Elektrobit

EB tresos® E2E Wrapper 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

Confidential information.

Table of Contents

1. Overview of EB tresos E2E Wrapper documentation

Welcome to the EB tresos E2E Wrapper (E2EPW) product documentation.

This document provides:

- Chapter 2, “E2EPW release notes“: release notes for the E2EPW modules
- Chapter 3, “E2EPW module user's guide“: containing background information and instructions
- Chapter 4, “E2EPW module references“: information about configuration parameters and the application programming interface

2. E2EPW release notes

2.1. Overview

This chapter provides the E2EPW 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.

2.2. Scope of the release

2.2.1. 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

2.2.2. AUTOSAR modules

The following table lists the AUTOSAR modules that are part of this E2EPW release.

Module name	AUTOSAR version and revision	SWS version and revision	Module version	Supplier
E2EPW	4.0.3 []	2.0.0 [0000]	2.3.20	Elektrobit Automotive GmbH
E2EPWExt	4.0.3 []	2.0.0 [0000]	2.3.17	Elektrobit Automotive GmbH

Table 2.1. Hardware-Independent Modules specified by the AUTOSAR standard

2.2.3. 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 2.2. Modules not specified by the AUTOSAR standard

2.2.4. 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`.

2.3. Module release notes

2.3.1. E2EPW module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 2.0.0
- Module version: 2.3.20.B439717
- Supplier: Elektrobit Automotive GmbH

2.3.1.1. Change log

This chapter lists the changes between different versions.

Module version 2.3.20
2020-06-19

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

Module version 2.3.19
2019-10-11

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

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

Module version 2.3.18
2019-06-14

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

Module version 2.3.17
2019-02-15

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

Module version 2.3.16
2018-07-27

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

Module version 2.3.15
2018-06-22

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

Module version 2.3.14
2018-02-01

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

Module version 2.3.13
2017-09-22

- Switch from MISRA-C:2004 to MISRA-C:2012

Module version 2.3.12
2016-05-25

- Added support for disabling automatic code verification

Module version 2.3.11
2015-10-09

- Implemented separate plug-ins for `E2EPW` BSW code and `E2EPW` module code generator including the E2EPW Check Tool
- Implemented support for efficient data transmission

Module version 2.3.10
2015-06-19

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

Module version 2.3.9
2014-11-14

- Improved API documentation
- Improved error reporting of E2EPW MCG
- ASCE2EPW-891 Fixed known issue: E2EPW Check Tool: Missing review items if no Atomic SWCs are specified in AUTOSAR SWC-D

Module version 2.3.8
2014-04-30

- Improved usability of E2EPW Check Tool

Module version 2.3.7
2014-04-18

- Added support for automatic check of data element layout in E2EPW Check Tool for safety-approved communication description files
- Improved usability of E2EPW Check Tool
- ASCE2EPW-820 Fixed known issue: Profile VCC: Wrong CRC calculation for signed data types

Module version 2.3.6
2014-03-11

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

- ASCE2EPW-789 Fixed known issue: E2EPW MCG may generate incorrect code in case of protection of a mixed inter-/intra-ECU communication

Module version 2.3.5
2014-01-24

- Internal module improvement. This module version update does not affect module functionality
- Added support that unconnected ports are not generated in case of pure intra-ECU communication
- Added support for optional execution of E2EPW Check Tool within EB tresos Studio

Module version 2.3.4
2013-12-10

- Internal module improvement. This module version update does not affect module functionality
- ASCE2EPW-695 Fixed known issue: E2E protection may incorrectly report a CRC error in case of pure intra-ECU communication protected with E2E Profile 01
- Disabled data invalid check for profile VCC

Module version 2.3.3
2013-08-16

- Added support of VCC proprietary E2E profile (different serialization technique)
- Added support to read the `DataIDNibbleOffset` parameter from the EndToEnd description (AUTOSAR 4.0 Rev 3) via special data groups

Module version 2.3.2
2013-04-20

- Added support of generating `DataIDNibbleOffset` which is special to some Profiles

Module version 2.3.1
2013-04-19

- ASCE2EPW-606 Fixed known issue: `E2EPW` does not generate code in case of intra-ECU communication
- Added support that `E2EPW` generates non-compilable code if several receiver ports in the same Atomic SWC are protected with the same E2E description

Module version 2.3.0
2013-02-15

- Added range and de-serialization checks without constraints of padding bytes
- Provided a Basic Software Module Description that specifies the memory mappings
- ASCE2EPW-555 Fixed known issue: Generation of incorrect data mappings from data element members to group signals
- Removed usage of `Rte_IsUpdated` for redundant channels

Module version 2.2.2
2012-12-12

- Internal module improvement. This module version update does not affect module functionality
- Added support of E2E Profile 01 with different CRC and counter positions

Module version 2.2.1
2012-11-20

- ASCE2EPW-468 Fixed known issue: The E2EPW MCG does not generate all protected ports if the `DataElement` to `SignalGroup` mappings are added via the Rte assistant

Module version 2.2.0
2012-10-24

- Added support for import of AUTOSAR 3.2 Rev 2 EndToEnd protection information
- Updated E2EPW Check Tool to support AUTOSAR 3.2 Rev 2 system/SWCD files
- Implemented channel-specific initialization APIs for redundant channel wrappers

Module version 2.1.0
2012-07-27

- Added support for Profile re-sync after detection of an unexpected behavior of counter (AUTOSAR 3.-2 Rev 2)

Module version 2.0.0
2012-06-22

- Updated to AUTOSAR 4.0 Rev 3 regarding change of return values
- Updated to AUTOSAR 4.0 Rev 3 regarding initialization APIs
- Internal module improvement. This module version update does not affect module functionality
- Added support for import of AUTOSAR 3.2 Rev 1 and AUTOSAR 4.0 Rev 3 system and software component description

Module version 1.0.14
2012-01-14

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

Module version 1.0.13
2011-11-30

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

Module version 1.0.12
2011-11-25

- Internal module improvement. This module version update does not affect module functionality
- Implemented receiver-specific configuration of `maxDeltaCounterInit` parameter
- ASCE2EPW-342 Fixed known issue: In case of redundant channel wrappers, the linker signals an error of multiple defined symbols
- Implemented verification rules for the E2EPW Check Tool

Module version 1.0.11
2011-09-16

- ASCE2EPW-294 Fixed known issue: Support signed signals with signal size of `1` bit
- Added support for unaligned group signals in safety-related I-PDUs
- Internal module improvement. This module version update does not affect module functionality
- ASCE2EPW-309 Fixed known issue: A signal of type sint32 need to have a bit length >16 bit
- ASCE2EPW-285 Fixed known issue: Generated `E2EPW` target code uses the `DataElementPrototype` of composition if E2E description references to composition port

- ASCE2EPW-278 Fixed known issue: The `E2EPW` generator will fail if a protected signal group does not (partially) occupy PDU byte 0 for profiles `PROFILE_DAI_BR222_ID` and `PROFILE_DAI_BR222_noID`
- MCG: Added support of `ADMIN-Data Daimler:UsesE2EProtection` (consider only E2E-protected ports for code generation)
- MCG: Added support to issue a warning and ignore port if E2E-protected delegation ports are incompatible or degraded
- MCG: Added a warning and ignore port if E2E-protected data elements and mapped signal groups are not in an 1:1 relation (i.e. signal group degradation)
- MCG: Added a check for `IPdu:unusedBitPattern` specified in the communication description
- MCG: Added support for the definition of multiple `END-TO-END-PROTECTION-DATA-ELEMENT-PROTOTYPE` instances
- Added support for the configuration of unaligned group signals in safety-related I-PDUs
- Implemented E2EPW Check Tool (initial version)

Module version 1.0.10
2011-05-20

- Internal module improvement. This module version update does not affect module functionality
- ASCE2EPW-220 Fixed known issue: Signature of `Rte_IsUpdated`
- Internal module improvement. This module version update does not affect module functionality
- ASCE2EPW-234 Fixed known issue: Support `DataPrototype` references to delegation ports for intra-ECU communication

Module version 1.0.9
2011-04-20

- Removed unused parameter `MaxDataLength` from configuration data
- Corrected typing from `INITAL` to `INITIAL` in the E2E Profile receiver states

Module version 1.0.8
2011-04-06

- ASCE2EPW-157 Fixed known issue: Missing file generation for Rx-SWC in case of inter-ECU communication with Rte-fanout
- Added support to transmit protected data elements only if the library routine `E2E_PXXProtect()` returns without error

- ASCE2EPW-165 Fixed known issue: The E2EPW MCG creates duplicate functions for intra-ECU receiver if the receiver contains a `DataElement` to `SignalGroup` mapping
- Added support of CRC and counter signals placed in nested record structures
- ASCE2EPW-167 Fixed known issue: The `E2EPW` generator fails to generate code if an E2E protection element for pure intra-ECU communication contains an opaque byte array member
- ASCE2EPW-168 Fixed known issue: The E2E libraries do not correctly calculate the CRC if the communication is intra-ECU and the data element is complex
- Added files shall not be generated by the E2EPW MCG
- Implemented initialization functions for wrapper routines at sender side

Module version 1.0.7
2011-03-18

- Added support to put all function pointers into the generic configuration structure
- Removed code regarding existence of `RteInstance`
- Added support to configure multiple CRC and counter member prefixes
- ASCE2EPW-150 Fixed known issue: Possible multiple definition of initialization function for a protected receiving data element

Module version 1.0.6
2011-02-28

- Added support to generate initialization functions for all Rx E2E protected data elements
- Implemented E2EPW Check Tool

Module version 1.0.5
2011-02-14

- Added profile support for E2EPDAI1 and E2EPDAI2
- Implemented re-ordering of header file inclusions
- Added support that data element members themselves can be of complex type
- Added support that E2E modules shall be put into a new E2E-protection cluster for group selection in EB tresos Studio
- ASCE2EPW-88 Fixed known issue: Fixing stack overflow in case of inter-ECU communication of complex data elements

Module version 1.0.4
2011-01-28

- Added EndToEnd bypass mode (disabling the protection mechanism)
- Added `E2EPW` return value `E2EPW_STATUS_DATAINVALID` if E2E Profile 01A is used
- Added option for checking a maximum allowed data length to be E2E-protected
- Added support for intra-ECU communication protection with complex data elements
- Added support of dual import of Fibex 3 file and SW-C description that both reference different systems/ECUs
- ASCE2EPW-53 Fixed known issue: Reduction of macro expansion levels
- Added usage of EB tresos Studio 11.0.0 (beta)

Module version 1.0.3
2011-01-11

- ASCE2EPW-54 Fixed known issue: The `E2EPW` generator always generates both read and write functions
- Added return type `E2EPW_STATUS_NONEWDATA` for the `E2EPW` Read and Read1 wrappers in byte `0`

Module version 1.0.2
2010-12-23

- Implemented redundant wrapper support
- Implemented support for Profile 01A
- Implemented version check in the static code of the `E2EPW` module

Module version 1.0.1
2010-11-30

- Added support for optional `Rte_Instance` argument of wrapper routines
- Added support that the `E2EPW_Read()` function shall provide additional return values for byte `0` for the indication of detected failure modes
- ASCE2EPW-30 Fixed known issue: The E2EPW MCG does not find `DataElement-To-Signal` mappings if the port prototype reference is placed at the composition level

Module version 1.0.0
2010-11-12

- Initial release

2.3.1.2. New features

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

2.3.1.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- [ASCE2ESE-213] Range checks at sender side

  Description:

  The `E2EPW`E2EPW module performs range checks during operation of the transmitted Data Elements with respect to the underlying signal group layout.

  Rationale:

  Safety projects require defensive implementation techniques which ensure consistency of implemented interfaces.

- [ASCE2ESE-236] E2E Profile 1, Variant C

  Description:

  In addition to AUTOSAR 4.0 Rev 3, the E2EPW module supports the generation of E2EPW wrapper code for ports that are protected with E2E Profile 1 variant C, which makes use of 12-bit DataId where the higher 4 bits are transmitted explicitly at position `DataIdNibbleOffset` according to AUTOSAR version 4.1 Rev 1. The `DataIdNibbleOffset` can be specified in an AUTOSAR 4.0 Rev 3 EndToEnd description by using the special data group identifier `DataIdNibbleOffset`. Otherwise default value `12` is taken.

- [ASCE2ESE-256] Support of VCC proprietary E2E profile

  Description:

  The E2EPW module supports the generation of E2EPW wrapper code with a different serialization technique according to the proprietary E2E profile *ProfileVCC* for the calculation of the CRC value. That is, the data is serialized in alphabetic order with respect to the short names of the data element members which is different to the serialized data transmitted over the communication bus.

- [ASCE2EPW-61] Efficient intra-ECU (inter-partition) communication protection with complex data elements

  Description:

Since AUTOSAR does not specify a serialization technique for intra-ECU (inter-partition) communication, the following method is used:

If the E2E Profile is different than E2E Profile 01 and E2E Profile VCC, then the E2E Protection Wrapper serializes as follows:

1. Copy the memory allocated by the complex data element byte-wise to a byte array with padding bytes set to `0`.
2. Swap the memory of the CRC struct member with the memory at the profile-specific position of the CRC (1 byte).
3. Swap the memory of the counter struct member with the memory at the profile-specific position of the counter (1 byte).
4. Apply the specified protection mechanism on the memory with the swapped counter and CRC byte with data length set to the number of bytes of the memory allocated by the complex data element.

If the E2E Profile equals E2E Profile 01, then the E2E Protection Wrapper serializes as follows:

1. Copy the memory allocated by the complex data element byte-wise to a byte array with padding bytes set to `0xFFU`.
2. Apply the specified protection mechanism (E2E Profile 01) on the allocated memory with the following E2E Profile configuration data (setting of data length and offsets):
    a. The data length shall be set to the number of bytes of the memory allocated by the complex data element.
    b. The counter offset shall be set to the bit-offset of the related counter data element member within the complex data element type. This is platform-dependent.
    c. The CRC offset shall be set to the bit-offset of the related CRC data element member within the complex data element type. This is platform-dependent.
    d. The `DataIDNibble` offset, if used, shall be set to the bit-offset of the related `DataIDNibble` data element member within the complex data element. This is platform-dependent.
    e. The `DataIDNibble` offset, if not used, shall be set to `0`.

If the E2E Profile equals E2E Profile VCC, then the E2E Protection Wrapper serializes as follows:

1. Copy the memory allocated by the complex data element byte-wise to a byte array with padding bytes set to `0`.
2. Apply the specified protection mechanism (E2E Profile 01) on the allocated memory with the following E2E Profile configuration data (setting of data length and offsets):
    a. The data length shall be set to the number of bytes of the memory allocated by the complex data element.
    b. The counter offset shall be set to the bit-offset of the related counter data element member within the complex data element type. This is platform-dependent.

c. The CRC offset shall be set to the bit-offset of the related CRC data element member within the complex data element type. This is platform-dependent.

d. The `DataIDNibble` offset, if used, shall be set to the bit-offset of the related `DataIDNibble` Data Element member within the complex data element. This is platform-dependent.

e. The `DataIDNibble` offset, if not used, shall be set to `0`.

- [ASCE2EPW-506] Support detection of protected ports without duplication or extension of AUTOSAR EndToEnd description information

  Description:

  The E2EPW MCG supports the automatic detection of all protected sender-receiver ports that have `UsesEndToEndProtection` set to true. This behavior is based on the reference to the protected ISignalGroup/IPdu in the AUTOSAR EndToEnd description information without requiring additional references.

  Rationale:

  OEM/Tier-1 collaboration scenario where the AUTOSAR system description provided by the OEM specifies the protection of a composition port and the Tier-1 extends the OEM file without re-specifying or updating the AUTOSAR EndToEnd Communication Description.

- [ASCE2ESE-345] Support for efficient data transmission

  Description:

  The E2EPW provides an option to generate an efficient data transmission. If enabled, the `Rte API Rte_WriteArray_p_o()` is called with the serialized byte array instead of `Rte_Write_p_o()`. See http://www.autosar.org/bugzilla/show_bug.cgi?id=65084.

  Rationale:

  The serialization of the application data element is done twice otherwise, once by the E2E Protection Wrapper and a second time by the `Com` module.

2.3.1.4. Deviations

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

- Data element is only transmitted if E2E library returns without errors

  Description:

  In contrast to E2EUSE0280 and E2EUSE0264 which state that a data element is always transmitted, the `E2EPW` only transmits a data element via the `Rte`, if the E2E library returns without errors.

  Rationale:

If the E2E library returns an error, it is highly likely that the passed data element is invalid. If it is still transmitted, the receiver may recognize this invalid data element as valid. In this case, it is better to recognize a reception timeout.

Requirements:

E2EUSE0280, E2EUSE0264

- The return type of the initialization APIs is `Std_ReturnType`

  Description:

  In contrast to E2EUSE0296 and E2EUSE0300 which state that the initialization functions have return type uint8, the return type `Std_ReturnType` is used for error indication.

  Rationale:

  AUTOSAR provides the standard API return type `Std_ReturnType` for sharing error flags between the RTE and the BSW modules. It is used with values `E_OK` or `E_NOT_OK` and user-specific error flags with values between `0` and `63` of type uint8, e.g. `E2E_E_OK`. See http://www.autosar.org/bugzilla/show_bug.cgi?id=55708.

  Requirements:

  E2EUSE0296, E2EUSE0300

2.3.1.5. 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.

- Limitation: Rte signal fan-out restricted to data elements with identical signal-group layout

  Description:

  Limitation according to AUTOSAR E2E0255. Furthermore the timing of all related I-PDUs must be identical or message loss may occur at the related receivers and the E2E protection configuration parameter `maxDeltaCounterInit` has to be selected appropriately. Instead of a `Rte` signal fan-out for inter-ECU communication, a PDU fan-out at the `PduR` could be a better decision.

  Rationale:

  The E2E Protection Wrapper calls the E2ELib and writes the calculated CRC and sequence counter back to the complex data element, e.g. update of struct members CRC and sequence counter. If the PDU layouts of the *fan-out* signal groups are different, the E2E Protection Wrapper would have to create individual I-PDU representations for the different fan-out signal groups.

Based on these representations, separate CRC and sequence counter values would have to be calculated via multiple calls of the E2ELib. Furthermore, the data elements would require multiple CRC and sequence counters. While this could theoretically be achieved, it would be absolutely inefficient.

- CRC and sequence counter shall be of primitive unsigned data type

  Description:

  If the data element is complex, then the CRC and sequence counter shall be of the primitive data type uint8. This means that the corresponding finally referenced base type must be of type uint8.

  Rationale:

  Other primitive data types, e.g. sin8, uint8_n, etc. are either not compatible due to the bit extension for signed integer types or make no sense at all.

- Restriction of supported data elements

  Description:

  The safety-related data elements shall be of complex type. According to AUTOSAR 4.0, this means that the generated `ImplementationDataType` of a `VariableDataPrototype` shall be of category `STRUCTURE`. According to AUTOSAR 3.1 and 3.2, this means that the `VariableDataPrototype` shall be a composite data type of type `RecordType`.

  Rationale:

  Other types can be represented by a member of a composite data type.

- Limitation of start-up behavior

  Description:

  According to E2EUSE0280, E2EUSE0192, E2EUSE0262, E2EUSE0264, E2EUSE0266, and E2EUSE0268, the E2EPW should instantiate and initialize the required configuration and variables. This cannot be fulfilled.

  Rationale:

  That would require that `E2EPW` can detect that the current invocation of an `E2EPW` API is the first invocation of the API. Since that is not possible in case the start-up code is not correct, EB_E2EPW020014 of the Safety Manual shall be taken into account. See also http://www.autosar.org/bugzilla/show_bug.cgi?id=62921.

- Restriction on protection of multiple ports with same E2E description information

  Description:

In case an Atomic SWC contains multiple ports protected with the same E2E description information, then these ports must be either Tx or Rx ports only. Mixing types are not supported in this use case.

2.3.1.6. Open-source software

E2EPW does not use open-source software.

2.3.2. E2EPWExt module release notes

- Module version: 2.3.17.B439717
- Supplier: Elektrobit Automotive GmbH

2.3.2.1. Change log

This chapter lists the changes between different versions.

Module version 2.3.17
2020-06-19

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

Module version 2.3.16
2020-02-21

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

Module version 2.3.15
2019-10-11

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

Module version 2.3.14
2019-06-14

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

Module version 2.3.13
2018-12-21

- ASCE2EPW-1130 Fixed known issue: Wrongly computed byte size of signals using safety profile ProfileVCC

Module version 2.3.12
2018-10-26

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

Module version 2.3.11
2017-10-27

- Removed license checks from module `E2EPW`
- Extend prefix matches to regex matches for finding the CRC and the counter ApplicationRecordElement

Module version 2.3.10
2015-10-09

- Implemented separate plug-ins for `E2EPW` BSW code and `E2EPW` module code generator including the E2EPW Check Tool
- ASCE2EPW-933 Fixed known issue: E2EPW Check Tool may incorrectly report a failed verification if an AUTOSAR 3.2 Rev 2 System Description file is used as input for the E2EPW Check Tool
- ASCE2EPW-918 Fixed known issue: Support for import of OEM/Tier-1 files that contain different `EndToEndProtectionSets` referencing same communication
- ASCE2EPW-953 Fixed known issue: `E2EPW` may not generate the wrapper routines if the related port specifies multiple `DataElements`
- Implemented support for efficient data transmission
- ASCE2EPW-954 Fixed known issue: `E2EPW` does not support specification of receiver properties `SyncCounterInit` and `MaxNoNewOrRepeatedData` for AUTOSAR 4.x

2.3.2.2. New features

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

2.3.2.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

2.3.2.4. Deviations

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

- Restrictions of port interface mappings in AUTOSAR 4

  Description:

  In contrast to AUTOSAR 4.0 which allows a mapping between different port interfaces, the `E2EPW` code generator does not support the generation of protected ports which are connected between different Port Interfaces. That is, port interfaces of protected ports on the same protected 1:n communication must have the same short name.

  Rationale:

  Different port interfaces are only used for degraded data elements.

- Restriction on the specification for the protection of pure intra-ECU communication in AUTOSAR 3.1 Rev 4DAI4

  Description:

  If an E2E description in AUTOSAR 3.1 Rev 4 DAI4 specifies a pure intra-ECU communication, i.e. it contains only references to the protected software component ports which have no mapping to a signal group, then all referenced sender and receiver ports must be referenced in one `EndToEndProtectionDataElementPrototype`. That is, it is not possible to specify multiple instances of `EndToEndProtectionDataElementPrototype`.

  Rationale:

  Restriction of importer tooling of Eb tresos Studio.

- AUTOSAR 4.0 meta model restriction regarding specification of E2E-protected data element type

  Description:

  In system/software component description files according to AUTOSAR 4.0, all protected `VariableDataPrototype` shall be of type `ImplementationDataType`. That is, `ApplicationDataTypes` are not allowed.

  Workaround

  Use the specification of `ImplementationDataTypes` instead of `ApplicationDataTypes` for protected data elements.

Rationale:

The `ApplicationDataType` was introduced in AUTOSAR 4.0 in order to provide an abstract data type for the software components which are mapped to an `ImplementationDataType` via data conversion methods. Since AUTOSAR does not provide the specification of the inverse data conversion (which is even possible in some cases) and would be required for the E2E Protection Wrapper, the specification of `ApplicationDataType` makes no sense for protected data elements.

2.3.2.5. 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.

- Limitation: Rte signal fan-out restricted to data elements with identical signal-group layout

  Description:

  Limitation according to AUTOSAR E2E0255. Furthermore the timing of all related I-PDUs must be identical or message loss may occur at the related receivers and the E2E protection configuration parameter `maxDeltaCounterInit` has to be selected appropriately. Instead of a `Rte` signal fan-out for inter-ECU communication, a PDU fan-out at the `PduR` could be a better decision.

  Rationale:

  The E2E Protection Wrapper calls the E2ELib and writes the calculated CRC and sequence counter back to the complex data element, e.g. update of struct members CRC and sequence counter. If the PDU layouts of the *fan-out* signal groups are different, the E2E Protection Wrapper would have to create individual I-PDU representations for the different fan-out signal groups.

  Based on these representations, separate CRC and sequence counter values would have to be calculated via multiple calls of the E2ELib. Furthermore, the data elements would require multiple CRC and sequence counters. While this could theoretically be achieved, it would be absolutely inefficient.

- CRC and sequence counter shall be of primitive unsigned data type

  Description:

  If the data element is complex, then the CRC and sequence counter shall be of the primitive data type uint8. This means that the corresponding finally referenced base type must be of type uint8.

  Rationale:

  Other primitive data types, e.g. sin8, uint8_n, etc. are either not compatible due to the bit extension for signed integer types or make no sense at all.

- Restriction of supported data elements

Description:

The safety-related data elements shall be of complex type. According to AUTOSAR 4.0, this means that the generated `ImplementationDataType` of a `VariableDataPrototype` shall be of category `STRUCTURE`. According to AUTOSAR 3.1 and 3.2, this means that the `VariableDataPrototype` shall be a composite data type of type `RecordType`.

Rationale:

Other types can be represented by a member of a composite data type.

- Limitation of start-up behavior

  Description:

  According to E2EUSE0280, E2EUSE0192, E2EUSE0262, E2EUSE0264, E2EUSE0266, and E2EUSE0268, the E2EPW should instantiate and initialize the required configuration and variables. This cannot be fulfilled.

  Rationale:

  That would require that `E2EPW` can detect that the current invocation of an `E2EPW` API is the first invocation of the API. Since that is not possible in case the start-up code is not correct, EB_E2EPW020014 of the Safety Manual shall be taken into account. See also http://www.autosar.org/bugzilla/show_bug.cgi?id=62921.

- Restriction on protection of multiple ports with same E2E description information

  Description:

  In case an Atomic SWC contains multiple ports protected with the same E2E description information, then these ports must be either Tx or Rx ports only. Mixing types are not supported in this use case.

2.3.2.6. Open-source software

The software that is delivered with EB tresos AutoCore Generic can be classified into the following two categories:

- Software that is executed on the electronic control unit (ECU).
- Software that is used for the development infrastructure (configuration, generation, building) and thus executed on the development platform.

2.3.2.6.1. Open-source software in software executed on the ECU

No open-source software that runs on the ECU is delivered with E2EPWExt.

2.3.2.6.2. Open-source software in software used for the development infrastructure

The following list of open-source software that is used in development is delivered with E2EPWExt

- libxml2
  2.7.7
  xmlsoft.org

  List of licenses:

  - libxml2-license.txt

  List of copyrights:

  - Copyright (c) 2011 xmlsoft.org
- libxslt2
  1.1.26
  xmlsoft.org

  List of licenses:

  - libxslt2-license.txt

  List of copyrights:

  - Copyright (c) 2011 xmlsoft.org
- Python
  2.7.2
  Python Software Foundation

  List of licenses:

  - python-license.txt
  - python-3rd-party-licenses.txt

  List of copyrights:

  - Copyright (c) 2001-2011 Python Software Foundation
- lxml
  2.3.0
  Infrae

  List of licenses:

  - lxml-license.txt

  List of copyrights:

  - Copyright (c) 2004 Infrae

- ANTLR
  3.1.3
  Terence Parr

  List of licenses:

  - antlr-license.txt

  List of copyrights:

  - Copyright (c) 2010 Terence Parr

3. E2EPW module user's guide

3.1. Overview

This user's guide describes the E2EPW module. From this user's guide you learn the basic functionality of the E2EPW. You also learn which related modules are necessary to configure the E2EPW module. The E2EPW module reference provides further information on how to configure the E2EPW itself.

Note that this user's guide is intended for readers who have good knowledge of AUTOSAR and about the purpose of the E2EPW. The information provided here helps you to integrate the E2EPW in your AUTOSAR project.

- Section 3.2, “Background information” provides an overview of the basic functionality of the E2EPW.
- Section 3.3, “Configuring E2EPW” provides information on related modules that are needed in order to configure the E2EPW.
- Section 3.4, “E2EPW integration notes” provides notes for the integration of the E2EPW module into your project.
- For details on how to configure the E2EPW itself, see the parameter descriptions provided in the E2EPW module reference Chapter 4, “E2EPW module references“.

3.2. Background information

The exchange of data between two SW-C components over an unprotected communication link which consists of communication software at the sender and the receiver and a physical communication link may be affected by faults. For more information, see Figure 3.1, “Unprotected communication”. Such faults in the communication link can affect the integrity of exchanged data which may lead to a violation of the safety goal if such data is safety-related. Possible faults are:

- Random hardware faults, e.g. corrupt registers of a FlexRay controller
- Transient faults, e.g. interference due to EMC on the physical link
- Systematic faults within the communication software, e.g. an issue in the Com module

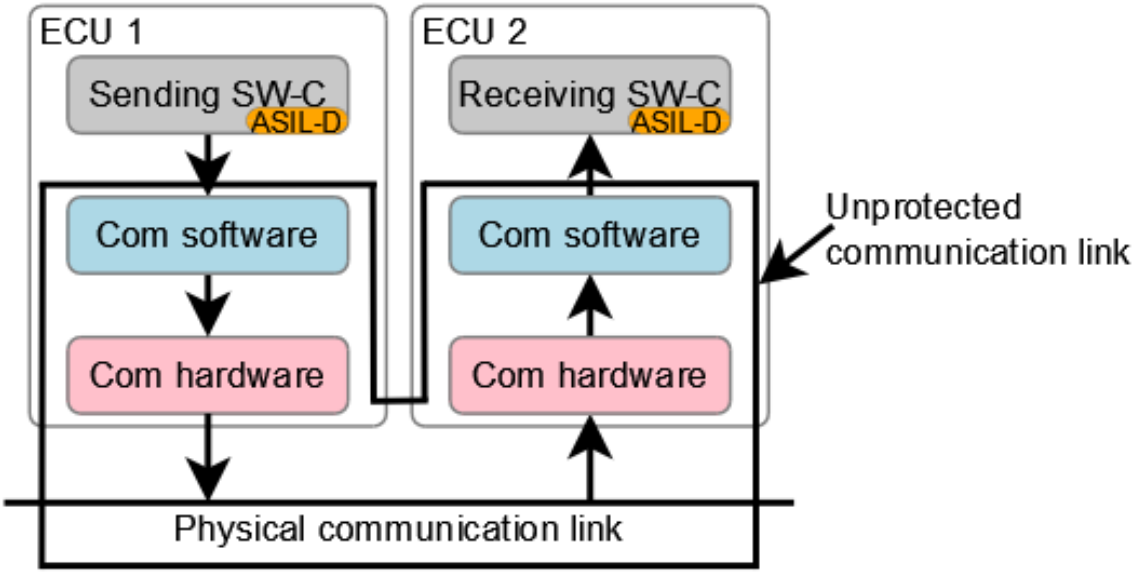

Figure 3.1. Unprotected communication

According to ISO 26262, part 6, clause D.2.4 [2], the causes for faults or effects of faults such as these listed below can be considered for each sender or each receiver with respect to the exchange of information. That is, such faults can cause interference between software elements):

ID	Failure mode name
1	repetition of information
2	loss of information
3	delay of information
4	insertion of information
5	masquerade or incorrect addressing of information
6	incorrect sequence of information
7	corruption of information
8	asymmetric information sent from a sender to multiple receivers
9	information from a sender received by only a subset of the receivers
10	blocking access to a communication channel

Table 3.1. Causes for communication faults or effects of faults.

The detection of these failure modes at the receiving SW-C requires the integration of safety mechanisms that includes the transmission of related protection information to the safety-related user data by the sending SW-C and evaluation of this protection information data by the receiving SW-C. For more information, see Figure 3.2, “Protected communication”. Therefore AUTOSAR has specified a standard AUTOSAR library [1], which implements safety mechanisms for protected communication according to specific E2E Profiles.

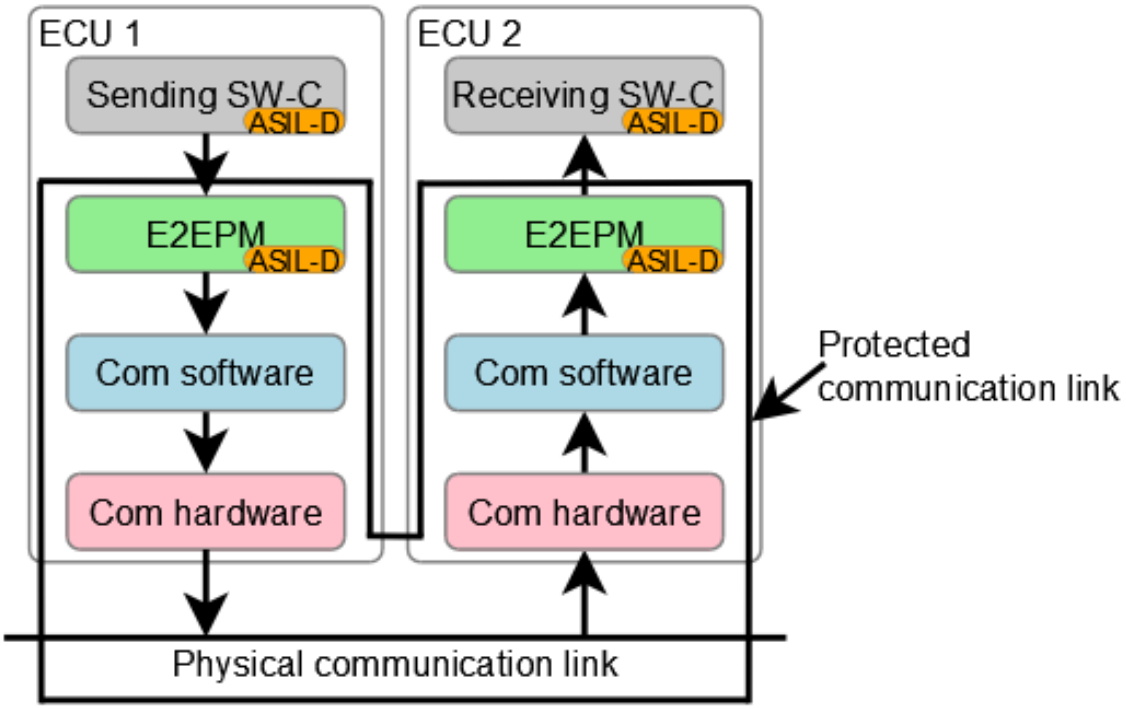

Figure 3.2. Protected communication

A safety mechanism in the `E2EPW` module is a functionality with the purpose to detect communication faults. This includes functionality at the sender that usually adds protection information to the payload data, as well as functionality at the receiver side that processes the received information in order to detect a communication fault.

This `E2EPW` module implements these safety mechanisms and is located between the sending SW-C and the communication software (RTE and AUTOSAR communication stack) respectively between the receiving SW-C and the communication software.

3.3. Configuring E2EPW

To configure the `E2EPW` module, add the module to your project using EB tresos Studio. Parameter descriptions are provided to guide the configuration. You find these in the module references section of this document. You also find these in the parameter description in EB tresos Studio.

To use the `E2EPW` module, you must configure additional modules as outlined below:

- Eb tresos E2E Protection: The `E2EPW` module requires the following library modules to add and check protection information:

- `E2EPxx`: AUTOSAR E2E Profile called from `E2EPW` where xx specifies the profile ID, e.g. `E2EP01` or `E2EP02`.
- `E2E`: generic data types for E2E Profiles
- `SCrc`: CRC routines called from the AUTOSAR E2E Profile

3.3.1. Creating a new EB tresos Studio project for E2E protection

To create a new EB tresos Studio project, follow the instructions of the EB tresos Studio user's guidein section `7.3.Working with projects .`

On the **Module Configurations** page, add the `E2EPW` module as well as the safety modules `E2E`, `E2EPxx`, and `SCrc`.

If your project shall contain the complete communication stack, add all remaining required modules in the same way.

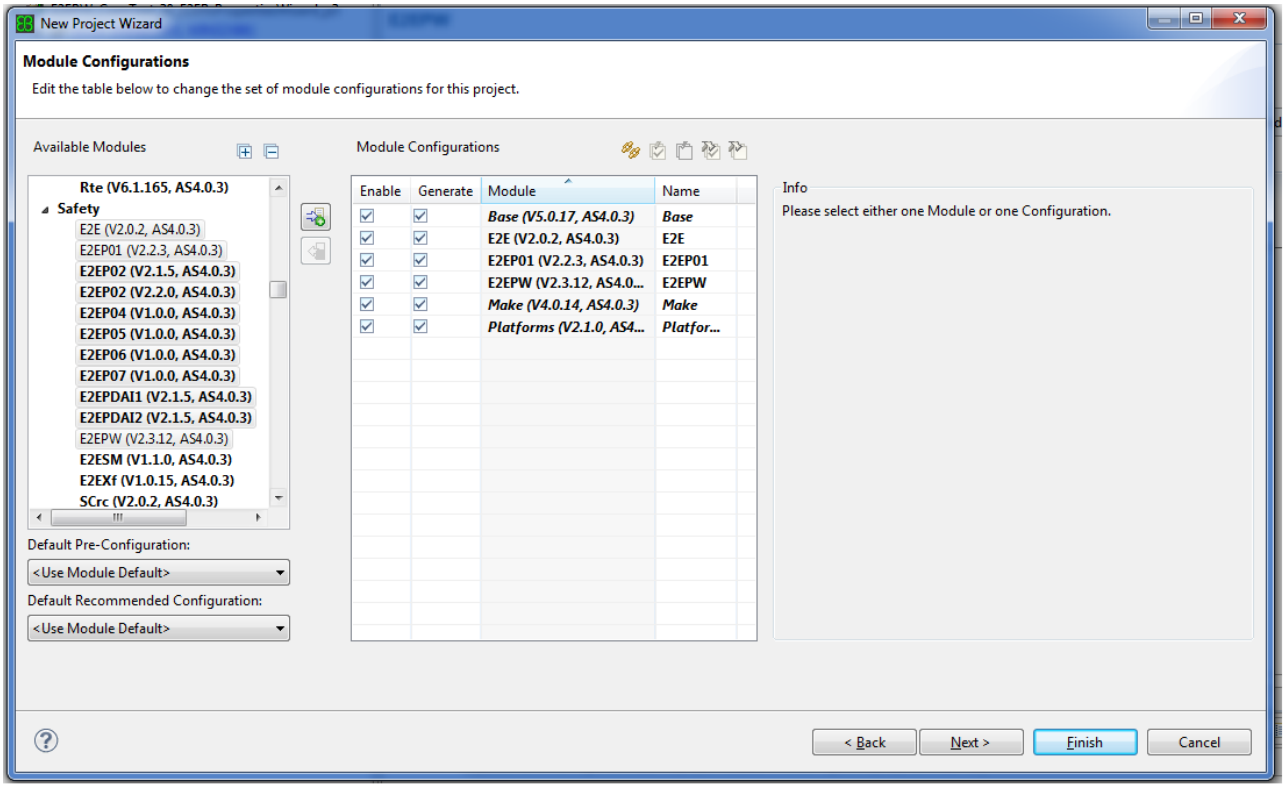

Figure 3.3. New Project Wizard

3.3.2. Importing configuration data

To import configuration data, follow the instructions of the EB tresos Studio user's guide in section `7.10.2.3.Importing AUTOSAR system descriptions .`

3.3.3. Editing parameters of E2EPW

To create a new EB tresos Studio project, follow the instructions of the EB tresos Studio user's guide in section `7.3.Working with projects.`

Set the following parameters for `E2EPW` :

- For **Wrapper Mode** either `SingleChannel` or `RedundantChannel`.
- Select the RteIsUpdated feature if you want to use it.
- If a pure intra-ECU communication is protected, a list of CRC, Counter, and DataIDNibble (Profile 01 variant C only) prefixes must be configured to identify the related CRC, counter, and DataIDNibble member of a protected complex data element.

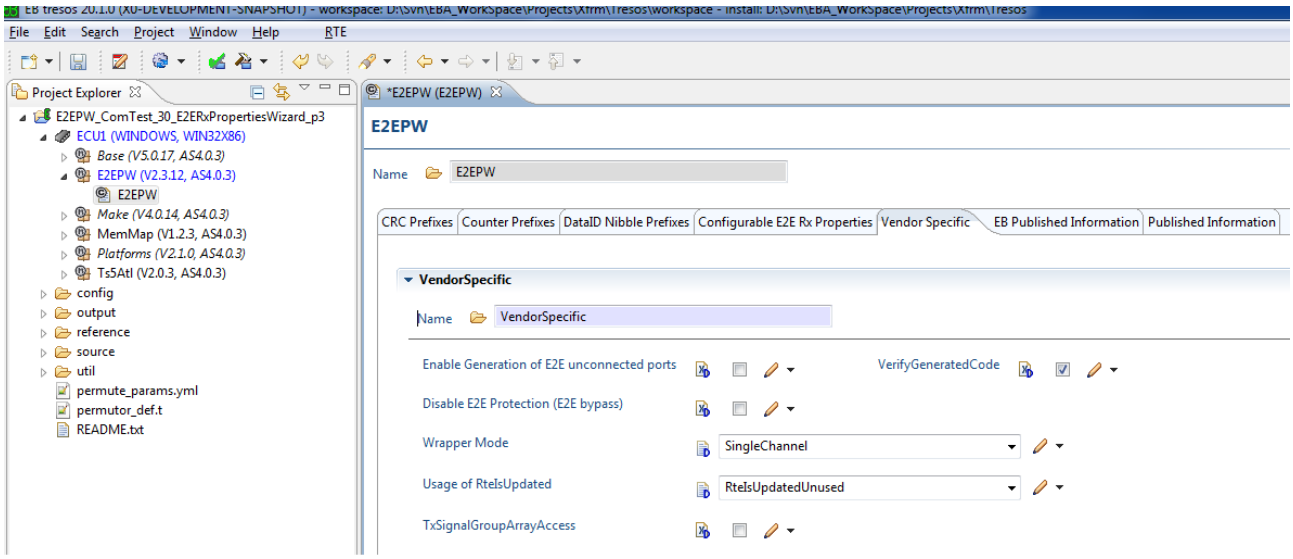

Figure 3.4. Configuration of the E2EPW

3.3.4. Extended editing

As an extended configuration, the E2EPW configuration plugin allow to reconfigure the imported `MaxDelta-CounterInit` parameter. In contrast to AUTOSAR, you can configure this parameter per receiver and not only per E2E protection.

The usage of a user-specific E2E Rx property `MaxDeltaCounterInit` instead of values specified in the communication description is required if only a global default value for E2E Rx property `MaxDeltaCounterInit` is used in the OEM communication description, whereas an ECU-specific value is defined in the requirements specification of the ECU.

The usage of a receiver-specific definition of E2E Rx property `MaxDeltaCounterInit` could be required if a protected message shall be received by more than one receiver and at least two receivers have different ASIL levels assigned.

You can configure the E2E Rx property `MaxDeltaCounterInit` in the *Configurable E2E Rx Properties* tab of the `E2EPW` editor. As the parameter is specified per triple of protected software component type, port, and data element, this information has to be retrieved from the system configuration first. Therefore the **Update list of configurable E2E Rx properties** wizard needs to be called. Figure 3.5, "**Update list of configurable E2E Rx properties** wizard" shows where you can start this wizard in EB tresos Studio. The wizard then creates for each E2E receiver a list entry and uses the `MaxDeltaCounterInit` value from the communication description as an initial value. An example result is shown in Figure 3.6, "Result after the **Update list of configurable E2E Rx properties** wizard is called". Therein, you can optionally edit the configurable parameter for each entry in this list, e.g. `MaxDeltaCounterInit`.

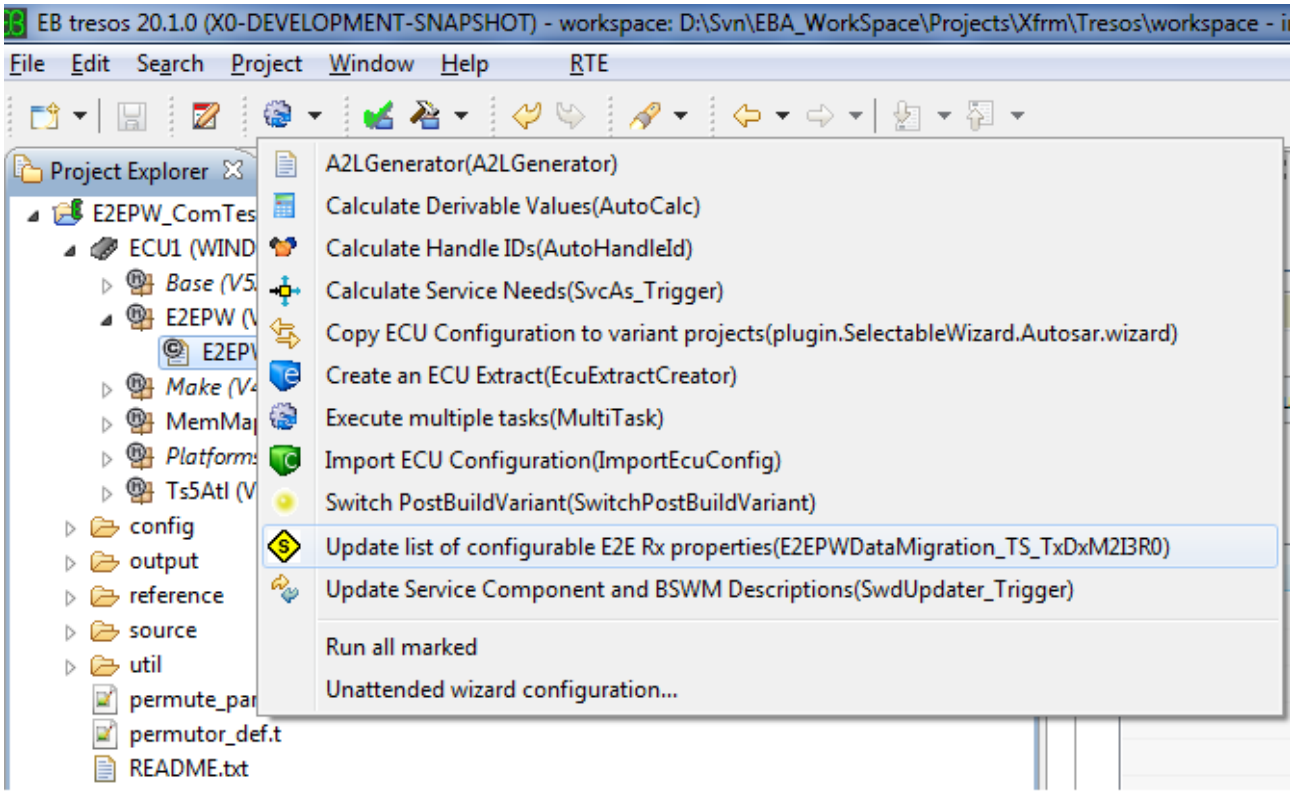

Figure 3.5. **Update list of configurable E2E Rx properties** wizard

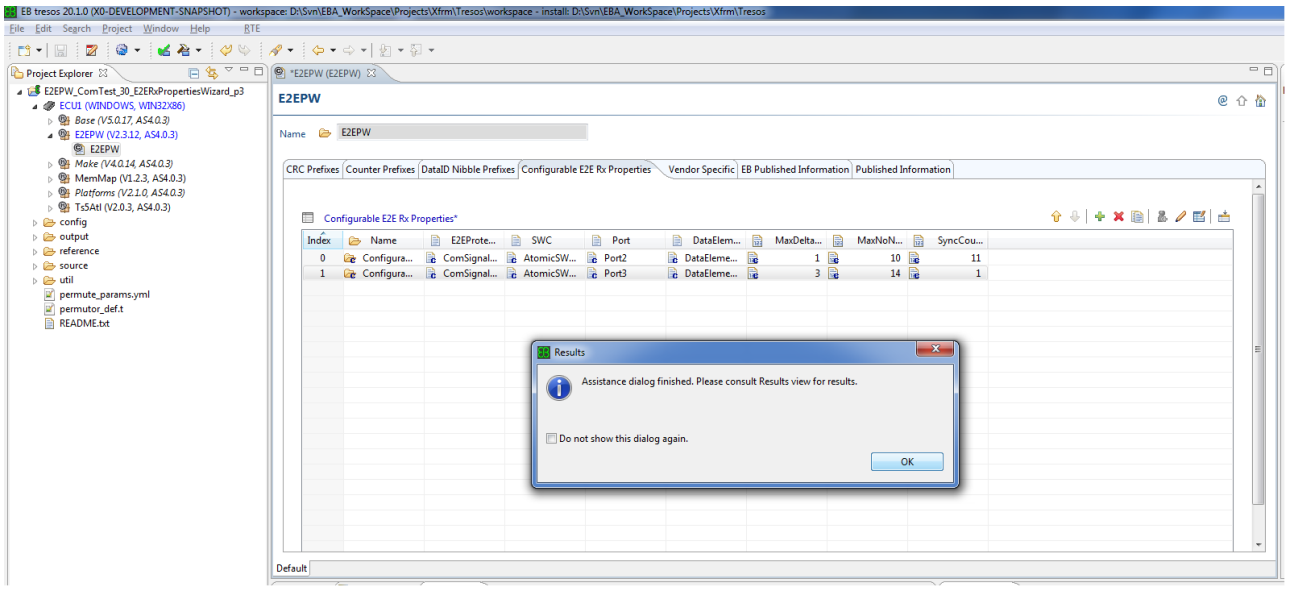

Figure 3.6. Result after the **Update list of configurable E2E Rx properties** wizard is called

Description of the list parameters:

- E2EProtection: Short name of the E2E protection information associated with `SWC-Port-DataElement`
- SWC: Short name of the protected software component type
- Port: Short name of the protected port associated to SWC
- DataElement: Short name of the protected data element associated to Port
- `MaxDeltaCounterInit`: user-definable value for the triple of SWC, port and data element.

If the **Configurable E2E Rx Properties** tab already contains a list, the wizard performs a merge of the data from the communication description and the existing list according to the following merge policy:

- List entries are deleted, which are no longer associated to any E2E protection information of the imported communication description.
- List entries are not updated if all of the following conditions are met:
  - A triple of SWC, port, and data element is available in both the communication description and the list of configurable E2E Rx properties.
  - A triple of SWC, port, and data element is associated with a valid E2E protection description.
  - `MaxDeltaCounterInit` has been changed.
- `MaxDeltaCounterInit` is updated for list entries if the following conditions are met:
  - A triple of SWC, port, and data element is available in both the communication description and the list of configurable E2E Rx properties.
  - A triple of SWC, port, and data element is associated with an E2E protection description.
  - `MaxDeltaCounterInit` has not been changed.

- MaxDeltaCounterInit changed in the associated E2E protection description.

- E2EDescription is updated for list entries if the following conditions are met:
  - A triple of SWC, port, and data element is available in both the communication description and the list of configurable E2E Rx properties.
  - A triple of SWC, port and data element is associated with an E2E protection description.
  - The short name of the associated E2E protection description has been changed.

3.3.5. Generating code

To generate E2E Protection Wrapper code, follow the instructions of the EB tresos Studio user's guidein section `7.9.2. Generating a project`.

3.3.6. Code examples of the E2E Protection Wrapper

The usage of the `E2EPW` is distinguished between the sender and receiver side and consists of several steps. The code examples below are provided without warranty of any kind.

3.3.6.1. Sender SW-C (single channel wrapper)

```
/* Step S0: optionally initialize wrapper (if not done during ECU startup) */
(void)E2EPW_WriteInit_<p>_<o>();

/* Step S1: write Data Element */
AppDataEl.speed = U16_V_MAX;
AppDataEl.accel = U8_0;

/* Step S2: call E2EPW Write */
Ret = E2EPW_Write_<p>_<o>(&AppDataEl);

RetRteWrite = extractRetRteWrite(Ret);
RetE2EPW = extractRetE2EPW(Ret);
RetProtect = extractRetProtect(Ret);
RetStatus = extractRetStatus(Ret);

/* Step S9: perform error handling */
if((((RetRteWrite != RTE_E_OK) ||
     (RetE2EPW != E2E_E_OK)) ||
     (RetProtect != E2E_E_OK)) ||
```

```
     (RetStatus != E2E_E_OK))
{
   /* perform error handling */
}
```

3.3.6.2. Receiver SW-C (single channel wrapper)

```
/* Step R0: optionally initialize wrapper (if not done during ECU startup) */
(void)E2EPW_ReadInit_<p>_<o>();

/* Step R1: call E2EPW Read */
Ret = E2EPW_Read_<p>_<o>(&AppDataEl);

RetRteRead = extractRetRteRead(Ret);
RetE2EPW = extractRetE2EPW(Ret);
RetCheck = extractRetCheck(Ret);
RetCommStatus = extractRetCommStatus(Ret);

if (TRUE == isStartUp)
{
   /* check if data are available */
   if((RetRteRead == RTE_E_OK) &&
      (RetE2EPW == E2E_E_OK)) &&
      (RetCheck == E2E_E_OK)) &&
      (RetCommStatus == E2EPW_STATUS_INITIAL))
   {
      /* Startup phase is over, a valid message has been received */
      isStartUp = FALSE;
   }
   else
   {
       /* Step R0: Initialize E2EPW for Port1/DataElement1 as long as no valid data are
        * available */
      (void)E2EPW_ReadInit_<p>_<o>();
   }

   /* check for startup timeout if required */

}
else
{
   /* Step R9: check timeout and perform error handling */
```

```
   /* check Byte 0, Byte 1, and Byte 2 of return value */
   if((RetRteRead != RTE_E_OK) ||
      (RetE2EPW != E2E_E_OK) ||
      (RetCheck != E2E_E_OK))
   {
      /* check for timeout */

      /* perform error handling */
   }
   else
   {
      /* Check communication status */
      if((RetCommStatus != E2EPW_STATUS_OK) &&
         (RetCommStatus != E2EPW_STATUS_OKSOMELOST))
      {
         /* check for timeout */

         /* perform error handling */
      }
      else
      {
         /* use AppDataEl */
         swc_app(&AppDataEl);
      }
   }
}
```

3.3.6.3. Sender SW-C (redundant channel wrapper)

```
/* Step S0 / Step S10: optionally initialize wrapper (if not done during ECU startup) */
(void)E2EPW_WriteInit1_<p>_<o>();
(void)E2EPW_WriteInit2_<p>_<o>();

/* Step S1: write Data Element */
AppDataEl.speed = U16_V_MAX;
AppDataEl.accel = U8_0;

/* Step S2: call E2EPW Write1 */
Ret1 = E2EPW_Write1_<p>_<o>(&AppDataEl);

Ret1RteWrite = extractRetRteWrite(Ret1);
```

```
Ret1E2EPW = extractRetE2EPW(Ret1);
Ret1Protect = extractRetProtect(Ret1);
Ret1Status = extractRetStatus(Ret1);

/* Step S9: perform error handling */
if((((Ret1RteWrite != RTE_E_OK) ||
     (Ret1E2EPW != E2E_E_OK)) ||
     (Ret1Protect != E2E_E_OK)) ||
     (Ret1Status != E2E_E_OK))
{
   /* perform error handling */
}
else
{
   /* Step S11: write values again to data element */
   AppDataEl.speed = U16_V_MAX;
   AppDataEl.accel = U8_0;

   /* Step S12: call E2EPW Write2 */
   Ret2 = E2EPW_Write2_<p>_<o>(&AppDataEl);

   /* Step S19: error handling */
   if(Ret2 != Ret1)
   {
      /* perform error handling */
   }
}
```

3.3.6.4. Receiver SW-C (redundant channel wrapper)

```
/* Step R0 / Step R10: optionally initialize wrapper (if not done during ECU startup) */
(void)E2EPW_ReadInit1_<p>_<o>();
(void)E2EPW_ReadInit2_<p>_<o>();

/* Step R1: call E2EPW Read 1 */
Ret1 = E2EPW_Read1_<p>_<o>(&AppDataEl);

Ret1RteRead = extractRetRteRead(Ret1);
Ret1E2EPW = extractRetE2EPW(Ret1);
Ret1Check = extractRetCheck(Ret1);
Ret1CommStatus = extractRetCommStatus(Ret1);
```

```
if (TRUE == isStartUp)
{
   /* check if data are available */
   if((Ret1RteRead == RTE_E_OK) &&
      (Ret1E2EPW == E2E_E_OK)) &&
      (Ret1Check == E2E_E_OK)) &&
      (Ret1CommStatus == E2EPW_STATUS_INITIAL))
   {
      /* Startup phase is over, a valid message has been received */
      isStartUp = FALSE;
   }
   else
   {
      /* Step R0: Initialize E2EPW for Port1/DataElement1 as long as no valid data are
       * available */
      (void)E2EPW_ReadInit1_<p>_<o>();
      (void)E2EPW_ReadInit2_<p>_<o>();
   }

   /* check for startup timeout if required */

}
else
{
   /* Step R9: check timeout and perform error handling */
   /* check Byte 0, Byte 1, and Byte 2 of return value */
   if((Ret1RteRead != RTE_E_OK) ||
      (Ret1E2EPW != E2E_E_OK) ||
      (Ret1Check != E2E_E_OK))
   {
      /* check for timeout */

      /* perform error handling */
   }
   else
   {
      /* Check communication status */
      if((Ret1CommStatus != E2EPW_STATUS_OK) &&
         (Ret1CommStatus != E2EPW_STATUS_OKSOMELOST))
      {
         /* check for timeout */

         /* perform error handling */
      }
      else
      {
         /* copy values from data element */
```

```
            RedundantAppDataEl = AppDataEl;
            /* Step R11: call E2EPW Read 2 */
            Ret2 = E2EPW_Read2_<p>_<o>(&AppDataEl);

            /* Step R19: check timeout and perform error handling */
            if(Ret1 != Ret2)
            {
               /* check for timeout */

               /* perform error handling */
            }
            else
            {
               /* check for corruption of AppDataEl after CRC has been checked */
               if((AppDataEl.speed != RedundantAppDataEl.speed) ||
                     (AppDataEl.accel != RedundantAppDataEl.accel))
               {
                  /* perform error handling */
               }
               else
               {
                  /* use AppDataEl/RedundantAppDataEl */
                  swc_appRC(&AppDataEl, &RedundantAppDataEl);
               }
            }
         }
      }
   }
```

3.3.7. Support of VCC proprietary E2E profile

The E2EPW module supports the generation of E2EPW wrapper code with a different serialization technique to the serialized data transmitted over the communication bus. If the E2E Description specifies the proprietary profile ProfileVCC in the category tag then the data is serialized in alphabetic order with respect to the short names of the data element members for the calculation of the CRC value.

3.4. E2EPW integration notes

You find general integration information in the EB tresos AutoCore Generic documentation.

In addition, you find module-specific information about exclusive areas, production errors and memory mapping in the module-specific integration notes. You find the module-specific integration notes in the module references chapter of this document. See Chapter 4, “E2EPW module references“ sub-section `Integration notes` in each module.

3.5. Usage of E2EPW Config Plugins

Tool name: EB tresos Studio (E2EPW Config Plugins are executed by EB tresos Studio)

Tool location: <TresosBase>\bin\tresos_gui.exe or <TresosBase>\bin\tresos_gui_64.exe

For the documentation of EB tresos Studio relevant for the E2EPW, see [3].

E2EPW provides project templates to generate the wrapper files in a few minutes based on pre-delivered EB tresos Studio projects.

- Start EB tresos Studio by executing <TresosBase>\bin\tresos_gui.exe or <TresosBase>\bin\tresos_gui_-64.exe:
  - If EB tresos Studio asks to create a new workspace, select YES. EB tresos Studio will then create a workspace directory in <TresosBase>\workspace
  - If EB tresos Studio is started the first time, a welcome screen is opened which must be closed to have access to the project view.

3.5.1. EB tresos Studio template project for E2E Protection

E2EPW provides project templates to generate the wrapper files in a few minutes based on pre-delivered EB tresos Studio projects.

- Start EB tresos Studio by executing <TresosBase>\bin\tresos_gui.exe:
  - If EB tresos Studio asks to create a new workspace, select YES. EB tresos Studio will then create a workspace directory in <TresosBase>\workspace
  - If EB tresos Studio is started the first time, a welcome screen is opened which must be closed to have access to the project view.
- Import a template project:
  - Select File->Import

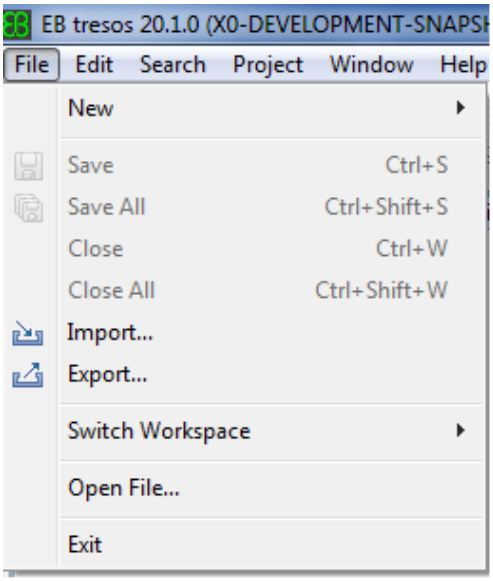

Figure 3.7. Import a template project

- Select "Existing Projects into Workspace" from the node "General"
- Press the "Next" button

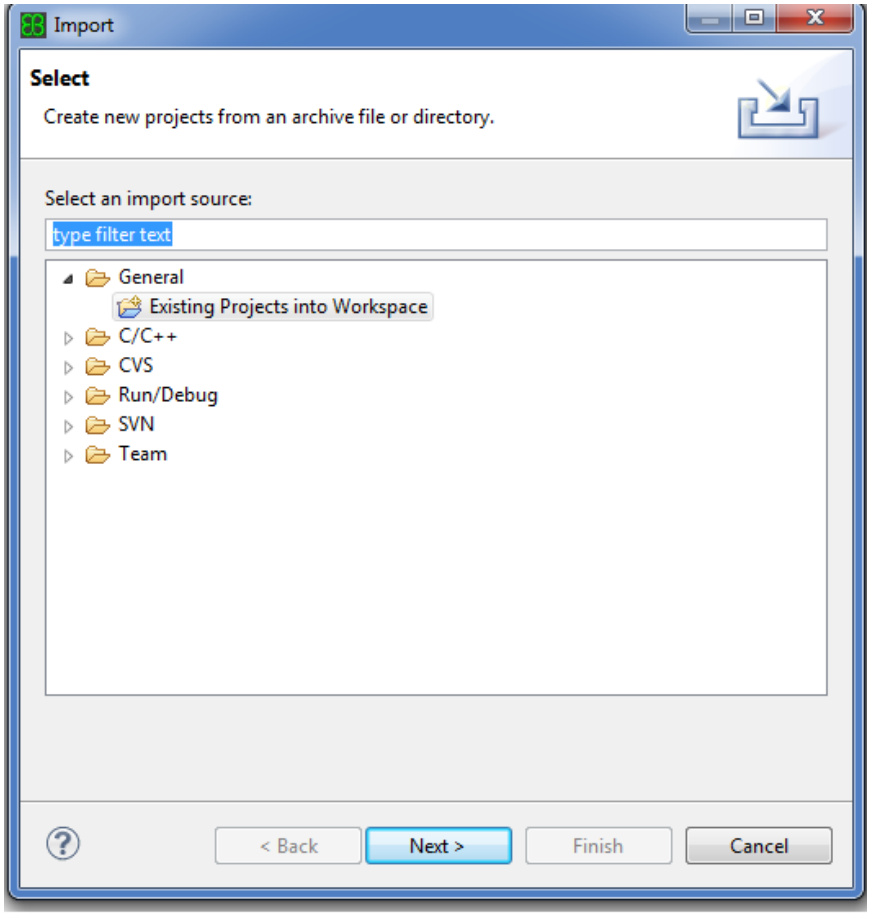

Figure 3.8. Import an existing project

- In the following window, select the archive file of EB tresos Studio template projects (<Tresos-Base>\templates)
- Press the "OK" button

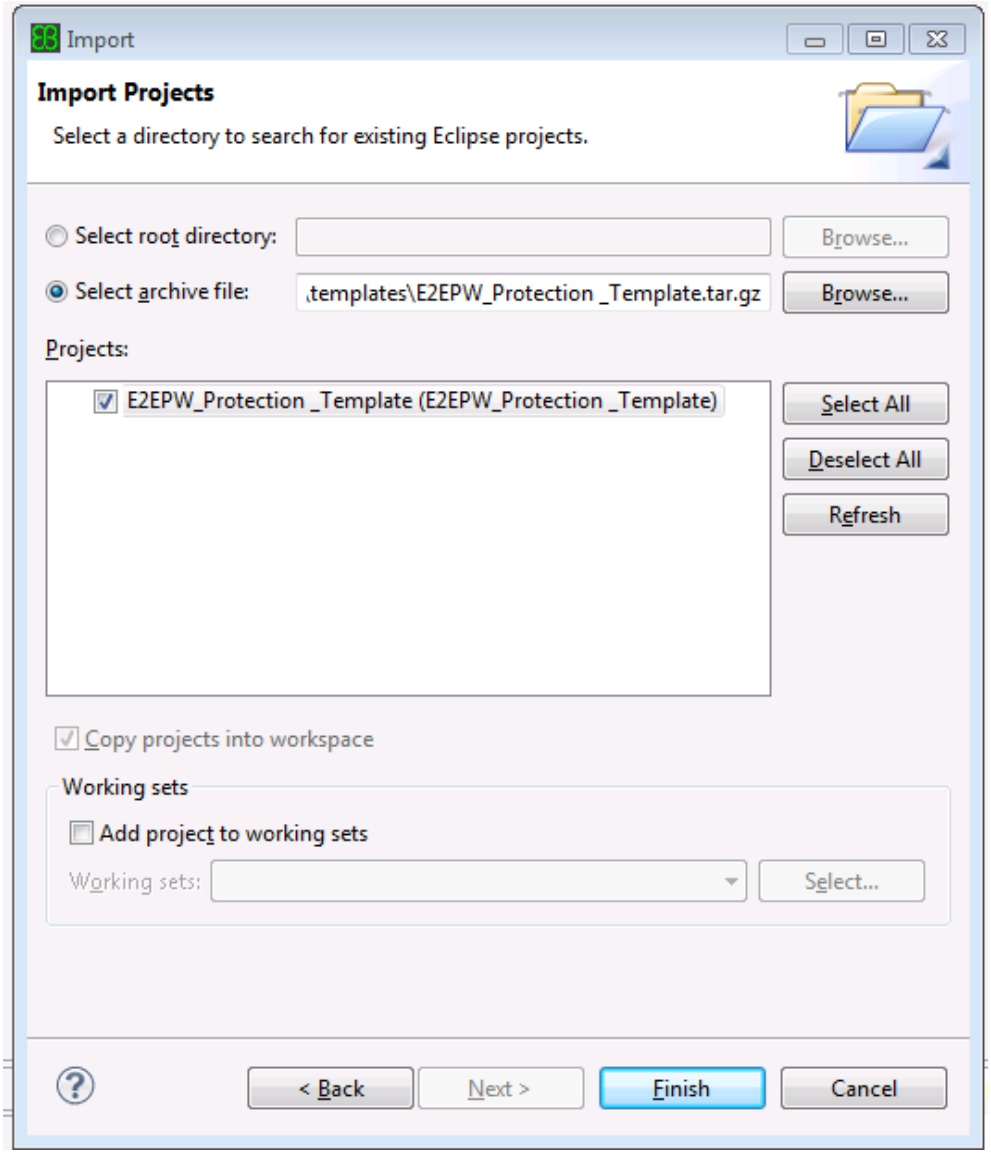

Figure 3.9. EB tresos Studio templates

- One of the following projects can be selected:
  - Project Name: E2EProtection_Template_SysD

    Description: This project template provides a pre-configured importer for ASR 4.0 / ASR 3.2 / 3.-1 projects with example ASR 4.0 System and Software Component Description files.
- Press the "Finish" button

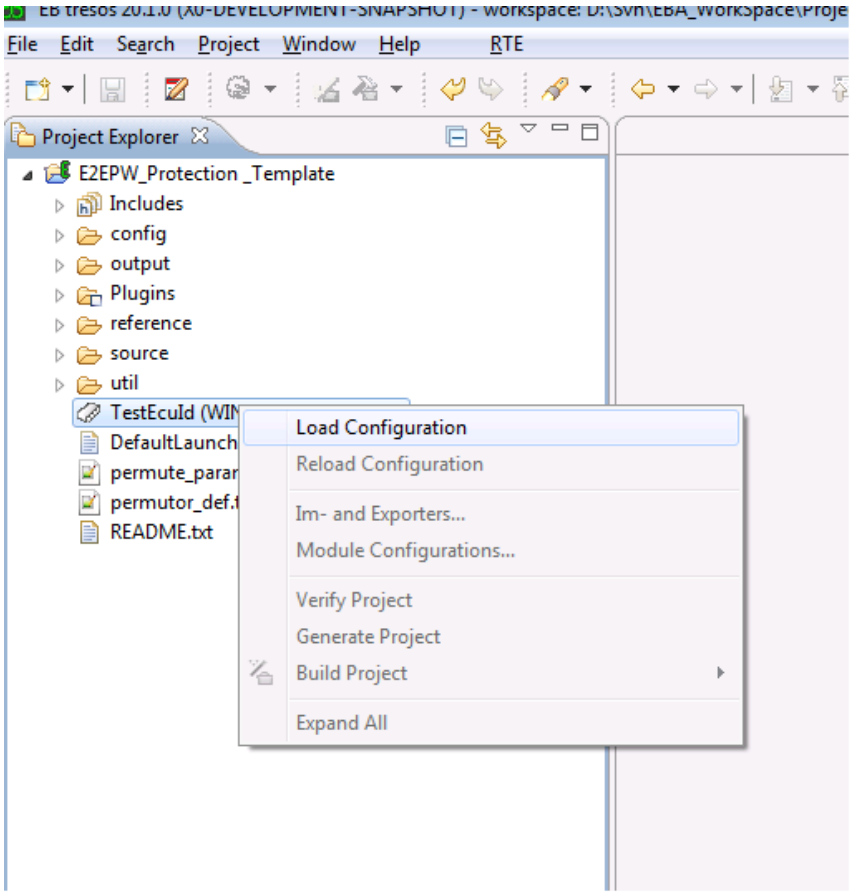

Figure 3.10. Selection of EB tresos Studio template project

- The template project now appears in the Project Explorer.
- Open this project with a double click and right click on it.
- Within the popup-window select the "Im- and Exporters ..." entry.

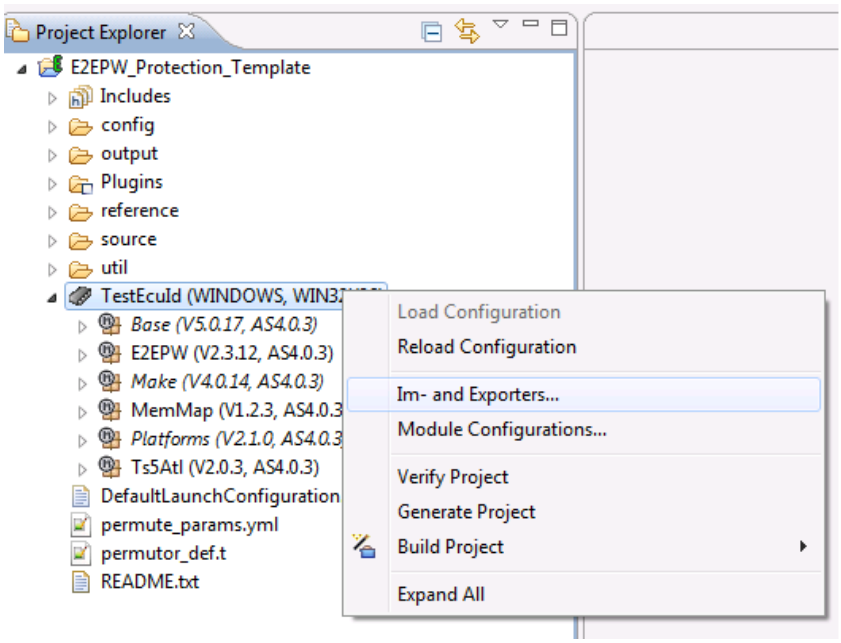

Figure 3.11. New Project Wizard

- Select the pre-configured importer and perform adjustments if required (e.g. select different System Description files within the "All Models" tab).
- press the "Run Importer" button
- A window "Warnings During Importer Run" pops up. Check that no unwanted warnings or errors are reported.
- Select "OK" to close the "Im- / and Export" window
- Right click on the project and select "Generate Project"
- The generated wrapper files can be found in the directory: "output/generated"

3.5.2. Import of configuration data

3.5.2.1. Import with ASR System Description

For a detailed description of the System Description Importer, see chapter "Importing AUTOSAR system descriptions" in [3].

- In the Project Explorer of the EB tresos main window right click on the new project "E2EProject" and select "Im- and Exporters" from the popup menu.

- Click on the green plus button situated in the top left corner of the window and the New Importer/Export window appears.
- Import Step 1 (see Figure 3.12, “System Description Import”):
  - Add all Autosar System Description files that shall be imported in this project via the button "Add".

    Note: The "Meta Model Version" should be automatically selected to the correct version. If there is a validation problem, then it is highly likely that the wrong Meta Model Version was selected.
  - Meta model version: 'AUTOSAR 3.1 (XSD Document Version 3.3.0 Revision 0004 (Daimler specific version 4), xmlns="http://autosar.org/3.1.4.DAI.4")'
  - Note: If multiple system description files are specified within one instance of the System Description Importer, then EB tresos Studio imports the files in alphabetical order of the absolute file system path / filename according to the ASCII characters (e.g. "B_file.arxml" is imported before "a_file.arxml"). An element which is defined across several files in the same import process is taken only from the file with the highest priority. For sake of clarity, it is recommended to add a number as a prefix to the imported files which specify the import order (e.g. "01_filename.arxml", "02_filename.arxml", etc.).
  - Activate "Validate against XML schema"
  - "Import Customization" shall keep "None"
  - Select the System and ECU instance which shall be imported
  - Press the "Next" button
  - Right click on the new project "E2EProject" and select "Im- and Exporters" from the popup menu
  - Click on the green plus button situated in the top left corner of the window and the New Importer/Export window appears.
  - Enter a name (e.g. SYSD)
  - Select the Importer/Exporter: 'System Description Importer (Autosar 2.1/3.0/3.1)'

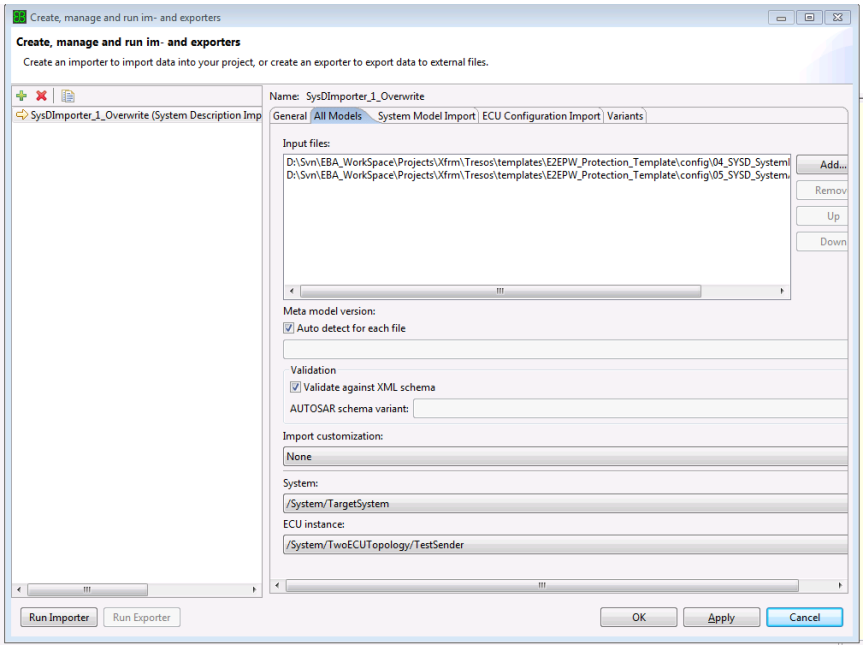

Figure 3.12. System Description Import

- Import Step 3 (see Figure 3.13, “System Description Import - Step 3”):
  - Enable the check-box "Enable system model import"
  - Enable the check-box "Overwrite existing model"
  - Press the "Apply" button
  - Press the "Run Importer" button
  - A window "Warnings During Importer Run" pops up. Check that no unwanted warnings or errors are reported.
  - Select "OK" to close the "Im- / and Export" window

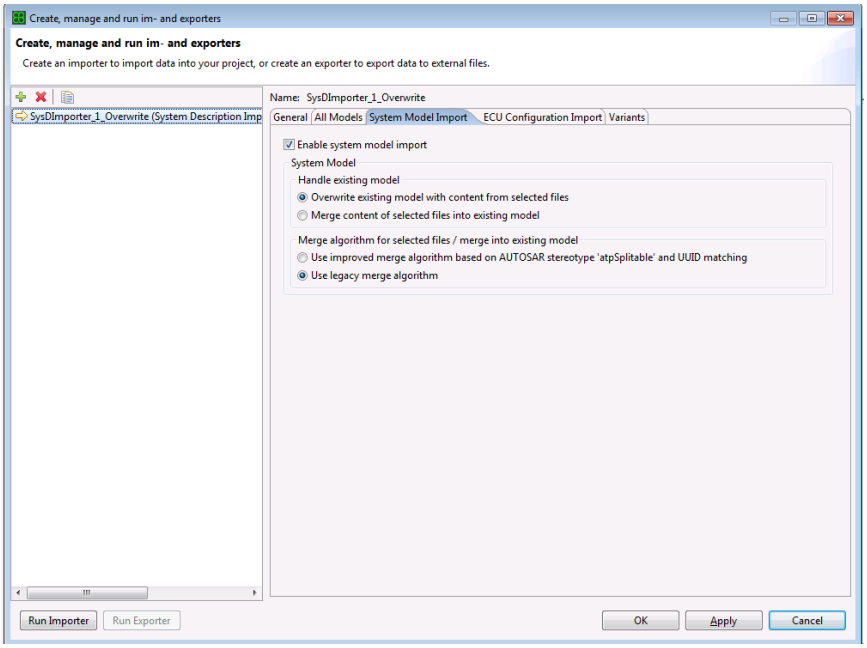

Figure 3.13. System Description Import - Step 3

3.5.3. Configure E2EPW

In the Project Explorer of the EB tresos main window double click on the E2EPW module in order to open the configuration window.

- Select the Wrapper Mode either SingleChannel or RedundantChannel.
- Select the if the RteIsUpdated feature shall be used.
- In case of protection of a pure intra-ECU communication, a list of CRC, Counter, and DataIDNibble (Profile 01 variant C only) prefixes must be configured for identifying the related CRC, Counter, and DataIDNibble member of a protected complex Data Element.

3.5.4. Extended Configuration

As an extended configuration the E2EPW Config Plugins allow to reconfigure the imported MaxDeltaCounterInit parameter. In contrast to ASR, this parameter can be configured per receiver and not only per E2E Protection.

The usage of a user specific E2E Rx property MaxDeltaCounterInit instead of values specified in the Communication Description is required if only a global default value for E2E Rx property MaxDeltaCounterInit is used in the OEM communication description, whereas an ECU specific value is defined in the requirements specification of the ECU.

The usage of a receiver specific definition of E2E Rx property MaxDeltaCounterInit could be required if a protected message shall be received by more than one receiver and at least two receivers have different ASIL levels assigned.

The E2E Rx property MaxDeltaCounterInit can be configured in the tab *Configurable E2E Rx Properties* of the E2EPW Config. As the parameter is specified per triple of protected software component type, port and data element this information has to be retrieved from the system configuration first. Therefore the wizard *Update list of configurable E2E Rx properties* need to be called. Figure 3.5, "**Update list of configurable E2E Rx properties** wizard" shows where this wizard can be started in EB tresos Studio. The wizard then creates for each E2E receiver a list entry and uses the MaxDeltaCounterInit value from the Communication Description as initial value. An example result is shown in Figure 3.6, "Result after the **Update list of configurable E2E Rx properties** wizard is called". Therein, the user can optionally edit the configurable parameter for each entry in this list (e.g. MaxDeltaCounterInit). Description of the list parameters:

- E2EProtection: Shortname of the E2E Protection information associated with SWC-Port-DataElement
- SWC: Shortname of the protected software component type
- Port: Shortname of the protected port associated to SWC
- DataElement: Shortname of the protected data element associated to Port
- MaxDeltaCounterInit: user definable value for the triple of SWC, Port and DataElement.

In case *Configurable E2E Rx Properties* already contains a list, the wizard performs a merge of the data from the Communication Description and the existing list according to the following merge policy:

- list entries which are no longer associated to any E2E Protection information of the imported Communication description are deleted.
- list entries are not updated if
  - triple of SWC, Port, DataElement is available in both, the Communication Description and the list of configurable E2E Rx properties, and
  - triple of SWC, Port, DataElement is associatied with a valid E2E protection description, and
  - MaxDeltaCounterInit has been changed by the user.
- MaxDeltaCounterInit will be updated for list entries where
  - triple of SWC, Port, DataElement is available in both, the Communication Description and the list of configurable E2E Rx properties, and
  - triple of SWC, Port, DataElement is associated with an E2E protection description, and
  - MaxDeltaCounterInit has not been changed by the user, and
  - MaxDeltaCounterInit changed in the associated E2E protection description.
- E2EDescription will be updated for list entries where
  - triple of SWC, Port, DataElement is available in both, the Communication Description and the list of configurable E2E Rx properties, and

- triple of SWC, Port, DataElement is associated with an E2E protection description, and
- the short name of the associated E2E protection description has been changed.

3.5.5. Generate Code

Generation of E2E Protection Wrappers:

- Right click on project and select "Generate Project"
- The generated files are now generated to <TresosBase>\workspace\<ProjectName>\output\generated\

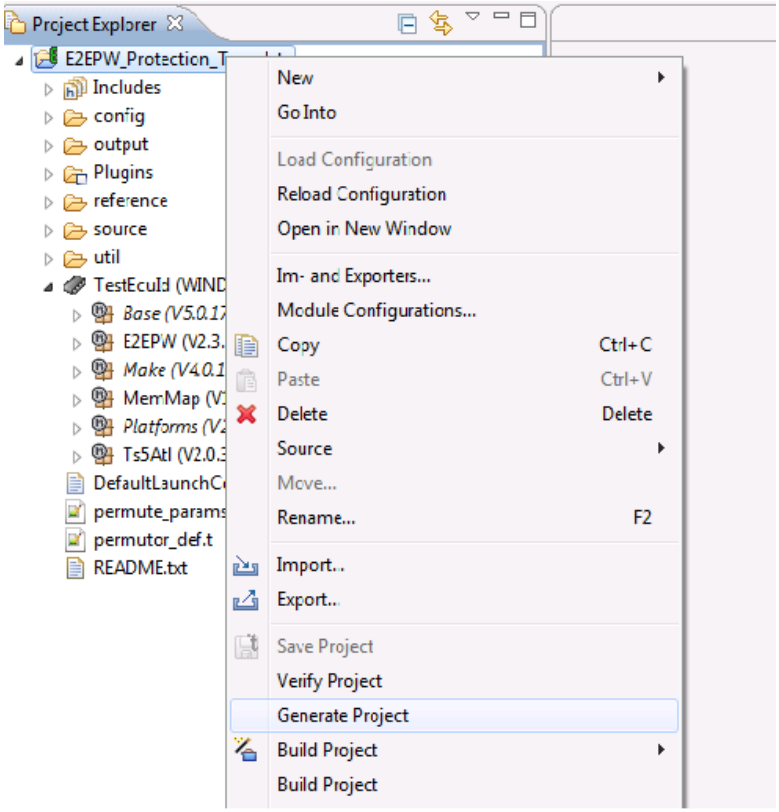

Figure 3.14. Generate Project

Note that there exists a template E2E Protection project in <TresosBase>\templates\E2EProtection_Template, which can be used for the first steps with EB tresos Studio for E2E Protection. This project can be simply imported in EB tresos Studio (importing an existing project is documented in [3]).

4. E2EPW module references

4.1. Overview

This chapter provides module references for the E2EPW 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 E2EPW user's guide.

4.1.1. 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.

4.1.1.1. 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.

4.1.1.2. 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.

4.2. E2EPW

4.2.1. 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.
VendorSpecific	1..1	
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

4.2.1.1. 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	2
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	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	2
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	3
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	20
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	0
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

4.2.1.2. VendorSpecific

Containers included		
Container name	**Multiplicity**	**Description**
ConfigurableE2ERxProperties	0..n	Unique short name for triple of software component type, port, data element.

Parameters included	
Parameter name	**Multiplicity**
GenerateE2EUnconnectedPorts	1..1
VerifyGeneratedCode	1..1
E2EByPass	1..1
WrapperMode	1..1
UsingRteIsUpdated	1..1
TxSignalGroupArrayAccess	1..1
CRCRegex	0..n
CounterRegex	0..n
DataIDNibbleRegex	0..n
CRCPrefix	0..n
CounterPrefix	0..n
DataIDNibblePrefix	0..n

Parameter Name	**GenerateE2EUnconnectedPorts**

Label	Enable Generation of E2E unconnected ports
Description	Activates/Deactivates the generation of E2E unconnected protected sender / receiver ports. This feature is only used for specific development purposes and shall be deactivated for mass production projects. ► `TRUE`: Generation of E2E unconnected ports is activated. In this case E2EPW code for pure Intra-ECU communication is generated for all unconnected ports that are referenced by an E2E description ► `FALSE`: Generation of E2E unconnected ports is deactivated. In this case no E2EPW code is generated for unconnected ports that are referenced by an E2E description
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**VerifyGeneratedCode**
Description	Activates/Deactivates the automatic code verification of generated code by call to E2EPWCheck tool at the post-generation step. ► `TRUE`: Automatic code verification is activated. ► `FALSE`: Automatic code verification is deactivated.
Multiplicity	1..1
Type	BOOLEAN
Default value	true

Parameter Name	**E2EByPass**
Label	Disable E2E Protection (E2E bypass)
Description	Activates/Deactivates the E2E Protection. ► `TRUE`: E2E Protection is deactivated. ► `FALSE`: E2E Protection is activated.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**WrapperMode**
Label	Wrapper Mode
Description	Globally defines if either single channel or redundant channel E2E Protection wrapper routines shall be generated.

	▶ `SingleChannel`: Single channel wrapper routines are generated. ▶ `RedundantChannel`: Redundant channel wrapper routines are generated.
Multiplicity	1..1
Type	ENUMERATION
Default value	SingleChannel
Range	SingleChannel RedundantChannel

Parameter Name	**UsingRteIsUpdated**
Label	Usage of RteIsUpdated
Description	Defines if the Rte API `Rte_IsUpdated_p_o()` shall be used for the E2E Protection. Note that this API is not available in all AUTOSAR versions. ▶ `RteIsUpdatedUnused`: `Rte_IsUpdated_p_o()` is not used. ▶ `RteIsUpdatedUsed`: `Rte_IsUpdated_p_o()` is used.
Multiplicity	1..1
Type	ENUMERATION
Default value	RteIsUpdatedUnused
Range	RteIsUpdatedUnused RteIsUpdatedUsed

Parameter Name	**TxSignalGroupArrayAccess**
Label	TxSignalGroupArrayAccess
Description	Activates/Deactivates the E2E data flow optimization at the sender-side. That is, Rte_WriteArray_p_o() is called with the serialized byte array instead of Rte_Write_p_o(). ▶ `TRUE`: E2E data flow optimization is activated. ▶ `FALSE`: E2E data flow optimization is deactivated.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**CRCRegex**
Description	List of CRC regular Expressions for identifying the CRC member of a protected complex Data Element in case of pure intra-ECU communication. If regular expressions apply on several different or no Data Element members, an error

	is reported and no wrappers are generated. For example, if the regex `CRC_` is configured and a complex Data Element contains the members `CRC_signal, SEQ_counter, and SIG_mysignal1`, then `CRC_signal` is identified to be the CRC member. This is a simple example but the full range of regular expression functionality is supported here.
Multiplicity	0..n
Type	STRING
Default value	CRC_

Parameter Name	**CounterRegex**
Description	List of Counter regular expressions for identifying the Counter member of a protected complex Data Element in case of pure intra-ECU communication. If regular expressions apply on several different or no Data Element members, an error is reported and no wrappers are generated. For example, if the regular expression `SEQ_` is configured and a complex Data Element contains the members `CRC_signal, SEQ_counter, and SIG_mysignal1`, then `SEQ_signal` is identified to be the Counter member. This is a simple example but the full range of regular expression functionality is supported here.
Multiplicity	0..n
Type	STRING
Default value	SEQ_

Parameter Name	**DataIDNibbleRegex**
Description	List of DataID Nibble Regular Expressions for identifying the DataID member of a protected complex Data Element in case of pure intra-ECU communication (only relevant for specific Profiles making use of explicit transmission of the DataID). If regular expressions apply on several different or no Data Element members, an error is reported and no wrappers are generated. For example, if the regex `DID_` is configured and a complex Data Element contains the members `CRC_signal, DID_counter, and SIG_mysignal1`, then `DID_counter` is identified to be the DataID Nibble member. This is a simple example but the full range of regular expression functionality is supported here.
Multiplicity	0..n
Type	STRING
Default value	DID_

Parameter Name	**CRCPrefix**
Description	List of CRC prefixes for identifying the CRC member of a protected complex Data Element in case of pure intra-ECU communication. If prefixes apply on several different or no Data Element members, an error is reported and no wrappers

	are generated. For example, if the prefix `CRC_` is configured and a complex Data Element contains the members `CRC_signal, SEQ_counter, and SIG_mysignal1`, then `CRC_signal` is identified to be the CRC member.
Multiplicity	0..n
Type	STRING
Default value	CRC_

Parameter Name	**CounterPrefix**
Description	List of Counter prefixes for identifying the Counter member of a protected complex Data Element in case of pure intra-ECU communication. If prefixes apply on several different or no Data Element members, an error is reported and no wrappers are generated. For example, if the prefix `SEQ_` is configured and a complex Data Element contains the members `CRC_signal, SEQ_counter, and SIG_mysignal1`, then `SEQ_signal` is identified to be the Counter member.
Multiplicity	0..n
Type	STRING
Default value	SEQ_

Parameter Name	**DataIDNibblePrefix**
Description	List of DataID Nibble prefixes for identifying the DataID member of a protected complex Data Element in case of pure intra-ECU communication (only relevant for specific Profiles making use of explicit transmission of the DataID). If prefixes apply on several different or no Data Element members, an error is reported and no wrappers are generated. For example, if the prefix `DID_` is configured and a complex Data Element contains the members `CRC_signal, DID_counter, and SIG_mysignal1`, then `DID_counter` is identified to be the DataID Nibble member.
Multiplicity	0..n
Type	STRING
Default value	DID_

4.2.1.3. ConfigurableE2ERxProperties

Parameters included	
Parameter name	**Multiplicity**
E2EProtection	1..1
SWC	1..1

Parameters included	
Port	1..1
DataElement	1..1
MaxDeltaCounterInit	1..1
MaxNoNewOrRepeatedData	1..1
SyncCounterInit	1..1

Parameter Name	E2EProtection
Description	Shortname of the E2E Protection element that defines the protection of this triple (SWC, Port, DataElement).
Multiplicity	1..1
Type	STRING

Parameter Name	SWC
Description	Shortname of the protected software component types.
Multiplicity	1..1
Type	STRING

Parameter Name	Port
Description	Shortname of the protected receiver port associated to the specified software componen type.
Multiplicity	1..1
Type	STRING

Parameter Name	DataElement
Description	Shortname of the protected data element associated to the specified port and software component type.
Multiplicity	1..1
Type	STRING

Parameter Name	MaxDeltaCounterInit
Description	E2E Description parameter MaxDeltaCounterInit associated to the receiver specified with the triple software component type, port, data element.
Multiplicity	1..1
Type	INTEGER
Default value	0

Parameter Name	MaxNoNewOrRepeatedData

Description	E2E Description parameter MaxNoNewOrRepeatedData associated to the receiver specified with the triple software component type, port, data element. This parameter defines the maximum amount of missing or repeated Data which the receiver does not expect to exceed under normal communication conditions. Note: In order to be application-wise backward compatible to the E2E_PxxCheck()-function of earlier AUTOSAR releases, this configuration parameter must be set to the maximum allowed counter value (i.e. 14 for Profile 1 variants, else 15).
Multiplicity	1..1
Type	INTEGER
Default value	15

Parameter Name	**SyncCounterInit**
Description	E2E Description parameter SyncCounterInit associated to the receiver specified with the triple software component type, port, data element. This parameter defines the number of Data required for validating the consistency of the counter that must be received with a valid counter (i.e. counter within the allowed lock-in range) after the detection of an unexpected behavior of a received counter. Note: In order to be application-wise backward compatible to the E2E_PxxCheck()-function of earlier AUTOSAR releases, this configuration parameter must be set to 0.
Multiplicity	1..1
Type	INTEGER
Default value	0

4.2.1.4. PublishedInformation

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

Parameter Name	**PbcfgMSupport**
Label	PbcfgM support
Description	Specifies whether or not the E2EPW can use the PbcfgM module for post-build support.

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

4.2.2. Application programming interface (API)

4.2.2.1. Functions

4.2.2.1.1. E2EPW_Read1_p_o

Purpose	Performs a safe explicit redundant channel read on a sender-receiver safety-related communication data element.	
Synopsis	`uint32 E2EPW_Read1_p_o ( E2EDataType * E2EPW_Data );`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (out)	`E2EPW_Data`	Parameter to pass back the received data. Parameter <data> must remain valid until the function call returns.
Return Value	Byte 0 (least significant byte) is the status of Rte_Read function or RTE_E_OK if Rte_Read() is not called because Rte_IsUpdate is activated and returns FALSE: RTE_E_OK Data read successfully or Rte_Read() is not called because Rte_IsUpdate is activated and returns FALSE. RTE_E_INVALID Data element invalid. RTE_E_MAX_AGE_EXCEEDED Data element outdated. RTE_E_NEVER_RECEIVED No data received since system start or partition restart. RTE_E_UNCONNECTED Indicates that the receiver port is not connected. Byte 1 is the status of runtime checks done within E2E Protection Wrapper function: E2E_E_INPUTERR_NULL At least one pointer parameter is a NULL pointer.	

	E2E_E_INPUTERR_WRONG At least one input parameter is erroneous. E2E_E_INTERR An internal library error has occurred. (e.g. error in program flow monitoring, invariant, or postcondition) E2EPW_E_DESERIALIZATION bit extension/expansion error(s) occurred E2E_E_OK Function completed successfully. Byte 2 is the return value of E2E_PXXCheck function: E2E_E_INPUTERR_NULL At least one pointer parameter is a NULL pointer. E2E_E_INPUTERR_WRONG At least one input parameter is erroneous. E2E_E_OK Function completed successfully. Byte 3 is the value of the verification status of the Data determined by the E2E Check function of the E2E library. E2EPW_STATUS_OK - OK: The new data has been received according to communication medium, the CRC is correct, the Counter is incremented by 1 with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that no Data has been lost since the last correct data reception. E2EPW_STATUS_OKSOMELOST - OK: The new data has been received according to communication medium, the CRC is correct, the Counter is incremented by DeltaCounter (1 < DeltaCounter <= MaxDeltaCounter) with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that some Data in the sequence have been probably lost since the last correct/initial reception, but this is within the configured tolerance range E2EPW_STATUS_NONEWDATA - Error: the Check function has been invoked but no new Data is not available since the last call, according to communication medium (e.g. RTE, COM). As a result, no E2E checks of Data have been consequently executed. E2EPW_STATUS_WRONGCRC - Error: The data has been received according to communication medium, but the CRC is incorrect. E2EPW_STATUS_INITIAL - Error: The new data has been received according to communication medium, the CRC is correct, but this is the first Data since the receiver's initialization or reinitialization, so the Counter cannot be verified yet. E2EPW_STATUS_REPEATED - Error: The new data has been received according to communication medium, the CRC is correct, but the Counter is identical to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST.

	E2EPW_STATUS_WRONGSEQUENCE - Error: The new data has been received according to communication medium, the CRC is correct, but the Counter Delta is too big (DeltaCounter > MaxDeltaCounter) with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that too many Data in the sequence have been probably lost since the last correct/initial reception. E2EPW_STATUS_DATAINVALID - NOT VALID: Is returned only if E2E Profile 01 is used and the E2E library recognized that all bits in the data (except for byte 0) are set to one. E2EPW_STATUS_SYNC - NOT VALID: The new data has been received after detection of an unexpected behaviour of counter. The data has a correct CRC and a counter within the expected range with respect to the most recent Data received, but the determined continuity check for the counter is not finalized yet.
Description	Performs a safe explicit read on a sender-receiver safety-related communication data element with data semantics. The function calls optionally the corresponding function RTE_IsUpdated. This is not supported by AUTOSAR but implemented. If enabled, the user must be aware that the second channel read always assumes an updated DataElement which may result into different states (e.g. different counter value) between channel 1 and channel 2 which requires a subsequent re-initialization of both channels. Then it calls the corresponding function RTE_Read, and then checks received data with E2E_PXXCheck.

4.2.2.1.2. E2EPW_Read2_p_o

Purpose	Performs a safe explicit redundant channel read on a sender-receiver safety-related communication data element.	
Synopsis	`uint32 E2EPW_Read2_p_o ( E2EDataType * E2EPW_Data );`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`E2EPW_Data`	Received protected data element Parameter <data> must remain valid until the function call returns.
Return Value	Byte 0 (least significant byte) is equal to RTE_E_OK (Rte_Read is not invoked) Byte 1 is the status of runtime checks done within E2E Protection Wrapper function: E2E_E_INPUTERR_NULL At least one pointer parameter is a NULL pointer. E2E_E_INPUTERR_WRONG At least one input parameter is erroneous.	

	E2E_E_INTERR An internal library error has occurred. (e.g. error in program flow monitoring, invariant, or postcondition) E2EPW_E_DESERIALIZATION bit extension/expansion error(s) occurred E2E_E_OK Function completed successfully. Byte 2 is the return value of E2E_PXXCheck function: E2E_E_INPUTERR_NULL At least one pointer parameter is a NULL pointer. E2E_E_INPUTERR_WRONG At least one input parameter is erroneous. E2E_E_OK Function completed successfully. Byte 3 is the value of the verification status of the Data determined by the E2E Check function of the E2E library. E2EPW_STATUS_OK - OK: The new data has been received according to communication medium, the CRC is correct, the Counter is incremented by 1 with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that no Data has been lost since the last correct data reception. E2EPW_STATUS_OKSOMELOST - OK: The new data has been received according to communication medium, the CRC is correct, the Counter is incremented by DeltaCounter (1 < DeltaCounter <= MaxDeltaCounter) with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that some Data in the sequence have been probably lost since the last correct/initial reception, but this is within the configured tolerance range E2EPW_STATUS_WRONGCRC - Error: The data has been received according to communication medium, but the CRC is incorrect. E2EPW_STATUS_INITIAL - Error: The new data has been received according to communication medium, the CRC is correct, but this is the first Data since the receiver's initialization or reinitialization, so the Counter cannot be verified yet. E2EPW_STATUS_REPEATED - Error: The new data has been received according to communication medium, the CRC is correct, but the Counter is identical to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. E2EPW_STATUS_WRONGSEQUENCE - Error: The new data has been received according to communication medium, the CRC is correct, but the Counter Delta is too big (DeltaCounter > MaxDeltaCounter) with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that too many Data in the sequence have been probably lost since the last correct/initial reception.

	E2EPW_STATUS_DATAINVALID - NOT VALID: Is returned only if E2E Profile 01 is used and the E2E library recognized that all bits in the data (except for byte 0) are set to one. E2EPW_STATUS_SYNC - NOT VALID: The new data has been received after detection of an unexpected behaviour of counter. The data has a correct CRC and a counter within the expected range with respect to the most recent Data received, but the determined continuity check for the counter is not finalized yet.
Description	The function re-checks the data received with corresponding function Read1 by means of execution of E2E_PXXCheck.

4.2.2.1.3. E2EPW_ReadInit1_p_o

Purpose	Initializes the E2EPW redundant channel 1 Receiver.
Synopsis	`Std_ReturnType` **`E2EPW_ReadInit1_p_o`** `( void );`
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Return Value	E2E_E_OK
Description	Initializes the redundant channel wrappers for the reception of a safety-related data element for channel 1.

4.2.2.1.4. E2EPW_ReadInit2_p_o

Purpose	Initializes the E2EPW Receiver.
Synopsis	`Std_ReturnType` **`E2EPW_ReadInit2_p_o`** `( void );`
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Return Value	E2E_E_OK
Description	Initializes the redundant channel 2 wrapper for the reception of a safety-related data element.

4.2.2.1.5. E2EPW_ReadInit_p_o

Purpose	Initializes the E2EPW Receiver.
Synopsis	`Std_ReturnType` **`E2EPW_ReadInit_p_o`** `( void );`

Sync/Async	Synchronous
Reentrancy	Non Reentrant
Return Value	E2E_E_OK
Description	Initializes the single channel wrapper for the reception of a safety-related data element.

4.2.2.1.6. E2EPW_Read_p_o

Purpose	Performs a safe explicit single channel read on a sender-receiver safety-related communication data element.	
Synopsis	`uint32 E2EPW_Read_p_o ( E2EDataType * E2EPW_Data );`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (out)	`E2EPW_Data`	Parameter to pass back the received data. Parameter <data> must remain valid until the function call returns.
Return Value	Byte 0 (least significant byte) is the status of Rte_Read function or RTE_E_OK if Rte_Read() is not called because Rte_IsUpdate is activated and returns FALSE: RTE_E_OK Data read successfully or Rte_Read() is not called because Rte_IsUpdate is activated and returns FALSE. RTE_E_INVALID Data element invalid. RTE_E_MAX_AGE_EXCEEDED Data element outdated. RTE_E_NEVER_RECEIVED No data received since system start or partition restart. RTE_E_UNCONNECTED Indicates that the receiver port is not connected. Byte 1 is the status of runtime checks done within E2E Protection Wrapper function: E2E_E_INPUTERR_NULL At least one pointer parameter is a NULL pointer. E2E_E_INPUTERR_WRONG At least one input parameter is erroneous. E2E_E_INTERR An internal library error has occurred. (e.g. error in program flow monitoring, invariant, or postcondition) E2EPW_E_DESERIALIZATION bit extension/expansion error(s) occurred E2E_E_OK Function completed successfully.	

	Byte 2 is the return value of E2E_PXXCheck function: E2E_E_INPUTERR_NULL At least one pointer parameter is a NULL pointer. E2E_E_INPUTERR_WRONG At least one input parameter is erroneous. E2E_E_OK Function completed successfully. Byte 3 is the value of the verification status of the Data determined by the E2E Check function of the E2E library. E2EPW_STATUS_OK - OK: The new data has been received according to communication medium, the CRC is correct, the Counter is incremented by 1 with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that no Data has been lost since the last correct data reception. E2EPW_STATUS_OKSOMELOST - OK: The new data has been received according to communication medium, the CRC is correct, the Counter is incremented by DeltaCounter (1 < DeltaCounter <= MaxDeltaCounter) with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that some Data in the sequence have been probably lost since the last correct/initial reception, but this is within the configured tolerance range E2EPW_STATUS_NONEWDATA - Error: the Check function has been invoked but no new Data is not available since the last call, according to communication medium (e.g. RTE, COM). As a result, no E2E checks of Data have been consequently executed. E2EPW_STATUS_WRONGCRC - Error: The data has been received according to communication medium, but the CRC is incorrect. E2EPW_STATUS_INITIAL - Error: The new data has been received according to communication medium, the CRC is correct, but this is the first Data since the receiver's initialization or reinitialization, so the Counter cannot be verified yet. E2EPW_STATUS_REPEATED - Error: The new data has been received according to communication medium, the CRC is correct, but the Counter is identical to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. E2EPW_STATUS_WRONGSEQUENCE - Error: The new data has been received according to communication medium, the CRC is correct, but the Counter Delta is too big (DeltaCounter > MaxDeltaCounter) with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that too many Data in the sequence have been probably lost since the last correct/initial reception.

	E2EPW_STATUS_DATAINVALID - NOT VALID: Is returned only if E2E Profile 01 is used and the E2E library recognized that all bits in the data (except for byte 0) are set to one. E2EPW_STATUS_SYNC - NOT VALID: The new data has been received after detection of an unexpected behaviour of counter. The data has a correct CRC and a counter within the expected range with respect to the most recent Data received, but the determined continuity check for the counter is not finalized yet.
Description	Performs a safe explicit read on a sender-receiver safety-related communication data element with data semantics. The function calls optionally the corresponding function RTE_IsUpdated. Then it calls the corresponding function RTE_Read, and then checks received data with E2E_PXXCheck.

4.2.2.1.7. E2EPW_Write1_p_o

Purpose	Initiates a safe explicit redundant channel sender-receiver transmission of a safety-related data element.	
Synopsis	`uint32 E2EPW_Write1_p_o ( E2EDataType * E2EPW_Data );`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in,out)	`E2EPW_Data`	Data element to be protected
Return Value	Byte 0 (least significant byte) is equal to RTE_E_OK (Rte_Write is not invoked) Byte 1 is the status of runtime checks done within E2E Protection Wrapper function: E2E_E_INPUTERR_NULL At least one pointer parameter is a NULL pointer. E2E_E_INPUTERR_WRONG At least one input parameter is erroneous. E2E_E_OK Function completed successfully. Byte 2 is the return value of E2E_PXXProtect function: E2E_E_INPUTERR_NULL At least one pointer parameter is a NULL pointer. E2E_E_INPUTERR_WRONG At least one input parameter is erroneous. E2E_E_OK Function completed successfully. Byte 3 is equal to E2E_E_OK.	
Description	It protects data with E2E Library function E2E_PXXProtect. It does not call the corresponding RTE_Write function.	

4.2.2.1.8. E2EPW_Write2_p_o

Purpose	Initiates a safe explicit redundant channel sender-receiver transmission of a safety-related data element.	
Synopsis	`uint32` **`E2EPW_Write2_p_o`** `( E2EDataType * E2EPW_Data );`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`E2EPW_Data`	Protected data element to be sent
Return Value	Byte 0 (least significant byte) is the status of Rte_Write function: RTE_E_OK Data passed to communication service successfully. RTE_E_COM_STOPPED The RTE could not perform the operation because the COM service is currently not available (inter ECU communication only). RTE_E_SEG_FAULT A segmentation violation is detected in the handed over parameters to the RTE API. No transmission is executed. Byte 1 is the status of runtime checks done within E2E Protection Wrapper function: E2E_E_INPUTERR_NULL At least one pointer parameter is a NULL pointer. E2E_E_INPUTERR_WRONG At least one input parameter is erroneous. E2EPW_E_REDUNDANCY The control fields computed by Write1 and Write2 are not equal. E2E_E_OK Function completed successfully. Byte 2 is the return value of E2E_PXXProtect function: E2E_E_INPUTERR_NULL At least one pointer parameter is a NULL pointer. E2E_E_INPUTERR_WRONG At least one input parameter is erroneous. E2E_E_OK Function completed successfully. Byte 3 is equal to E2E_E_OK.	
Description	Initiates a safe explicit sender-receiver transmission of a safety-related data element with data semantic. It protects data with E2E Library function E2E_PXXProtect, compares the computed control fields with the ones computed by Write1, and then it calls the corresponding RTE_Write function.	

4.2.2.1.9. E2EPW_WriteInit1_p_o

Purpose	Initializes the E2EPW Sender.
Synopsis	`Std_ReturnType` **`E2EPW_WriteInit1_p_o`** `( void );`
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Return Value	E2E_E_OK
Description	Initializes the redundant channel 1 wrapper for the transmission of a safety-related data element.

4.2.2.1.10. E2EPW_WriteInit2_p_o

Purpose	Initializes the E2EPW Sender.
Synopsis	`Std_ReturnType` **`E2EPW_WriteInit2_p_o`** `( void );`
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Return Value	E2E_E_OK
Description	Initializes the redundant channel 2 wrapper for the transmission of a safety-related data element.

4.2.2.1.11. E2EPW_WriteInit_p_o

Purpose	Initializes the E2EPW Sender.
Synopsis	`Std_ReturnType` **`E2EPW_WriteInit_p_o`** `( void );`
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Return Value	E2E_E_OK
Description	Initializes the single channel wrapper for the transmission of a safety-related data element.

4.2.2.1.12. E2EPW_Write_p_o

Purpose	Initiates a safe explicit single channel sender-receiver transmission of a safety-related data element.	
Synopsis	`uint32` **`E2EPW_Write_p_o`** `( E2EDataType * E2EPW_Data );`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in,out)	`E2EPW_Data`	Data element to be protected and sent
Return Value	Byte 0 (least significant byte) is the status of Rte_Write function: RTE_E_OK Data passed to communication service successfully. RTE_E_COM_STOPPED The RTE could not perform the operation because the COM service is currently not available (inter ECU communication only). RTE_E_SEG_FAULT A segmentation violation is detected in the handed over parameters to the RTE API. No transmission is executed. Byte 1 is the status of runtime checks done within E2E Protection Wrapper function: E2E_E_INPUTERR_NULL At least one pointer parameter is a NULL pointer. E2E_E_INPUTERR_WRONG At least one input parameter is erroneous. E2E_E_INTERR An internal library error has occurred. (e.g. error in program flow monitoring, invariant, or postcondition) E2E_E_OK Function completed successfully. Byte 2 is the return value of E2E_PXXProtect function: E2E_E_INPUTERR_NULL At least one pointer parameter is a NULL pointer. E2E_E_INPUTERR_WRONG At least one input parameter is erroneous. E2E_E_OK Function completed successfully. Byte 3 is equal to E2E_E_OK.	
Description	Initiates a safe explicit sender-receiver transmission of a safety-related data element with data semantic. It protects data with E2E Library function E2E_PXXProtect and then it calls the corresponding RTE_Write function.	

4.2.3. Integration notes

4.2.3.1. Exclusive areas

Exclusive areas are not used by the `E2EPW` module.

4.2.3.2. Production errors

Production errors are not reported by the `E2EPW` module.

4.2.3.3. 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
<Atomic-SWC-Name>_VAR_INIT_UNSPECIFIED
RC_<Atomic-SWC-Name>_VAR_INIT_UNSPECIFIED

4.2.3.4. 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 guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the E2EPW module.

5. Bibliography

Bibliography

[1] *AUTOSAR Specification of SW-C End-to-End Communication Protection Library*, Issue Version 2.0.0, Release 4.0.3, Revision 0003, Publisher: AUTOSAR

[2] *INTERNATIONAL STANDARD ISO 26262*, Road vehicles - Functional safety, 2011

[3] *Documentation*, EB Tresos Studio Documentation, 2017