Configuring a `KeyM` certificate element

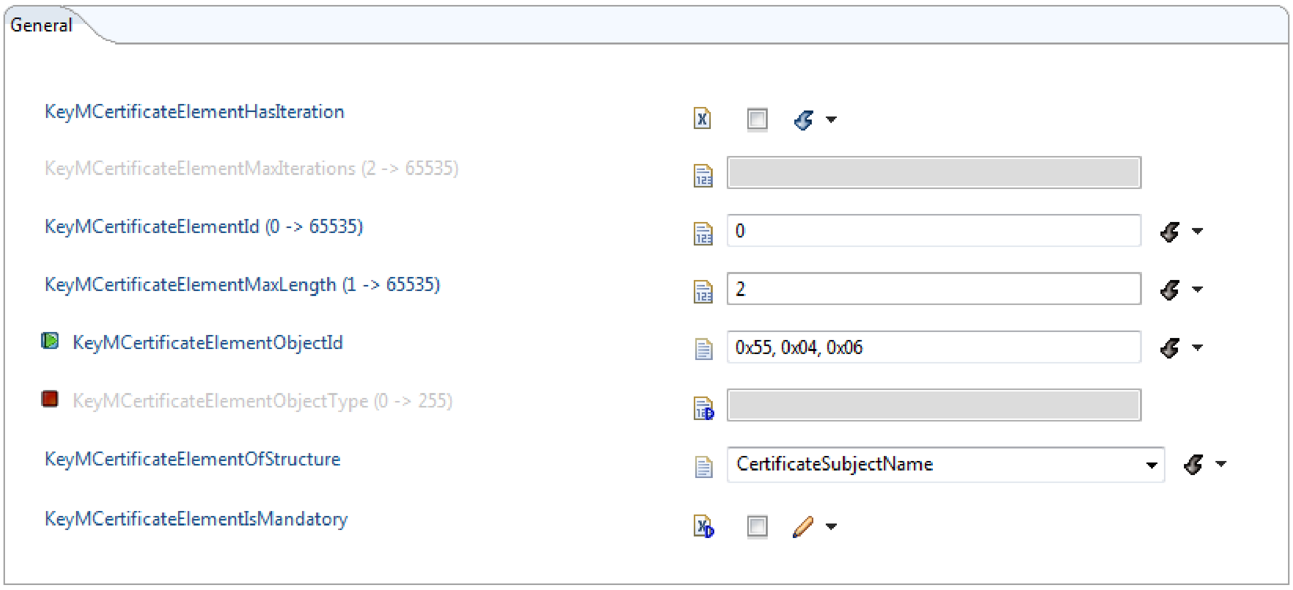
The certificate elements that shall be made available for retrieval by e.g. `KeyM_CertElementGet()` have to be configured. The element configuration is further used to validate the certificate during parsing, e.g. regarding size.

On the **KeyMCertificateElement** tab, you configure a `KeyM` certificate element. Specify the following parameters by selecting an entry from a drop-down list box.

- `KeyMCertificateElementHasIteration`: Defines if the certificate element can occur more than once. If the element is an extension, this parameter defines if the extension has subelements. If `KeyM-CertificateElementHasIteration` is set to TRUE, an iterator can be used to retrieve the individual data values of this certificate element.

- `KeyMCertificateElementMaxIterations`: Maximum number of iterations.
- `KeyMCertificateElementMaxLength`: Maximum length in bytes of the certificate element. The length is validated during the parsing of a certificate.
- `KeyMCertificateElementObjectId`: The object identifier (OID) of the certificate element that either corresponds to the configured `KeyMCertificateElementOfStructure` or that is a subelement of the configured one. The value is parsed as comma-separated byte values given in hexadecimal representation. For example, `0x55,0x1D,0x23` stands for `2.5.29.35` and denotes the `AuthorityKeyIdentifier` extension.
- `KeyMCertificateElementObjectType`: The type of the corresponding ASN.1 TLV (tag-length-value). For example: The tag value '2' denotes the ASN.1's built-in simple type INTEGER. By setting `KeyMCertificateElementObjectType`, the parser will choose the first occurrence of an element in the `KeyMCertificateElementOfStructure` that matches the given datatype. If the element is additionally configured for iteration, then all occurrences of the given `KeyMCertificateElementObjectType` can be retrieved.
- `KeyMCertificateElementOfStructure`: The structure that contains this certificate element.
- `KeyMCertificateElementIsMandatory`: Sets the certificate element as mandatory (TRUE) or optional (FALSE). If the certificate element is set as mandatory, the parsing of the certificate fails if the element is not found. If it is set as optional, the parser ignores the certificate element if it is not found.

For the configuration of the other parameters, see the corresponding descriptions in Chapter 5, “ACG8 SECEXT module references“. For a configuration example, see the following picture. Here, a subelement of the structure SubjectName is configured by supplying an OID.


General

KeyMCertificateElementHasIteration	☐
KeyMCertificateElementMaxIterations (2 -> 65535)	
KeyMCertificateElementId (0 -> 65535)	0
KeyMCertificateElementMaxLength (1 -> 65535)	2
KeyMCertificateElementObjectId	0x55, 0x04, 0x06
KeyMCertificateElementObjectType (0 -> 255)	
KeyMCertificateElementOfStructure	CertificateSubjectName
KeyMCertificateElementIsMandatory	☐

Figure 4.9. Configuration example of a specific `KeyM` certificate element

4.4.3.5. Configuring a `KeyM` certificate element verification

On the **KeyMCertificateElementVerification** tab, you configure a `KeyM` certificate element verification. You need to reference a `KeyMCertificateElementRule`. Further, specify the following parameters by selecting an entry from a drop-down list box.

- `KeyMCertificateElementCondition`: A list of certificate element compare conditions that can be used as arguments for a `KeyMCertificateElementRule`.
- `KeyMCertificateElementRule`: A list of certificate element compare rules.

4.4.3.6. Configuring a `KeyM` certificate element rule

On the **KeyMCertificateElementRule** tab, you configure a `KeyM` certificate element rule. Specify the following parameters by selecting an entry from a drop-down list box.

- `KeyMLogicalOperator`: Specifies the logical operator that is applied to the rules and conditions referenced in `KeyMArgumentRef`.
- `KeyMArgumentRef`: A list of certificate element rules and conditions that are checked by this rule. Circular references of rules and a rule that only contains another rule are not allowed.

4.4.3.7. Configuring a `KeyM` certificate element condition

On the **KeyMCertificateElementCondition** tab, you configure a `KeyM` certificate element condition. Specify the following parameters by selecting an entry from a drop-down list box.

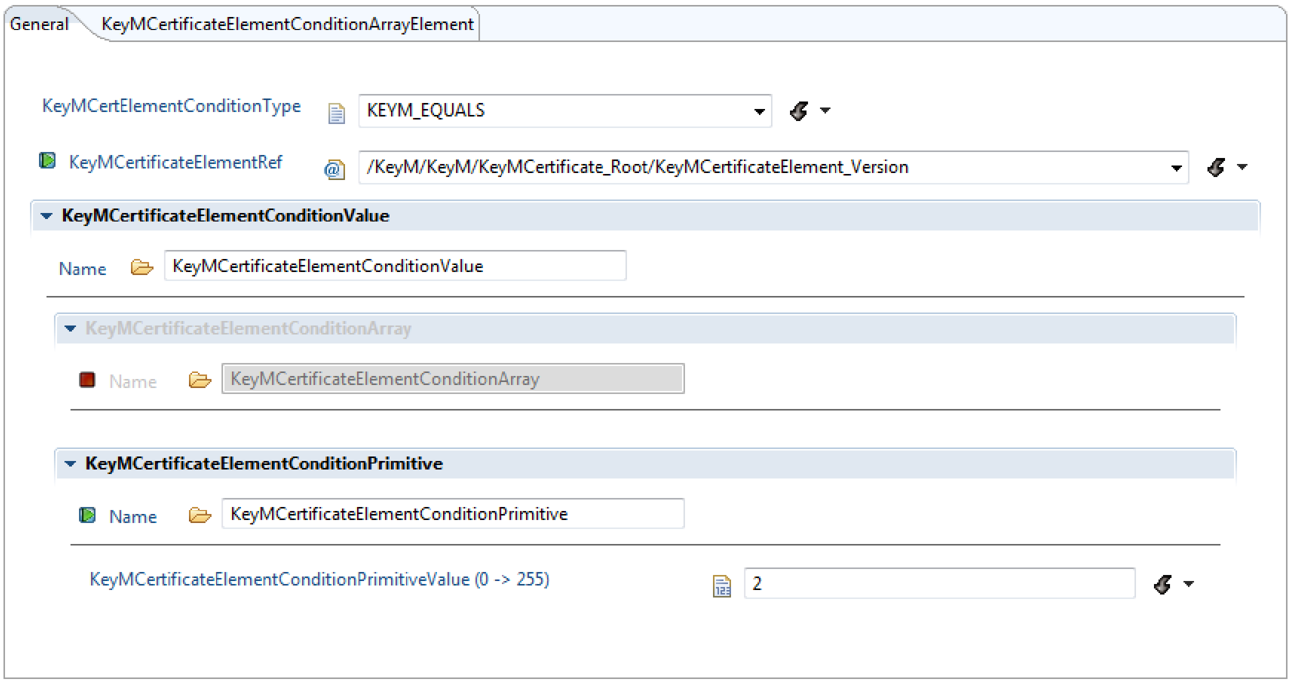
- `KeyMCertElementConditionType`: The kind of comparison that shall be made for the evaluation of the condition, e.g. KEYM_EQUALS.
- `KeyMCertificateElementRef`: Reference to the `KeyMCertificateElement` this condition check shall be applied to. The referenced certificate element is the first parameter in the condition check.
- `KeyMCertificateElementConditionArray`: Enables (TRUE) or disables (FALSE) the comparison of the certificate element against the array that is configured in the **KeyMCertificateElementConditionArray** tab.
- `KeyMCertificateElementConditionPrimitive`: Enables (TRUE) or disables (FALSE) the comparison of the certificate element against the value that is configured in the `KeyMCertificateElementConditionPrimitiveValue` parameter.

NOTE

Configuring index in `KeyMCertificateElementConditionArray`

The check of a `KeyMCertificateElementConditionArray` condition fails during parsing of a certificate if the index `KeyMCertificateElementConditionArrayElementIndex` is out of range of the actual elements size.

For the configuration of the other `KeyM` configuration parameters, see the corresponding descriptions in Chapter 5, “ACG8 SECEXT module references“. For a configuration example, see the following picture. Here, a subelement of the structure SubjectName is configured by supplying an OID.



KeyMCertElementConditionType: KEYM_EQUALS
KeyMCertificateElementRef: /KeyM/KeyM/KeyMCertificate_Root/KeyMCertificateElement_Version

KeyMCertificateElementConditionValue
Name: KeyMCertificateElementConditionValue

KeyMCertificateElementConditionArray
Name: KeyMCertificateElementConditionArray

KeyMCertificateElementConditionPrimitive
Name: KeyMCertificateElementConditionPrimitive
KeyMCertificateElementConditionPrimitiveValue (0 -> 255): 2

Figure 4.10. Configuration example of a specific `KeyM` certificate condition

4.4.3.8. Additional notes on configuring the `KeyM`

Unsupported `KeyM` configuration options are not visible. Therefore you cannot configure them. For some of these hidden configuration options, EB tresos Studio issues warnings as follows:

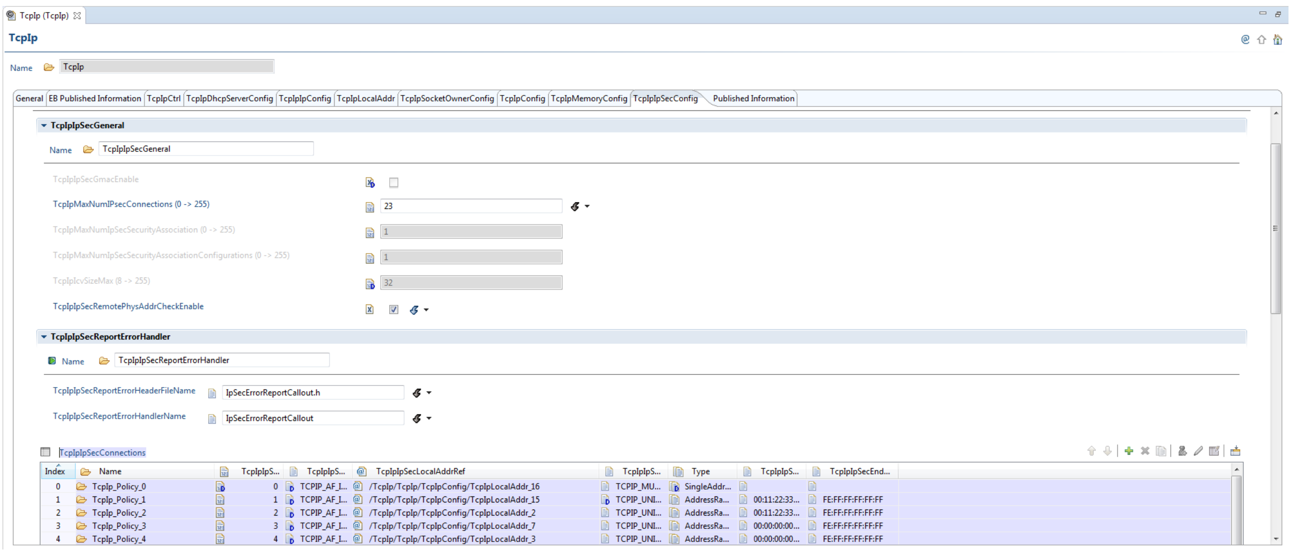
- The node "/AUTOSAR/TOP-LEVEL-PACKAGES/KeyM/ELEMENTS/KeyM/KeyMCryptoKey/<KeyMCryptoKeyName>/KeyMCryptoKeyCsmKeyTargetRef" with value "" does not refer to nodes.
- The node "/AUTOSAR/TOP-LEVEL-PACKAGES/KeyM/ELEMENTS/KeyM/KeyMCertificate/<KeyMCertificateName>/KeyMCertTimebaseRef" with value "" does not refer to nodes.

You can ignore these warnings.

4.5. TcpIp firewall user guide

4.5.1. Overview

This user guide provides information about the firewall feature of the `TcpIp` module in the AUTOSAR context.


TcpIp (TcpIp)

TcpIp

Name: TcpIp

General | EB Published Information | TcpIpCtrl | TcpIpDhcpServerConfig | TcpIpIpConfig | TcpIpLocalAddr | TcpIpSocketOwnerConfig | TcpIpConfig | TcpIpMemoryConfig | TcpIpIpSecConfig | Published Information

TcpIpIpSecGeneral

Name	TcpIpIpSecGeneral
TcpIpIpSecGmacEnable	☐
TcpIpMaxNumIPsecConnections (0 -> 255)	23
TcpIpMaxNumIpSecSecurityAssociation (0 -> 255)	1
TcpIpMaxNumIpSecSecurityAssociationConfigurations (0 -> 255)	1
TcpIpIcvSizeMax (8 -> 255)	32
TcpIpIpSecRemotePhysAddrCheckEnable	✓

TcpIpIpSecReportErrorHandler

Name	TcpIpIpSecReportErrorHandler
TcpIpIpSecReportErrorHeaderFileName	IpSecErrorReportCallout.h
TcpIpIpSecReportErrorHandlerName	IpSecErrorReportCallout

TcpIpIpSecConnections

Index	Name	TcpIpIpS...	TcpIpIpS...	TcpIpIpSecLocalAddrRef	TcpIpIpS...	Type	TcpIpIpS...	TcpIpIpSecEnd...
0	TcpIp_Policy_0	0	TCPIP_AF_I...	/TcpIp/TcpIp/TcpIpConfig/TcpIpLocalAddr_16	TCPIP_MU...	SingleAddr...		
1	TcpIp_Policy_1	1	TCPIP_AF_I...	/TcpIp/TcpIp/TcpIpConfig/TcpIpLocalAddr_15	TCPIP_UNI...	AddressRa...	00:11:22:33...	FE:FF:FF:FF:FF:FF
2	TcpIp_Policy_2	2	TCPIP_AF_I...	/TcpIp/TcpIp/TcpIpConfig/TcpIpLocalAddr_2	TCPIP_UNI...	AddressRa...	00:11:22:33...	FE:FF:FF:FF:FF:FF
3	TcpIp_Policy_3	3	TCPIP_AF_I...	/TcpIp/TcpIp/TcpIpConfig/TcpIpLocalAddr_7	TCPIP_UNI...	AddressRa...	00:00:00:00...	FE:FF:FF:FF:FF:FF
4	TcpIp_Policy_4	4	TCPIP_AF_I...	/TcpIp/TcpIp/TcpIpConfig/TcpIpLocalAddr_3	TCPIP_UNI...	AddressRa...	00:00:00:00...	FE:FF:FF:FF:FF:FF

- Section 4.5.2, “Background information” describes the basic concept of the `TcpIp` firewall.
- Section 4.5.3, “Configuring the filtering feature” provides information on how to configure the packet filtering policies feature of the firewall.
- Section 4.5.4, “Configuring the logging feature” provides information on how to configure the logging feature of the firewall.

4.5.2. Background information

The `TcpIp` firewall feature provides an access control at the IP layer by offering packet filtering and logging. Inbound messages are filtered by the following information:

- protocol
- remote port
- remote address
- local port
- local address

- remote physical address

The logging functionality keeps track of the number of packets accepted and rejected by the firewall. Additionally, an error callout can be configured to report every rejected message. By enabling `TcpIpGetAndReset-MeasurementDataApi`, the counters can be read and/or reset.

4.5.2.1. Packet filtering

The `TcpIp` firewall feature allows you to configure connection policies with the following parameters.

Parameter	**Description**
Protocol	Header value indicating the protocol after the IP header. A possible combination of values is: - TCP (6) - UDP (17) - ICMP - V4 (1) or V6 (58) - ANY - wildcard, any protocol value is accepted - In case of an IpV4 address, ARP messages are also filtered for source IP, destination IP, and remote physical address.
Local IP address	IP address of the local host
Local port	Port range of the local host Not applicable if the Protocol value is configured to `ICMP`.
Remote IP address	IP address of the remote host - If the remote IP address of the security policy is configured to `MULTICAST`, only outbound packets can be filtered. - Can be expressed as a single address or an address range.
Remote port	Port range of the remote host Not applicable if the Protocol value is configured to `ICMP`.
Remote physical address	Physical address of the remote host - Expressed as a range of values - Can be enabled or disabled per policy

4.5.3. Configuring the filtering feature

Configuring the firewall filtering

Prerequisite:

- The `TcpIp` module is added and configured in EB tresos Studio. For configuration guidance, see the parameter descriptions in EB tresos Studio or in the `TcpIp` module references of the EB tresos AutoCore Generic 8 IP Stack documentation.

Step 1
Set the `TcpIpGeneral/TcpIpSecurityMode` parameter to `FIREWALL`.

Step 2
Go to tab **TcpIpIpSecGeneral/TcpIpIpSecConfig**.

Step 3
Set the `TcpIpMaxNumIPsecConnections` parameter to the desired number of trusted connections, i.e. security policies. This number defines the maximum number of connection entries that you can add to the `TcpIpIpSecConnections` list.

Step 4
For filtering by remote physical address, enable the parameter `TcpIpIpSecGeneral/TcpIpIpSecRemotePhysAddrCheckEnable`. This allows you to configure a range of physical addresses for each policy.

Step 5
Open an entry in `TcpIpIpSecConnections`. On tab **General**, configure the following:

- `TcpIpIpSecConId`: unique connection ID
- `TcpIpIpSecDomainType`:
  - For IPv4, select TCPIP_AF_INET.
  - For IPv6, select TCPIP_AF_INET6.
- `TcpIpIpSecLocalAddrRef`: reference to the local address entry in the `TcpIpConfig/TcpIpLocalAddr` list.
- `TcpIpIpSecRemoteAddrType`: Select `UNICAST` or `MULTICAST`.
- `IpSecRemoteAddrConfig`:
  - `SingleAddress`: If selected, specify an address in `SingleAddress/TcpIpIpSecRemoteAddr`.
  - `AddressRange`: If selected, specify the start address in `AddressRange/TcpIpIpSecStartRemoteAddr` and the end address in `AddressRange/TcpIpIpSecEndRemoteAddr`. This defines the interval of IP addresses accepted by this policy.

    Notes:

  - If `TcpIpIpSecRemoteAddrType` is set to `TCPIP_UNICAST`, make sure to use only UNICAST address values. This is to avoid overlapping with a multicast address in range mode and vice versa.
  - The IPv4 MULTICAST address range is 224.0.0.0 to 239.0.0.0.
  - BROADCAST 255.255.255.255 and LIMITED BROADCAST (e.g. netmask 24 xxx.xxx.255.255) are also considered as IPv4 MULTICAST.
  - Any IPv6 address with prefix FF02 is considered as MULTICAST.
- `TcpIpIpSecRemotePhysAddr` (optional): Configurable if `TcpIpIpSecGeneral/TcpIpIpSecRemotePhysAddrCheckEnable` is enabled and `TcpIpIpSecRemoteAddrType` is set to `TCPIP_UNICAST`.

  If enabled, specify the start physical address in `TcpIpIpSecRemotePhysAddr/TcpIpIpSecStartRemotePhysAddr` and the end physical address in `TcpIpIpSecRemotePhysAddr/TcpIpIpSecEndRemotePhysAddr`. This defines the interval of IP addresses accepted by this policy.

  Note: Keep in mind that the multicast physical address also differs for the version IPv4 (multicast prefix is 01:00:5E:xx:xx:xx with the first bit of the fourth byte set to 0) and IPv6 (multicast prefix is 33:33:xx:xx:xx:xx). If the configured address interval contains a multicast physical address and during run-time a packet is received from it, the packet is rejected.

Step 6
Go to tab **TcpIpIpIpSecSecurityPolicies** to configure the protocol and ports.

Step 7
Add an entry to the `TcpIpIpIpSecSecurityPolicies` list and open it to configure the following parameters:

- `TcpIpIpSecSecurityPolicyMechanism`: BYPASSED or SECURED (not available in FIREWALL mode)

  In FIREWALL mode, only the BYPASS mechanism is available. This means the policy accepts messages matching the configured parameters.
- `TcpIpIpSecSecurityPolicyDirection`: Specifies the direction in which the policy is applied:
  - INBOUND - received messages
  - OUTBOUND - transmitted messages
- On tab **TcpIpIpIpSecSecurityRule**, add entries and configure as required:
  - `TcpIpIpSecSecurityPolicyDirection`: can be set to `LOCAL` or `REMOTE`.
  - `TcpIpIpSecSecurityPolicyUpperLayer`: can be set to `TCP`, `UDP`, `ICMP`, or `ANY`.
  - `TcpIpIpSecSecurityPolicyStartPort`: a 16 bit integer representing the start of the port interval.

- `TcpIpIpSecSecurityPolicyEndPort`: a 16 bit integer representing the end of the port interval.

  You can configure multiple security rules, with at least one rule configured for LOCAL and one rule for REMOTE.

Step 8
If applicable, add and configure another `TcpIpIpIpSecSecurityPolicies` entry.

You can configure multiple `TcpIpIpIpSecSecurityPolicies` entries. There should always be configured at least an INBOUND policy and an OUTBOUND policy, with the following exceptions:

- If the remote address is MULTICAST, only an OUTBOUND policy is allowed.
- If the local address is MULTICAST, only an INBOUND policy is allowed.

4.5.4. Configuring the logging feature

With two separate 32-bit counters, the firewall feature keeps track of inbound packets that are rejected and accepted by the security policies.

When you configured security policies for packet filtering, you can also enable the logging feature. The following services are available:

- GetAndResetMeasurementData
- TcpIpIpSecReportErrorHandler

4.5.4.1. Configuring the GetAndResetMeasurementData service

Configuring the GetAndResetMeasurementData service

Step 1
To activate this service, enable the parameter `TcpIpGeneral/TcpIpGetAndResetMeasurement-DataApi`. If enabled, `TcpIp_GetAndResetMeasurementData()` can be called during run-time in order to read the counters and/or reset them.

When calling the API, the following values can be passed as `MeasurementIdx`:

Value	Description
`TCPIP_MEAS_PASS_-VALID_POLICY`	▶ Returns the number of inbound messages accepted by the firewall through the `MeasurementDataPtr`.

Value	Description
	▶ Optionally, you can set `MeasurementResetNeeded` to `TRUE`/`FALSE` in order to reset the counter or not after it was read.
`TCPIP_MEAS_PASS_INVALID_POLICY`	▶ Returns the number of inbound messages rejected by the firewall through the `MeasurementDataPtr`. ▶ Optionally, you can set `MeasurementResetNeeded` to `TRUE`/`FALSE` in order to reset the counter or not after it was read.
`TCPIP_MEAS_ALL`	Restarts both counters. Note that this resets all other counters, e.g. UDP, ICMP, TCP.

4.5.4.2. Configuring the TcpIpIpSecReportErrorHandler service

Configuring the TcpIpIpSecReportErrorHandler service

Step 1
To activate this service, enable `TcpIpIpSecConfig/TcpIpIpSecReportErrorHandler`. If enabled, the API `TcpIp_IpSecReportError()` is called every time the firewall rejects a packet.

When `TcpIpIpSecReportErrorHandler` is enabled, make sure to provide a custom function call name and the file in the configuration.

Step 2
In `TcpIpIpSecConfig/TcpIpIpSecReportErrorHandler/TcpIpIpSecReportErrorHeaderFileName`, specify the file name with the extension `.h`.

Step 3
In `TcpIpIpSecConfig/TcpIpIpSecReportErrorHandler/TcpIpIpSecReportErrorHandlerName`, specify the function name.

Step 4
For the function syntax, see the following example:

```
TcpIpFirewallReportError
       (
         uint8 ctrlIdx,
         P2CONST(TcpIp_SockAddrType, AUTOMATIC, TCPIP_APPL_DATA) localSockAddPtr,
         P2CONST(TcpIp_SockAddrType, AUTOMATIC, TCPIP_APPL_DATA) remoteSockAddPtr,
         uint32 numberOfValidPolicies,
         uint32 numberOfInvalidPolicies,
         uint8 errorCode
       )
```

where:

- `ctrlIdx` indicates the `TcpIp` controller on which the rejected message was received.
- `localSockAddPtr` indicates the destination (local) address of the rejected message received.
- `remoteSockAddPtr` indicates the source (remote) address of the rejected message received.
- `numberOfValidPolicies` indicates the total number of packets accepted by the firewall.
- `numberOfInvalidPolicies` indicates the total number of packets rejected by the firewall, including the current one.
- errorCode should indicate `TCPIP_MEAS_DROP_INVALID_POLICY`.

5. ACG8 SECEXT module references

5.1. Overview

This chapter provides module references for the ACG8 SECEXT product modules. These include a detailed description of all configuration parameters. Furthermore this chapter lists the application programming interface with all data types, constants and functions.

The content of the sections is sorted alphabetically according the EB tresos AutoCore Generic module names.

For further information on the functional behavior of these modules, refer to the chapter ACG8 SECEXT user's guide.

5.1.1. Notation in EB module references

EB notation may differ from the AUTOSAR standard notation in the software specification documents (SWS). This section describes the notation of *default value* and *range* fields in the EB module references.

5.1.1.1. Default value of configuration parameters

If there is no default value specified for a parameter, the default value field is omitted to prevent ambiguity with parameters that have `--` as default values.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has no default value field, therefore it is omitted.

5.1.1.2. Range information of configuration parameters

The range of a configuration parameter contains an upper and a lower boundary. However, in special cases the range of allowed values can be computed by means of an XPath function that is evaluated at configuration time. An XPath function can either be a standard `xpath:<function>()` or a custom `cxpath:<function>()` function. The range of a configuration parameter may be computed based on other configuration parameters that are referenced from the XPath function. For more information on custom XPath functions, see section *Custom XPath Functions API* of the EB tresos Studio developer's guide.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has the custom XPath function `cxpath:getCompuMethodsVT()` in the range field which provides the allowed values.

5.2. KeyM

5.2.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
KeyMGeneral	1..1	This container holds general configuration (parameters) for key manager.
KeyMCertificate	0..65535	This container contains the certificate configuration.
KeyMCertificateElementVerification	0..65535	This container defines if and how certificate elements are to be verified.
KeyMCryptoKey	0..65535	This container contains the crypto keys that can be updated.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	**IMPLEMENTATION_CONFIG_VARIANT**
Label	Config Variant
Multiplicity	1..1
Type	ENUMERATION
Default value	VariantPreCompile
Range	VariantPreCompile

Configuration class	**VariantPreCompile:**	VariantPreCompile

5.2.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	**ArMajorVersion**	
Label	AUTOSAR Major Version	
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.	
Multiplicity	1..1	
Type	INTEGER_LABEL	
Default value	4	
Configuration class	**PublishedInformation:**	
Origin	Elektrobit Automotive GmbH	

Parameter Name	**ArMinorVersion**
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	4

Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArPatchVersion**
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version

Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	11
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	109
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**Release**
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.2.1.2. KeyMGeneral

Parameters included	
Parameter name	**Multiplicity**
KeyMCertificateChainMaxDepth	1..1
KeyMCertificateManagerEnabled	1..1
KeyMDevErrorDetect	1..1
KeyMKeyCertNameMaxLength	1..1
KeyMMainFunctionPeriod	1..1
KeyMServiceCertificateFunctionEnabled	0..1
KeyMBackgroundEnabled	1..1

Parameter Name	KeyMCertificateChainMaxDepth	
Description	Maximum number of certificates defined in a certificate chain.	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Range	<=255	
	>=1	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	KeyMCertificateManagerEnabled	
Description	Enables (TRUE) or disables (FALSE) the part that manages certificates.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	KeyMDevErrorDetect
Description	Switches the development error detection and notification on or off.
Multiplicity	1..1
Type	BOOLEAN

Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMKeyCertNameMaxLength**	
Description	Maximum length in bytes of certificate or key names used for the service interface.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=255	
	>=1	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMMainFunctionPeriod**	
Description	Specifies the period of main function KeyM_MainFunction in seconds.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.01	
Range	<=4294967295	
	>0.0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMServiceCertificateFunctionEnabled**	
Description	Enables (TRUE) or disables (FALSE) the certificate service function of the key manager. If set to true, the KeyM_ServiceCertificate() function has to be called accordingly.	
Multiplicity	0..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	KeyMBackgroundEnabled	
Description	Enables (TRUE) or disables (FALSE) execution of background processing in the KeyM_MainBackgroundFunction instead of the default KeyM_MainFunction.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.3. KeyMCertificate

Containers included		
Container name	**Multiplicity**	**Description**
KeyMCertificateElement	0..65535	This container contains the certificate element configuration.

Parameters included	
Parameter name	**Multiplicity**
KeyMCertAlgorithmType	1..1
KeyMCertFormatType	1..1
KeyMCertificateId	1..1
KeyMCertificateMaxLength	1..1
KeyMCertificateName	1..1
KeyMCertificateVerifyCallbackNotificationFunc	0..1
KeyMServiceCertificateCallbackNotificationFunc	0..1
KeyMCertCertificateElementRuleRef	0..65535
KeyMCertCsmSignatureVerifyJobRef	1..1
KeyMCertStorageCryptoKeyRef	1..1
KeyMCertUpperHierarchicalCertRef	1..1

Parameter Name	KeyMCertAlgorithmType
Description	Specify in which format the certificate will be provided.
Multiplicity	1..1
Type	ENUMERATION
Default value	RSA

Range	ECC	
	RSA	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertFormatType**	
Description	Specify in which format the certificate will be provided.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	X509	
Range	CRL	
	CVC	
	X509	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertificateId**	
Description	Identifier of the certificate. The set of configured identifiers shall be consecutive and gapless.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=65535	
	>=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertificateMaxLength**	
Description	Specify the maximum length in bytes of the certificate.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertificateName**

Description	Provides a unique name of the certificate for identification. The certificate provisional will reference certificates by this unique name.	
Multiplicity	1..1	
Type	STRING	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertificateVerifyCallbackNotificationFunc**	
Description	This parameter provides the function name for the callback <KeyM_CertificateVerifyCallbackNotification>. It indicates if a certificate verification operation was finished and provides its status. If this parameter is omitted, no callback will be provided.	
Multiplicity	0..1	
Type	FUNCTION-NAME	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMServiceCertificateCallbackNotificationFunc**	
Description	This parameter provides the function name for the service certificate callback <KeyM_ServiceCertificateCallbackNotification>. It indicates if a certificate service operation was finished and provides its status. If this parameter is not set, no callback will be provided.	
Multiplicity	0..1	
Type	FUNCTION-NAME	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertCertificateElementRuleRef**	
Description	Reference to certificate element rules which should be verified within the certification validation step.	
Multiplicity	0..65535	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile

	VariantPreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertCsmSignatureVerifyJobRef**	
Description	Reference to the CSM job that is used to verify the signature	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertStorageCryptoKeyRef**	
Description	Defines a storage location of the certificate.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**KeyMCertUpperHierarchicalCertRef**	
Description	Identifier of the certificate that is the next higher in the PKI hierarchical structure. The reference points to itself for root certificates.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.4. KeyMCertificateElement

Parameters included	
Parameter name	**Multiplicity**
KeyMCertificateElementHasIteration	1..1
KeyMCertificateElementMaxIterations	1..1
KeyMCertificateElementId	1..1
KeyMCertificateElementMaxLength	1..1
KeyMCertificateElementObjectId	0..1

Parameters included	
KeyMCertificateElementObjectType	0..1
KeyMCertificateElementOfStructure	1..1
KeyMCertificateElementIsMandatory	1..1

Parameter Name	**KeyMCertificateElementHasIteration**	
Description	Defines if the certificate element can occur more than one time. If so, the iterator can be used to retrieve the individual data values of this certificate element.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertificateElementMaxIterations**	
Description	Maximum number of iterations.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=255	
	>=2	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPreCompile:	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**KeyMCertificateElementId**	
Description	Identifier of a certificate element.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=65535	
	>=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertificateElementMaxLength**

Description	Maximum length in bytes.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=65535	
	>=1	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertificateElementObjectId**	
Description	EN: This is the object identifier (OID) that is used to identify the certificate element within its element structure. The value is parsed as comma-separated byte values given in hexadecimal representation (uint8 array). E.g. 0x12, 0xab, 0xff would be a valid input.	
Multiplicity	0..1	
Type	STRING	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertificateElementObjectType**	
Description	Certificate elements are stored in ASN.1 format. In this item the type of ASN.-1 TLV can be specified (e.g. INTEGER has the value '2'). This can be used to identify only such certificate elements. If the type is different, the element is not included in the search.	
Multiplicity	0..1	
Type	INTEGER	
Range	<=255	
	>=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertificateElementOfStructure**
Description	This defines in which structure the certificate element is located.
Multiplicity	1..1

Type	ENUMERATION	
Range	CertificateExtension	
	CertificateIssuerName	
	CertificateIssuerUniqueIdentifier	
	CertificateSerialNumber	
	CertificateSignature	
	CertificateSignatureAlgorithm	
	CertificateSignatureAlgorithmID	
	CertificateSubjectName	
	CertificateSubjectPublicKeyInfo_PublicKeyAlgorithm	
	CertificateSubjectPublicKeyInfo_SubjectPublicKey	
	CertificateSubjectUniqueIdentifier	
	CertificateValidityPeriodNotAfter	
	CertificateValidityPeriodNotBefore	
	CertificateVersionNumber	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertificateElementIsMandatory**	
Description	Defines if the certificate element is mandatory.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.5. KeyMCertificateElementVerification

Containers included		
Container name	**Multiplicity**	**Description**
KeyMCertificateElementCon-dition	1..n	This container contains the configuration of KeyElement compare conditions which can be used as arguments for a KeyMCertificateElementRule.

Containers included		
KeyMCertificateElementRule	1..n	This container contains the configuration of a mode rule which represents a logical expression with KeyMCertificateElementCondition or other KeyMCertificateElementRule as arguments.

5.2.1.6. KeyMCertificateElementCondition

Containers included		
Container name	**Multiplicity**	**Description**
KeyMCertificateElementConditionValue	1..1	This container contains the configuration of a compare value.

Parameters included	
Parameter name	**Multiplicity**
KeyMCertElementConditionType	1..1
KeyMCertificateElementRef	0..1

Parameter Name	KeyMCertElementConditionType	
Description	This parameter specifies what kind of comparison that is made for the evaluation of the mode condition.	
Multiplicity	1..1	
Type	ENUMERATION	
Range	KEYM_EQUALS	
	KEYM_EQUALS_NOT	
	KEYM_GREATER_OR_EQUAL	
	KEYM_LESS_OR_EQUAL	
	KEYM_LESS_THAN	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	KeyMCertificateElementRef
Description	Reference to a certificate element used for the condition.
Multiplicity	0..1

Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.7. KeyMCertificateElementConditionValue

Containers included		
Container name	**Multiplicity**	**Description**
KeyMCertificateElementConditionArray	0..1	This container contains the configuration of a array compare value.
KeyMCertificateElementConditionCerificateElement	0..1	This container contains the configuration of a certificate element as a compare value.
KeyMCertificateElementConditionPrimitive	0..1	This container contains the configuration of a primitive compare value.
KeyMCertificateElementConditionSenderReceiver	0..1	This container contains the configuration of a dynamic compare value in a sender-/receiver interface.

5.2.1.8. KeyMCertificateElementConditionArray

Containers included		
Container name	**Multiplicity**	**Description**
KeyMCertificateElementConditionArrayElement	0..n	This container contains the configuration of a array compare value.

5.2.1.9. KeyMCertificateElementConditionArrayElement

Parameters included	
Parameter name	**Multiplicity**
KeyMCertificateElementConditionArrayElementIndex	1..1
KeyMCertificateElementConditionArrayElementValue	1..1

Parameter Name	**KeyMCertificateElementConditionArrayElementIndex**

Description	Index to an element of the compare value array.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCertificateElementConditionArrayElementValue**	
Description	Value of an array element compare value.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=255	
	>=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.10. KeyMCertificateElementConditionCerificateElement

Parameters included	
Parameter name	**Multiplicity**
KeyMCertificateElementRef	1..1

Parameter Name	**KeyMCertificateElementRef**	
Description	Reference to another certificate element.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.11. KeyMCertificateElementConditionPrimitive

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
KeyMCertificateElementConditionPrimitiveValue	1..1

Parameter Name	KeyMCertificateElementConditionPrimitiveValue	
Description	Primitive compare value	
Multiplicity	1..1	
Type	INTEGER	
Range	<=255	
	>=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.12. KeyMCertificateElementConditionSenderReceiver

Containers included		
Container name	**Multiplicity**	**Description**
KeyMCertificateElementConditionSenderReceiver	1..1	This parameter references a mode in a particular mode request port of a software component that is used for the condition.

5.2.1.13. KeyMCertificateElementConditionSenderReceiver

Parameters included	
Parameter name	**Multiplicity**
TARGET	1..1
CONTEXT	0..n

Parameter Name	TARGET
Multiplicity	1..1
Type	REFERENCE
Origin	AUTOSAR_ECUC

Parameter Name	CONTEXT

Multiplicity	0..n
Type	REFERENCE
Range	ROOT-SW-COMPOSITION-PROTOTYPE SW-COMPONENT-PROTOTYPE PORT-PROTOTYPE
Origin	AUTOSAR_ECUC

5.2.1.14. KeyMCertificateElementRule

Parameters included	
Parameter name	**Multiplicity**
KeyMLogicalOperator	0..1
KeyMArgumentRef	1..n

Parameter Name	**KeyMLogicalOperator**	
Description	This parameter specifies the logical operator to be used in the logical expression. If the expression only consists of a single condition this parameter shall not be used.	
Multiplicity	0..1	
Type	ENUMERATION	
Range	KEYM_AND KEYM_OR	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMArgumentRef**	
Description	This is a choice reference either to a condition or another rule serving as sub-expression.	
Multiplicity	1..n	
Type	CHOICE-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPreCompile:	VariantPreCompile

Origin	AUTOSAR_ECUC

5.2.1.15. KeyMCryptoKey

Containers included		
Container name	**Multiplicity**	**Description**
KeyMNvmBlock	1..1	Configuration of optional usage of Nvm in case the KeyM module requires non volatile memory in the Ecu to store information (e.g. crypto keys or certificates).

Parameters included	
Parameter name	**Multiplicity**
KeyMCryptoKeyGenerationType	1..1
KeyMCryptoKeyId	1..1
KeyMCryptoKeyMaxLength	1..1
KeyMCryptoKeyStorage	1..1

Parameter Name	**KeyMCryptoKeyGenerationType**	
Description	Specifies how the CryptoKey will be generated. If it is derived from another key or simply stored with KeyElementSet.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	KEYM_STORED_KEY	
Range	KEYM_DERIVED_KEY	
	KEYM_STORED_KEY	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCryptoKeyId**
Description	Identifier of the crypto key. The set of configured identifiers shall be consecutive and gapless.
Multiplicity	1..1
Type	INTEGER

Range	<=65535	
	>=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCryptoKeyMaxLength**	
Description	The maximum size in bytes of a CryptoKey.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=4294967295	
	>=1	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**KeyMCryptoKeyStorage**	
Description	Specify the storage location of the certificate.	
Multiplicity	1..1	
Type	ENUMERATION	
Range	KEYM_STORAGE_IN_CSM	
	KEYM_STORAGE_IN_NVM	
	KEYM_STORAGE_IN_RAM	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.16. KeyMNvmBlock

Parameters included	
Parameter name	**Multiplicity**
KeyMNvmBlockDescriptorRef	1..1

Parameter Name	**KeyMNvmBlockDescriptorRef**
Description	Reference to the Nvm block description in the Nvm module configuration.

Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.17. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	**PbcfgMSupport**	
Label	PbcfgM support	
Description	Specifies whether or not the KeyM can use the PbcfgM module for post-build support.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PublishedInformation:**	
Origin	Elektrobit Automotive GmbH	

5.2.2. Application programming interface (API)

5.2.2.1. Macro constants

5.2.2.1.1. KEYM_E_CSMKEYELEMENTSET_FAILED

Purpose	Development Error to be raised if Csm_KeyElementSet failed.
Value	0xE0

5.2.2.1.2. KEYM_E_INIT_FAILED

Purpose	Development Error to be raised if initialization of KeyM module failed.
Value	4U

5.2.2.1.3. KEYM_E_PARAM_POINTER

Purpose	Development Error to be raised if API request called with invalid parameter. (Nullpointer).
Value	1U

5.2.2.1.4. KEYM_E_SMALL_BUFFER

Purpose	Development Error to be raised if API request called with provided output buffer is to small to store the requested result.
Value	2U

5.2.2.1.5. KEYM_E_UNINIT

Purpose	Development Error to be raised if API request called before initialization of KeyM module.
Value	3U

5.2.2.1.6. KEYM_INSTANCE_ID

Purpose	The index of the KeyM instance. As the KeyM is a single instance module it is always 0.
Value	0U

5.2.2.1.7. KEYM_SID_CERTELEMENTGET

Purpose	The 'KeyM_CertElementGet' API service identifier.

Value	15U

5.2.2.1.8. KEYM_SID_CERTELEMENTGETFIRST

Purpose	The 'KeyM_CertElementGetFirst' API service identifier.
Value	16U

5.2.2.1.9. KEYM_SID_CERTELEMENTGETNEXT

Purpose	The 'KeyM_CertElementGetNext' API service identifier.
Value	17U

5.2.2.1.10. KEYM_SID_CERTGETSTATUS

Purpose	The 'KeyM_CertGetStatus' API service identifier.
Value	18U

5.2.2.1.11. KEYM_SID_DEINIT

Purpose	The 'KeyM_Deinit' API service identifier.
Value	2U

5.2.2.1.12. KEYM_SID_FINALIZE

Purpose	The 'KeyM_Finalize' API service identifier.
Value	7U

5.2.2.1.13. KEYM_SID_GETCERTIFICATE

Purpose	The 'KeyM_GetCertificate' API service identifier.

Value	11U

5.2.2.1.14. KEYM_SID_GETVERSIONINFO

Purpose	The 'KeyM_GetVersionInfo' API service identifier.
Value	3U

5.2.2.1.15. KEYM_SID_INIT

Purpose	The 'KeyM_Init' API service identifier.
Value	1U

5.2.2.1.16. KEYM_SID_MAINBACKGROUNDFUNCTION

Purpose	The 'KeyM_MainBackgroundFunction' API service identifier.
Value	26U

5.2.2.1.17. KEYM_SID_MAINFUNCTION

Purpose	The 'KeyM_MainFunction' API service identifier.
Value	25U

5.2.2.1.18. KEYM_SID_PREPARE

Purpose	The 'KeyM_Prepare' API service identifier.
Value	5U

5.2.2.1.19. KEYM_SID_SERVICECERTIFICATE

Purpose	The 'KeyM_ServiceCertificate' API service identifier.

Value	9U

5.2.2.1.20. KEYM_SID_SETCERTIFICATE

Purpose	The 'KeyM_SetCertificate' API service identifier.
Value	10U

5.2.2.1.21. KEYM_SID_START

Purpose	The 'KeyM_Start' API service identifier.
Value	4U

5.2.2.1.22. KEYM_SID_UPDATE

Purpose	The 'KeyM_Update' API service identifier.
Value	6U

5.2.2.1.23. KEYM_SID_VERIFY

Purpose	The 'KeyM_Verify' API service identifier.
Value	8U

5.2.2.1.24. KEYM_SID_VERIFYCERTIFICATE

Purpose	The 'KeyM_VerifyCertificate' API service identifier.
Value	13U

5.2.2.1.25. KEYM_SID_VERIFYCERTIFICATECHAIN

Purpose	The 'KeyM_VerifyCertificateChain' API service identifier.

Value	14U

5.2.2.1.26. KEYM_SID_VERIFYCERTIFICATES

Purpose	The 'KeyM_VerifyCertificates' API service identifier.
Value	12U

5.2.2.2. Functions

5.2.2.2.1. KeyM_CertElementGet

Purpose	Provides the content of a specific certificate element. The certificate configuration defines how the certificate submodule can find the element, e.g. by providing the object identifier (OID). This function is used to retrieve this information if only one element is assigned to the respective OID.	
Synopsis	`Std_ReturnType` **`KeyM_CertElementGet`** `( KeyM_CertificateIdType CertId , KeyM_CertElementIdType CertElementId , uint8 * CertElementData , uint32 * CertElementDataLength );`	
Service ID	KEYM_SID_CERTELEMENTGET	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`CertId`	Holds the identifier of the last certificate in the chain.
	`CertElementId`	Specifies the ElementId where the data shall be read from.
Parameters (in,out)	`CertElementDataLength`	In: Pointer to a value that contains the maximum data length of the CertElementData buffer. Out: The data length will be overwritten with the actual length of data placed to the buffer if the function returns E_OK. Otherwise, the it will be overwritten with the value zero.
Parameters (out)	`CertElementData`	Pointer to a data buffer allocated by the caller of this function. If available, the function returns E_OK and copies the data into this buffer.

Return Value	Error value.	
	`E_OK`	Element found and data provided in the buffer
	`E_NOT_OK`	Element data not found
	`KEYM_E_PARAMETER_MISMATCH`	Certificate ID or certificate element ID invalid
	`KEYM_E_KEY_CERT_SIZE_MISMATCH`	Provided buffer for the certificate element too small
	`KEYM_E_KEY_CERT_EMPTY`	No certificate data available, the certificate slot is empty
	`KEYM_E_KEY_CERT_INVALID`	The certificate is not valid or has not yet been verified

5.2.2.2.2. KeyM_CertElementGetFirst

Purpose	This function is used to initialize the interative extraction of a certificate data element. It always retrieves the top element from the configured certificate element and initializes the structure KeyM_CertElementIterator so that consecutive data from this element can be read with KeyM_CertElementGetNext().	
Synopsis	`Std_ReturnType` **`KeyM_CertElementGetFirst`** `( KeyM_CertificateIdType CertId , KeyM_CertElementIdType CertElementId , KeyM_CertElementIteratorType * CertElementIterator , uint8 * CertElementData , uint32 * CertElementDataLength );`	
Service ID	KEYM_SID_CERTELEMENTGETFIRST	
Sync/Async	Synchronous	
Reentrancy	Reentrant Reentrant for one iterator.	
Parameters (in)	`CertId`	Holds the identifier of the last certificate in the chain.
	`CertElementId`	Specifies the CertElementId where the data shall be read from.
Parameters (in,out)	`CertElementIterator`	Pointer to a structure that is allocated and maintained by the caller. It shall not be destroyed or altered by the application until all elements have been retrieved through KeyM_CertElementGetNext().
	`CertElementDataLength`	In: Pointer to a value that contains the maximum data length of the CertElement-

		Data buffer. Out: The data length will be overwritten with the actual length of data placed to the buffer if the function returns E_OK.
Parameters (out)	`CertElementData`	Pointer to a data buffer allocated by the caller of this function. If available, the function returns E_OK and copies the data into this buffer.
Return Value	Error value.	
	`E_OK`	Element found and data provided in the buffer. The certElementIterator has been initialized accordingly
	`E_NOT_OK`	Element data not found. CertElementIterator cannot be used for further calls
	`KEYM_E_PARAMETER_MISMATCH`	Certificate ID or certificate element ID invalid
	`KEYM_E_KEY_CERT_SIZE_MISMATCH`	Provided buffer for the certificate element too small
	`KEYM_E_KEY_CERT_EMPTY`	No certificate data available, the certificate is empty
	`KEYM_E_CERT_INVALID`	Certificate is not valid or not verified successfully

5.2.2.2.3. KeyM_CertElementGetNext

Purpose	This function provides further data from a certificate element, e.g. if a set of data are located in one certificate element that shall be read one after another. This function can only be called if the function KeyM_CertElementGetFirst() has been called once before.	
Synopsis	`Std_ReturnType` **`KeyM_CertElementGetNext`** `( KeyM_CertElementIteratorType * CertElementIterator , uint8 * CertElementData , uint32 * CertElementDataLength );`	
Service ID	KEYM_SID_CERTELEMENTGETNEXT	
Sync/Async	Synchronous	
Reentrancy	Reentrant Reentrant for one iterator.	
Parameters (in,out)	`CertElementIterator`	Pointer to a structure that is allocated by the caller and used by the function. It shall

		not be destroyed or altered by the application until all elements have been read from the list.
	`CertElementDataLength`	In: Pointer to a value that contains the maximum data length of the CertElementData buffer. Out: The data length will be overwritten with the actual length of data placed to the buffer if the function returns E_OK.
Parameters (out)	`CertElementData`	Pointer to a data buffer allocated by the caller of this function. If available, the function returns E_OK and copies the data into this buffer.
Return Value	Error value.	
	`E_OK`	Element found and data provided in the buffer. The CertElementIterator has been initialized accordingly
	`E_NOT_OK`	Element data not found. CertElementIterator cannot be used for further calls
	`KEYM_E_PARAMETER_MISMATCH`	Certificate ID or certificate element ID invalid
	`KEYM_E_KEY_CERT_SIZE_MISMATCH`	Provided buffer for the certificate element too small
	`KEYM_E_KEY_CERT_EMPTY`	No certificate data available, the certificate is empty
	`KEYM_E_CERT_INVALID`	Certificate is not valid or not verified successfully

5.2.2.2.4. KeyM_CertGetStatus

Purpose	This function provides the status of a certificate.	
Synopsis	`Std_ReturnType` **`KeyM_CertGetStatus`** `( KeyM_CertificateIdType CertId , KeyM_CertificateStatusType * Status );`	
Service ID	KEYM_SID_CERTGETSTATUS	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`CertId`	Holds the identifier of the certificate.

Parameters (out)	`Status`	Provides the status of the certificate.
Return Value	Error value.	

5.2.2.2.5. KeyM_CertificateVerifyCallbackNotification

Purpose	Notifies the application that a certificate verification has been finished. The function name is configurable by KeyMCertificateVerifyCallbackNotificationFunc.	
Synopsis	`Std_ReturnType` **`KeyM_CertificateVerifyCallbackNotification`** `( KeyM_CertificateIdType CertId , KeyM_CertificateStatusType Result );`	
Parameters (in)	`CertId`	The certificate identifier that has been verified.
	`Result`	Contains information about the result of the operation.
Return Value	Error value.	
	`E_OK`	

5.2.2.2.6. KeyM_Deinit

Purpose	This function resets the key management module to the uninitialized state.
Synopsis	`void` **`KeyM_Deinit`** `( void );`
Service ID	KEYM_SID_DEINIT
Sync/Async	Synchronous
Reentrancy	Non Reentrant

5.2.2.2.7. KeyM_Finalize

Purpose	The function is used to finalize key update operations. It is typically used in conjunction with the KeyM_Start operation and returns the key operation into the idle mode. Further key prepare or update operations are not accepted until a new KeyM_Start operation has been initialized. This function is only available if KeyMCryptoKeyStartFinalizeFunctionEnabled is set to TRUE. In addition, updated key material will be persisted and set into valid state (calling Csm_KeySetValid).
Synopsis	`Std_ReturnType` **`KeyM_Finalize`** `( const uint8 * RequestDataPtr , uint16 RequestDataLength , uint8 * ResponseDataPtr , uint16 * ResponseMaxDataLength );`

Service ID	KEYM_SID_FINALIZE	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`RequestDataPtr`	Information that comes along with the request.
	`RequestDataLength`	Length of data in the RequestData array.
Parameters (in,out)	`ResponseMaxDataLength`	In: Max number of bytes available in ResponseData Out: Actual number of bytes in ResponseData or left untouched if service runs in asynchronous mode and function returns KEYM_E_OK.
Parameters (out)	`ResponseDataPtr`	Data returned by the function.
Return Value	Error value.	
	`E_OK`	Operation has been accepted and will be processed internally. Results will be provided through a callback
	`E_NOT_OK`	Operation not accepted due to an internal error
	`KEYM_E_BUSY`	Validation cannot be performed yet. KeyM is currently busy with other jobs
	`KEYM_E_PARAMETER_MISMATCH`	Parameter do not match with expected value
	`KEYM_E_KEY_CERT_SIZE_MISMATCH`	Parameter size doesn't match

5.2.2.2.8. KeyM_GetCertificate

Purpose	This function provides the certificate data.	
Synopsis	`Std_ReturnType` **`KeyM_GetCertificate`** `( KeyM_CertificateIdType CertId , KeyM_CertDataType * CertificateDataPtr );`	
Service ID	KEYM_SID_GETCERTIFICATE	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`CertId`	Holds the identifier of the certificate.
Parameters (in,out)	`CertificateDataPtr`	Provides a pointer to a certificate data structure. The buffer located by the pointer in the structure shall be provided by the

		caller of this function. The length information indicates the maximum length of the buffer when the function is called. If E_-OK is returned, the length information indicates the actual length of the certificate data in the buffer.
Return Value	Error value.	
	`E_OK`	Certificate data available and provided.
	`E_NOT_OK`	Operation not accepted due to an internal error
	`KEYM_E_PARAMETER_MISMATCH`	Certificate ID invalid
	`KEYM_E_KEY_CERT_SIZE_MISMATCH`	Provided buffer for the certificate too small
	`KEYM_E_KEY_CERT_EMPTY`	No certificate data available, the certificate slot is empty
	`KEYM_E_KEY_CERT_READ_FAIL`	Certificate cannot be provided, access denied

5.2.2.2.9. KeyM_GetVersionInfo

Purpose	Provides the version information of this module.	
Synopsis	`void` **`KeyM_GetVersionInfo`** `( Std_VersionInfoType * VersionInfo );`	
Service ID	KEYM_SID_GETVERSIONINFO	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (out)	`VersionInfo`	Pointer to the version information of this module.

5.2.2.2.10. KeyM_Init

Purpose	This function initializes the key management module.
Synopsis	`void` **`KeyM_Init`** `( const KeyM_ConfigType * ConfigPtr );`
Service ID	KEYM_SID_INIT
Sync/Async	Synchronous
Reentrancy	Non Reentrant

Parameters (in)	`ConfigPtr`	Pointer to the configuration set in VARIANT-POST-BUILD.

5.2.2.2.11. KeyM_MainBackgroundFunction

Purpose	Function is called from a pre-emptive operating system when no other task operation is needed. Can be used for calling time consuming synchronous functions such as KeyM_KH_Update().
Synopsis	`void` **`KeyM_MainBackgroundFunction`** `( void );`
Service ID	KEYM_SID_MAINBACKGROUNDFUNCTION
Sync/Async	Synchronous
Reentrancy	Non Reentrant

5.2.2.2.12. KeyM_MainFunction

Purpose	Function is called periodically according the specified time interval.
Synopsis	`void` **`KeyM_MainFunction`** `( void );`
Service ID	KEYM_SID_MAINFUNCTION
Sync/Async	Synchronous
Reentrancy	Non Reentrant

5.2.2.2.13. KeyM_Prepare

Purpose	This function is used to prepare a key update operation. The main intent is to provide information for the key operation to the key server. Other operations may start the negotiation for a common secret that is used further to derive key material. This function is only available if KeyMCryptoKeyPrepareFunctionEnabled is set to TRUE.	
Synopsis	`Std_ReturnType` **`KeyM_Prepare`** `( const uint8 * RequestData , uint16 RequestDataLength , uint8 * ResponseData , uint16 * ResponseDataLength );`	
Service ID	KEYM_SID_PREPARE	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`RequestData`	Information that comes along with the request.

	`RequestDataLength`	Length of data in the RequestData array.
Parameters (in,out)	`ResponseDataLength`	In: Max number of bytes available in ResponseData Out: Actual number of bytes.
Parameters (out)	`ResponseData`	Data returned by the function.
Return Value	Error value.	
	`E_OK`	Service has been accepted and will be processed internally. Results will be provided through a callback
	`E_NOT_OK`	Service not accepted due to an internal error
	`KEYM_E_PARAMETER_MISMATCH`	Parameter do not match with expected value
	`KEYM_E_KEY_CERT_SIZE_MISMATCH`	Parameter size doesn't match

5.2.2.2.14. KeyM_ServiceCertificate

Purpose	The key server requests an operation from the key client. The type of operation is specified in the first parameter KeyM_ServiceCertificateType. Certificate operation requests are operated through this function. This function is only available if the configuration parameter KeyMServiceCertificateFunctionEnabled is set to TRUE.	
Synopsis	`Std_ReturnType` **`KeyM_ServiceCertificate`** `( KeyM_ServiceCertificateType Service , const uint8 * CertNamePtr , uint16 CertNameLength , const uint8 * RequestData , uint16 RequestDataLength , uint8 * ResponseData , uint16 ResponseDataLength );`	
Service ID	KEYM_SID_SERVICECERTIFICATE	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`Service`	Provides the type of service the key manager has to perform.
	`CertNamePtr`	Points to an array that defines the name of the certificate to be updated.
	`CertNameLength`	Specifies the number of bytes in CertNamePtr. The value 0 indicates that no CertNamePtr is provided within this function.
	`RequestData`	Information that comes along with the request.

	`RequestDataLength`	Length of data in the RequestData array.
	`ResponseDataLength`	Max number of bytes available in ResponseDataPtr.
Parameters (out)	`ResponseData`	Data returned by the function.
Return Value	Error value.	
	`E_OK`	Service data operation successfully accepted
	`E_NOT_OK`	Operation not accepted due to an internal error
	`KEYM_E_PARAMETER_MISMATCH`	Parameter do not match with expected value
	`KEYM_E_KEY_CERT_SIZE_MISMATCH`	Parameter size doesn't match

5.2.2.2.15. KeyM_ServiceCertificateCallbackNotification

Purpose	Notifies the application that the certificate service operation has been finished. This function is used by the certificate submodule. This callback is only provided if KeyMServiceCertificateFunctionEnabled is set to TRUE. The function name is configurable by KeyMServiceCertificateCallbackNotificationFunc.	
Synopsis	`void` **`KeyM_ServiceCertificateCallbackNotification`** `( KeyM_CertificateIdType CertId , KeyM_ResultType Result , uint16 ResultDataLength , uint8 * ResultDataPtr );`	
Parameters (in)	`CertId`	The certificate identifier where this service was started for.
	`Result`	Contains information about the result of the operation.
	`ResultDataLength`	Contains the length of the resulting data of this operation if any.
	`ResultDataPtr`	Pointer to the data of the result.

5.2.2.2.16. KeyM_SetCertificate

Purpose	This function provides the certificate data to the key management module to temporarily store the certificate.
Synopsis	`Std_ReturnType` **`KeyM_SetCertificate`** `( KeyM_CertificateIdType CertId , const KeyM_CertDataType * CertificateDataPtr );`

Service ID	KEYM_SID_SETCERTIFICATE	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`CertId`	Holds the identifier of the certificate.
	`CertificateDataPtr`	Pointer to a structure that provides the certificate data.
Return Value	Error value.	
	`E_OK`	Certificate accepted
	`E_NOT_OK`	Certificate could not be set
	`KEYM_E_PARAMETER_MISMATCH`	Parameter do not match with expected value
	`KEYM_E_KEY_CERT_SIZE_MISMATCH`	Parameter size doesn't match

5.2.2.2.17. KeyM_Start

Purpose	This function is optional and only used if the configuration item KeyMCryptoKeyStart-FinalizeFunctionEnabled is set to true. It intents to allow key update operation.	
Synopsis	`Std_ReturnType` **`KeyM_Start`** `( KeyM_StartType StartType , const uint8 * RequestData , uint16 RequestDataLength , uint8 * Re-sponseData , uint16 * ResponseDataLength );`	
Service ID	KEYM_SID_START	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`StartType`	Defines in which mode the key operation shall be executed.
	`RequestData`	Information that comes along with the re-quest, e.g. signature.
	`RequestDataLength`	Length of data in the RequestData array.
Parameters (in,out)	`ResponseDataLength`	In: Max number of bytes available in Re-sponseData Out: Actual number.
Parameters (out)	`ResponseData`	Data returned by the function.
Return Value	Error value.	
	`E_OK`	Start operation successfully performed. Key update operations are now allowed
	`E_NOT_OK`	Start operation not accepted

	`KEYM_E_PARAMETER_MISMATCH`	Parameter do not match with expected value
	`KEYM_E_KEY_CERT_SIZE_MISMATCH`	Parameter size doesn't match

5.2.2.2.18. KeyM_Update

Purpose	This function is used to initiate the key generation or update process.	
Synopsis	`Std_ReturnType` **`KeyM_Update`** `( const uint8 * KeyNamePtr , uint16 KeyNameLength , const uint8 * RequestDataPtr , uint16 Request-DataLength , uint8 * ResultDataPtr , uint16 ResultDataMaxLength );`	
Service ID	KEYM_SID_UPDATE	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`KeyNamePtr`	Pointer to an array that defines the name of the key to be updated.
	`KeyNameLength`	Specifies the number of bytes in key-Name. The value 0 indicates that no key-Name is provided within this function.
	`RequestDataPtr`	Information that comes along with the re-quest.
	`RequestDataLength`	Length of data in the RequestData array.
	`ResultDataMaxLength`	Max number of bytes available in Result-DataPtr.
Parameters (out)	`ResultDataPtr`	Pointer to a data buffer used by the func-tion to store results.
Return Value	Error value.	
	`E_OK`	Service has been accepted and will be processed internally. Results will be pro-vided through a callback
	`E_NOT_OK`	Service not accepted due to an internal error
	`E_BUSY`	Service could not be accepted because another operation is already ongoing. Try next time
	`KEYM_E_PARAMETER_MISMATCH`	Parameter do not match with expected value

	`KEYM_E_KEY_CERT_SIZE_MISMATCH`	Parameter size doesn't match

5.2.2.2.19. KeyM_Verify

Purpose	The key server requests to verify the provided keys. The key manager performs operation on the assigned job and returns the result to the key server who verifies if the results was provided with this key as expected. This function is only available if KeyMCryptoKeyVerifyFunctionEnabled is set to TRUE.	
Synopsis	`Std_ReturnType` **`KeyM_Verify`** `( const uint8 * KeyNamePtr , uint16 KeyNameLength , const uint8 * RequestData , uint16 RequestDataLength , uint8 * ResponseData , uint16 * ResponseDataLength );`	
Service ID	KEYM_SID_VERIFY	
Sync/Async	Synchronous Synchronous/Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`KeyNamePtr`	Points to an array that defines the name of the key to be updated.
	`KeyNameLength`	Specifies the number of bytes in KeyNamePtr. The value 0 indicates that no KeyNamePtr is provided within this function.
	`RequestData`	Information that comes along with the request.
	`RequestDataLength`	Length of data in the RequestData array.
Parameters (in,out)	`ResponseDataLength`	In: Max number of bytes available in ResponseData Out: Actual number of bytes in ResponseData or left untouched if service runs in asynchronous mode and function returns KEYM_E_PENDING.
Parameters (out)	`ResponseData`	Data returned by the function.
Return Value	Error value.	
	`KEYM_E_PENDING`	Operation runs in asynchronous mode, has been accepted and will be processed internally. Results will be provided through callback
	`E_OK`	Operation was successfully performed. Result information are available

	E_NOT_OK	Operation not accepted due to an internal error
	KEYM_E_BUSY	Validation cannot be performed yet. KeyM is currently busy with other jobs (for asynchronous mode)
	KEYM_E_PARAMETER_MISMATCH	Parameter do not match with expected value
	KEYM_E_KEY_CERT_SIZE_MISMATCH	Parameter size doesn't match
	KEYM_E_KEY_CERT_INVALID	Key operation cannot be performed because the key name is invalid
	KEYM_E_KEY_CERT_EMPTY	The key for this slot has not been set

5.2.2.2.20. KeyM_VerifyCertificate

Purpose	This function verifies a certificate that was previously provided with KeyM_SetCertificate() against already stored and provided certificates stored with other certificate IDs.	
Synopsis	`Std_ReturnType **KeyM_VerifyCertificate** ( KeyM_CertificateIdType CertId );`	
Service ID	KEYM_SID_VERIFYCERTIFICATE	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CertId	Holds the identifier of the certificate.
Return Value	Error value.	
	E_OK	Certificate verification request accepted. Operation will be performed in the background and response is given through a callback
	E_NOT_OK	Operation not accepted due to an internal error
	KEYM_E_BUSY	Validation cannot be performed yet. KeyM is currently busy with other jobs
	KEYM_E_PARAMETER_MISMATCH	Certificate ID invalid
	KEYM_E_KEY_CERT_EMPTY	One of the certificate slots are empty
	KEYM_E_CERT_INVALID_CHAIN_OF_TRUST	An upper certificate is not valid

5.2.2.2.21. KeyM_VerifyCertificateChain

Purpose	This function performs a certificate verification against a list of certificates. It is a pre-requisite that the certificate that shall be checked has already been written with KeyM_SetCertificate() and that the root certificate is either in the list or is already assigned to one of the other certificates.	
Synopsis	`Std_ReturnType` **`KeyM_VerifyCertificateChain`** `( KeyM_CertificateIdType CertId , KeyM_CertDataType *const certChainData , uint8 NumberOfCertificates );`	
Service ID	KEYM_SID_VERIFYCERTIFICATECHAIN	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`CertId`	Holds the identifier of the last certificate in the chain.
	`certChainData`	This is a pointer to an array of certificates sorted according to the order in the PKI.
	`NumberOfCertificates`	Defines the number of certificates stored in the CertChainData array.
Return Value	Error value.	
	`E_OK`	Certificate verification request accepted. Operation will be performed in the background and response is given through a callback
	`E_NOT_OK`	Operation not accepted due to an internal error
	`KEYM_E_BUSY`	Validation cannot be performed yet. KeyM is currently busy with other jobs
	`KEYM_E_PARAMETER_MISMATCH`	Certificate ID invalid
	`KEYM_E_KEY_CERT_EMPTY`	One of the certificate slots are empty
	`KEYM_E_CERT_INVALID_CHAIN_OF_TRUST`	An upper certificate is not valid

5.2.2.2.22. KeyM_VerifyCertificates

Purpose	This function verifies two certificates that are stored and parsed internally against each other. The certificate referenced with CertId was signed by the certificate referenced with certUpperId. Only these two certificates are validated against each other.

Synopsis	`Std_ReturnType` **`KeyM_VerifyCertificates`** `( KeyM_CertificateIdType CertId , KeyM_CertificateIdType CertUpperId );`	
Service ID	KEYM_SID_VERIFYCERTIFICATES	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`CertId`	Holds the identifier of the lower certificate in the chain.
	`CertUpperId`	Holds the identifier of the upper certificate in the chain.
Return Value	Error value.	
	`E_OK`	Certificate verification request accepted. Operation will be performed in the back-ground and response is given through a callback
	`E_NOT_OK`	Operation not accepted due to an internal error
	`KEYM_E_BUSY`	Validation cannot be performed yet. KeyM is currently busy with other jobs
	`KEYM_E_PARAMETER_MISMATCH`	Certificate ID invalid
	`KEYM_E_KEY_CERT_EMPTY`	One of the certificate slots are empty
	`KEYM_E_CERT_INVALID_CHAIN_OF_-TRUST`	An upper certificate is not valid

5.2.3. Integration notes

5.2.3.1. Exclusive areas

This section describes the exclusive areas used by the `KeyM` module.

5.2.3.1.1. SCHM_KEYM_EXCLUSIVE_AREA_0

Protected data structures	All shared data that shall be protected from mutual access.
Recommended locking mechanism	This exclusive area must always be protected by a locking mechanism. The options for locking are described in the `EB`

	tresos AutoCore Generic documentation. Refer to the section Mapping exclusive areas in the basic software modules in the Integration notes section for details.

5.2.3.2. Production errors

Production errors information is not available for this module.

5.2.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section Memory mapping and compiler abstraction in the Integration notes section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
CODE
CONST_UNSPECIFIED
CONST_8
VAR_INIT_8
VAR_INIT_BOOLEAN
VAR_INIT_UNSPECIFIED

5.2.3.4. Integration requirements

WARNING

Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

5.2.3.4.1. KeyM.Req.Integration_KeyMInit

Description	KeyM_Init() shall be called during the start-up procedure of the ECU before any other API of the module is called.

5.2.3.4.2. KeyM.Req.Integration_KeyMDeinit

Description	KeyM_Deinit() shall be called during the shutdown procedure of the ECU.

5.2.3.4.3. KeyM.Req.Integration_StartupNvMRead

Description	If the use of NvM storage is enabled by setting the parameter KeyMCryptoKeyStorage of at least one KeyMCryptoKey to KEYM_STORAGE_IN_NVM, the KeyM_Init() shall be called after the NvM module is initialized.

5.2.3.4.4. KeyM.Req.Integration_StartupDet

Description	If the use of Det is enabled by setting the parameter KeyMDevErrorDetect, the KeyM_Init() shall be called after the Det module is initialized.

5.2.3.4.5. KeyM.Req.Integration_StartupCsm

Description	If the Csm is used, e.g. for verifying the signature parsed from a certificate, the KeyM_Init() shall be called after the Csm module is initialized.

5.2.3.4.6. KeyM.Req.Integration_MainFunc

Description	If the KeyM_MainFunction() shall be used instead of the KeyM_MainBackground-Function(), i.e. KeyMBackgroundEnabled has been configured with FALSE, it shall be configured for cyclic execution according to the configured KeyMMainFunctionPeriod, e.g. by mapping it to an appropriate task in the Rte configuration.

5.2.3.4.7. KeyM.Req.Integration_MainBackgroundFunc

Description	If the KeyM_MainBackgroundFunction() shall be used instead of the default KeyM_-MainFunction(), i.e. KeyMBackgroundEnabled has been configured with TRUE, it shall be configured for execution in a background task that only is executed when the CPU is idle, e.g. by using the Rte configuration.

5.2.3.4.8. KeyM.Req.Integration_CsmSignatureVerifyJob

Description	The Csm job referenced via parameter KeyMCertCsmSignatureVerifyJobRef shall be a Csm job referencing a primitive CsmSignatureVerify.

5.2.3.4.9. KeyM.Req.Integration_CsmSignatureVerifyKey

Description	The Csm job referenced via parameter KeyMCertCsmSignatureVerifyJobRef shall be a Csm job referencing a Csm key containing the key element CRYPTO_KE_SIGNATURE_KEY.

5.2.3.4.10. KeyM.Req.Integration_CsmSignatureVerifyCbk

Description	If the Csm job referenced via parameter KeyMCertCsmSignatureVerifyJobRef is asynchronous, it shall reference a Csm callback with enabled parameter CsmCallbackFunc set to the provided KeyM function KeyM_CsmSignatureVerifyCallback.

5.2.3.4.11. KeyM.Req.Integration_CsmSignatureVerifyCbk_OtherVendor

Description	The KeyM provided callback function KeyM_CsmSignatureVerifyCallback() considers the EB deviation to SWS_Csm_00970. This means that the API parameter of KeyM_CsmSignatureVerifyCallback() differ from the Csm SWS. If the KeyM is used with a Csm from a different vendor, an appropriate callback function, with the according API, has to be provided. And it has to be ensured that the Csm is configured for this particular callback function instead of the KeyM_CsmSignatureVerifyCallback function. See also https://bugzilla.autosar.org/show_bug.cgi?id=76940.

5.2.3.4.12. KeyM.Req.Integration_CsmJobPublicKey

Description	When calling KeyM_VerifyCertificate(), if the certificate has been successfully verified, the parsed public key is stored into the Csm job, that is referenced via parameter KeyMCertCsmSignatureVerifyJobRef, by calling Csm_KeyElementSet(). If an error occurs while writing, there is no possibility to inform the application about it. The particular certificate still has a status KEYM_CERTIFICATE_VALID. The parsed public key can be retrieved by KeyM_CertElementGet() correctly. But the key element CRYPTO_KE_SIGNATURE_KEY in the corresponding Csm job is not correct. This does not affect the KeyM, as the next verification request will redo the verification process. But if this key element shall be used in another context, e.g. inside the Crypto Stack, then this issue shall be considered. A solution could be to compare the re-

	trieved parsed public key against the content of the related key element before using it.

5.2.3.4.13. KeyM.Req.Integration_KeyMCertStorageCryptoKeyRef

Description	The KeyM crypto key referenced via parameter KeyMCertStorageCryptoKeyRef shall be an individual KeyM crypto key only referenced in one KeyM certificate.

5.2.3.4.14. KeyM.Req.Integration_KeyMCryptoKeyMaxLength

Description	If the parameter KeyMCryptoKeyStorage is set to KEYM_STORAGE_IN_NVM, the value of parameter KeyMCryptoKeyMaxLength shall have the same size as the configured NvM block referenced via KeyMNvmBlockDescriptorRef.

5.2.3.4.15. KeyM.Req.Integration_KeyMNvMMaxRetries

Description	If the parameter KeyMCryptoKeyStorage is set to KEYM_STORAGE_IN_NVM, the configured NvM block referenced via KeyMNvmBlockDescriptorRef shall be configured with a maximum number of read (NvMMaxNumOfReadRetries) and write (NvMMaxNumOfWriteRetries) retries in the NvM. The KeyM polls the error status after requesting a NvM read or write operation and awaits feedback to continue processing.

5.3. Tls

5.3.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

Containers included		
TlsDefensiveProgramming	1..1	**Label:** Defensive Programming Options Parameters for defensive programming
TlsGeneral	1..1	**Label:** TlsGeneral Container for incorporation of TlsGeneral.
TlsExtensionsDetectionCon-fig	0..1	Specifies the ExtensionDetection callout function
TlsConnection	0..32	**Label:** TlsConnection Container for incorporation of TlsConnection.

5.3.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	ArMajorVersion
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate im-plementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	4
Configuration class	**PublishedInformation:**

Origin	Elektrobit Automotive GmbH

Parameter Name	**ArMinorVersion**
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArPatchVersion**
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	3
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.

Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	8
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	789
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	Release
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.3.1.2. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	PbcfgMSupport
Label	PbcfgM support
Description	Specifies whether or not the Tls can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.3.1.3. TlsDefensiveProgramming

Parameters included	
Parameter name	**Multiplicity**
TlsDefProgEnabled	1..1
TlsPrecondAssertEnabled	1..1
TlsPostcondAssertEnabled	1..1
TlsStaticAssertEnabled	1..1

Parameters included	
TlsUnreachAssertEnabled	1..1
TlsInvariantAssertEnabled	1..1

Parameter Name	**TlsDefProgEnabled**
Label	Enable Defensive Programming
Description	Enables or disables the defensive programming feature for the module Tls. Note: This feature is dependent on the use of the development error detection module. To use the defensive programming feature, proceed as follows: 1. Enable development error detection 2. Enable defensive programming 3. Enable assertions as required
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**TlsPrecondAssertEnabled**
Label	Enable Precondition Assertions
Description	Enables handling of precondition assertion checks reported from the module Tls. Dependency on parameter(s): ▶ Enable Development Error Detection (`TlsDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`TlsDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**TlsPostcondAssertEnabled**
Label	Enable Postcondition Assertions

Description	Enables handling of postcondition assertion checks reported from the module Tls. Dependency on parameter(s): ▶ Enable Development Error Detection (`TlsDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`TlsDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**TlsStaticAssertEnabled**
Label	Enable Static Assertions
Description	Enables handling of static assertion checks reported from the module Tls. Dependency on parameter(s): ▶ Enable Development Error Detection (`TlsDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`TlsDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**TlsUnreachAssertEnabled**
Label	Enable Unreachable Code Assertions
Description	Enables handling of unreachable code assertion checks reported from the module Tls. Dependency on parameter(s): ▶ Enable Development Error Detection (`TlsDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`TlsDefProgEnabled`): must be enabled

Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsInvariantAssertEnabled**	
Label	Enable Invariant Assertions	
Description	Enables handling of invariant assertion checks reported from functions of the module Tls. Dependency on parameter(s): ▶ Enable Development Error Detection (`TlsDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`TlsDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.3.1.4. TlsGeneral

Parameters included	
Parameter name	**Multiplicity**
TlsDevErrorDetect	1..1
TlsCsmProcessing	1..1
TlsEnableJobBasedKeyExchange	1..1
TlsCsmKeyDeriveSupport	1..1
TlsMainFunctionPeriod	1..1
TlsEnableCertificates	1..1

Parameter Name	**TlsDevErrorDetect**
Label	TlsDevErrorDetect

Description	Switches the development error detection and notification on or off. TRUE = detection and notification is enabled. FALSE = detection and notification is disabled.
Multiplicity	1..1
Type	BOOLEAN
Default value	False
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**TlsCsmProcessing**
Label	TlsCsmProcessing
Description	Configures if the asynchronous or synchronous Csm interface is used. All jobs must be either synchronous or asynchronous.
Multiplicity	1..1
Type	ENUMERATION
Default value	TLS_CSM_ASYNCHRONOUS
Range	TLS_CSM_ASYNCHRONOUS TLS_CSM_SYNCHRONOUS
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**TlsEnableJobBasedKeyExchange**
Label	TlsEnableJobBasedKeyExchange
Description	TODO NOTE: Currently function is declared by Tls
Multiplicity	1..1
Type	BOOLEAN
Default value	False
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**TlsCsmKeyDeriveSupport**
Description	Specifies if the key derivation using Csm is supported: - TRUE: The key derivation is done externally through the Csm module. - FALSE: The key derivation is done in Tls.
Multiplicity	1..1

Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsMainFunctionPeriod**	
Label	TlsMainFunctionPeriod	
Description	Specifies the period of main function Tls_MainFunction in seconds.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.01	
Range	>0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsEnableCertificates**	
Label	TlsEnableCertificates	
Description	Specifies whether cipher suites using certificates are supported (True) or not (False).	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	False	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.3.1.5. TlsExtensionsDetectionConfig

Parameters included	
Parameter name	**Multiplicity**
TlsExtensionsDetectionHeaderFileName	1..1
TlsExtensionsDetectionCalloutName	1..1

Parameter Name	**TlsExtensionsDetectionHeaderFileName**

Description	This parameter specifies the name of the header file containing the definition of the ExtensionsDetection callout function.	
Multiplicity	1..1	
Type	STRING	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsExtensionsDetectionCalloutName**	
Description	This parameter defines the name of the ExtensionDetection callout function. This function is used to report each extensions to the TLS protocol received. Syntax: void Up_TlsExtensionDetection (P2CONST(TcpIp_SockAddrType, AUTOMATIC, TCPIP_APPL_DATA) remoteSockAddPtr, uint16 extensionType,)	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.3.1.6. TlsConnection

Parameters included	
Parameter name	**Multiplicity**
TlsConnectionId	1..1
TlsTransmitBufferSize	1..1
TlsClientIdentity	1..1
TlsServerIdentityHint	1..1
TlsReceiveBufferSize	1..1
TlsMaxRecordSize	1..1
TcpIpTlsUseSecurityExtensionRecordSizeLimit	1..1
TlsCipherSuite	1..1
TlsEndpoint	1..1
TlsDtls	1..1
CsmPreMasterSecretKeyRef	1..1
CsmMasterSecretKeyRef	1..1

Parameters included	
CsmMACGenerateKeyRef	1..1
CsmMACVerifyKeyRef	1..1
CsmEncryptKeyRef	1..1
CsmDecryptKeyRef	1..1
CsmPRFPreMasterSecretJobRef	1..1
CsmPRFMasterSecretJobRef	1..1
CsmMACGenerateJobRef	1..1
CsmMACVerifyJobRef	1..1
CsmHashJobRef	1..1
CsmRandomJobRef	1..1
CsmEncryptJobRef	1..1
CsmDecryptJobRef	1..1
TlsKeyMLocalCertRef	1..1
TlsKeyMLocalCertChainLength	1..1
TlsCsmLocalCertSigGenerateJobRef	1..1
TlsLocalCaDistinguishedName	0..1
TlsKeyMRemoteCertRef	0..1
TlsKeyMRemoteCertChainLength	0..1
TlsCsmKeyExchangeEphemeralGenerateKeyRef	1..1
TlsCsmKeyExchangeEphemeralExchangeKeyRef	1..1

Parameter Name	TlsConnectionId	
Label	TlsConnectionId	
Description	Identifier of the Tls connection.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	TlsTransmitBufferSize
Description	The size in bytes of the transmit buffer. It contains all data to be transmitted including TLS header overhead.
Multiplicity	1..1

Type	INTEGER	
Default value	1024	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsClientIdentity**	
Description	The TLS client identity that is sent during handshake.	
Multiplicity	1..1	
Type	STRING	
Default value		
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsServerIdentityHint**	
Description	The TLS server identity hint that is sent during handshake.	
Multiplicity	1..1	
Type	STRING	
Default value	Server_identity	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsReceiveBufferSize**	
Description	The size in bytes of the receive buffer. It contains all (encrypted) data that is received and the TLS record overhead(Header, MAC, IV).	
Multiplicity	1..1	
Type	INTEGER	
Default value	1024	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsMaxRecordSize**
Description	The maximum size of TLS records to be sent or received. Outgoing application data will be fragmented to fit this size. Incoming records that are bigger will be dropped. This parameter controls the size of the decryption buffer, and will be sent as the recordSizeLimit threshold if the corresponding extension is enabled.

	Also, this size should not be bigger than the TcpIp send buffers as a record must fit into one buffer. The size must be in the range 64 to 16384 bytes.	
Multiplicity	1..1	
Type	INTEGER	
Default value	1024	
Range	>=64	
	<=16384	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TcpIpTlsUseSecurityExtensionRecordSizeLimit**	
Description	TRUE Enables the RecordSizeLimit extension. The size is determined by the value configured in TlsMaxRecordSize.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	False	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsCipherSuite**	
Description	The cipher suite to negotiate for this connection.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	TLS_PSK_WITH_NULL_SHA256	
Range	TLS_PSK_WITH_NULL_SHA256	
	TLS_PSK_WITH_AES_128_GCM_SHA256	
	TLS_ECDHE_ECDSA_WITH_NULL_SHA	
	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsEndpoint**
Description	Defines whether Tls acts as a client (TLS_CLIENT) or as a server (TLS_SERV-ER) in this connection.

Multiplicity	1..1	
Type	ENUMERATION	
Default value	TLS_CLIENT	
Range	TLS_CLIENT	
	TLS_SERVER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsDtls**	
Description	TRUE Enables DTLS over UDP sockets.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	False	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CsmPreMasterSecretKeyRef**	
Label	CsmPreMasterSecretKeyRef	
Description	Reference to the TLS pre-master-secret Csm key. Can not be shared with other TLS connections.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CsmMasterSecretKeyRef**	
Label	CsmMasterSecretKeyRef	
Description	Reference to the TLS master-secret Csm key. Can not be shared with other TLS connections. Size must be 48 bytes.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CsmMACGenerateKeyRef	
Label	CsmMACGenerateKeyRef	
Description	Reference to the Csm key used for TLS record MAC generation. Can not be shared with other TLS connections. Size must match the size defined by the cipher suite.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CsmMACVerifyKeyRef	
Label	CsmMACVerifyKeyRef	
Description	Reference to the Csm key used for TLS record MAC verification. Can not be shared with other TLS connections. Size must match the size defined by the cipher suite.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CsmEncryptKeyRef	
Label	CsmEncryptKeyRef	
Description	Reference to the Csm key used for TLS AEAD encryption. Can not be shared with other TLS connections. Size must match the size defined by the cipher suite.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CsmDecryptKeyRef
Label	CsmDecryptKeyRef
Description	Reference to the Csm key used for TLS AEAD decryption. Can not be shared with other TLS connections. Size must match the size defined by the cipher suite.

Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CsmPRFPreMasterSecretJobRef**
Label	CsmPRFPreMasterSecretJobRef
Description	Reference to the Csm job used for TLS PRF with pre-master-secret key as input. Can not be shared with other TLS connections. Job primitive must be MacGenerate with HMAC-SHA256.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CsmPRFMasterSecretJobRef**
Label	CsmPRFMasterSecretJobRef
Description	Reference to the Csm job used for TLS PRF with master-secret key as input. Can not be shared with other TLS connections. Job primitive must be MacGenerate with HMAC-SHA256.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CsmMACGenerateJobRef**
Label	CsmMACGenerateJobRef
Description	Reference to the Csm job used for TLS MAC generation with CsmMACGenerateKeyRef as input. Can not be shared with other TLS connections. Job primitive must be MacGenerate with an algorithm that is defined by the cipher suite.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	CsmMACVerifyJobRef	
Label	CsmMACVerifyJobRef	
Description	Reference to the Csm job used for TLS MAC verification with CsmMACVerifyKeyRef as input. Can not be shared with other TLS connections. Job primitive must be MacVerify with an algorithm that is defined by the cipher suite.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CsmHashJobRef	
Label	CsmHashJobRef	
Description	Reference to the Csm job used for hashing during handshake. Hash algorithm must be SHA2-256.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CsmRandomJobRef	
Label	CsmRandomJobRef	
Description	Reference to the Csm random job used for generation of client_random, server_random and DTLS cookie. It must be seeded by the user.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CsmEncryptJobRef
Label	CsmEncryptJobRef
Description	Reference to the Csm job used for TLS AEAD encryption with CsmEncryptKeyRef as input. Job primitive must be configured according to the cipher suite. E.g. CsmAEADEncryptAlgorithmFamily = CRYPTO_ALGOFAM_AES, CsmAEADEncryptAlgorithmKeyLength = 16, CsmAEADEncryptAlgorithmMode = CRYPTO_ALGOMODE_GCM for TLS_PSK_WITH_AES_128_GCM_SHA256.

Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CsmDecryptJobRef**	
Label	CsmDecryptJobRef	
Description	Reference to the Csm job used for TLS AEAD decryption with CsmDecryptKeyRef as input. Job primitive must be configured according to the cipher suite. E.g. CsmAEADEncryptAlgorithmFamiliy = CRYPTO_ALGOFAM_AES, CsmAEADEncryptAlgorithmKeyLength = 16, CsmAEADEncryptAlgorithmMode = CRYPTO_ALGOMODE_GCM for TLS_PSK_WITH_AES_128_GCM_SHA256.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsKeyMLocalCertRef**	
Description	Reference to the local end-entity certificate.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsKeyMLocalCertChainLength**	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsCsmLocalCertSigGenerateJobRef**
Description	Reference to the job that generates signatures using the private key of the local end-entity certificate.
Multiplicity	1..1

Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsLocalCaDistinguishedName**	
Description	The distinguished name of the local certificate authority as an id-at-commonName uTF8String. It will be used to generate the CertificateRequest sent by the server.	
Multiplicity	0..1	
Type	STRING	
Default value		
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsKeyMRemoteCertRef**	
Description	Reference to the remote end-entity certificate.	
Multiplicity	0..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsKeyMRemoteCertChainLength**	
Multiplicity	0..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsCsmKeyExchangeEphemeralGenerateKeyRef**	
Description	Reference to the key that holds the local ephemeral private key used for the key exchange. This key is used for the generation of the local private key using Csm_KeyGenerate().	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild

Origin	Elektrobit Automotive GmbH	

Parameter Name	**TlsCsmKeyExchangeEphemeralExchangeKeyRef**	
Description	Reference to the key that holds the local ephemeral private key used for the key exchange. This key is used for the calculation of the local public key using Csm_KeyExchangeCalcPubVal() and the shared secret using Csm_KeyExchangeCalcSecret(). It can reference the same key as TlsCsmKeyExchangeEphemeralGenerateKeyRef.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.3.2. Application programming interface (API)

5.3.3. Integration notes

5.3.3.1. Exclusive areas

Exclusive areas information is not available for this module.

5.3.3.2. Production errors

Production errors information is not available for this module.

5.3.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
CODE
CONST_UNSPECIFIED
VAR_BOOLEAN
VAR_CLEARED_8
CONST_8
CONST_32
VAR_INIT_BOOLEAN
VAR_INIT_UNSPECIFIED
VAR_CLEARED_UNSPECIFIED

5.3.3.4. Integration requirements

WARNING **Integration requirements list is not exhaustive**

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the Tls module.