Document Title	Specification of Bus Mirroring
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	873

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R24-11

Document Change History

Date	Release	Changed by	Description
2024-11-27	R24-11	AUTOSAR Release Management	• Support for serialization to CAN FD
2023-11-23	R23-11	AUTOSAR Release Management	• Changed document name to include "CP" • Removed direct references to tables from SWS items • Added information about automatic handle IDs to configuration • `Mirror_NetworkType` changed to `uint8`
2022-11-24	R22-11	AUTOSAR Release Management	• Support for CAN XL
2021-11-25	R21-11	AUTOSAR Release Management	• Added detailed change history
2020-11-30	R20-11	AUTOSAR Release Management	• Improved structure of error sections • Replaced error descriptions with generated tables • Multi-partition support finalized • Replaced Mirror_CanIdType and Mirror_FlexRayChannelType by native types

▽

△

2019-11-28	R19-11	AUTOSAR Release Management	• Added multi-partition support {DRAFT} • Fixed configurable number of PDUs • Reworked requirements to avoid references to sections • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Initial release

Disclaimer

Contents

Known Limitations of the Current Document

Sequence diagrams and other diagrams have not yet been modeled in the BSW UML model, wherefore Chapter 9 is still empty.

1 Introduction and Functional Overview

This specification describes the functionality, the API, and the configuration for the AUTOSAR Basic Software module Bus Mirroring.

The purpose of the Bus Mirroring module is the replication of the traffic and the state of internal buses to an external bus, such that a tester connected to that external bus can monitor internal buses for debugging purposes.

The monitored traffic can be configured by the tester using diagnostic commands to the intermediate ECUs (gateways, controllers of sub-buses). Using the diagnostics protocol ensures that mirroring cannot be enabled without passing security checks.

The terms `Bus` and `Network` are used as synonyms within this specification. In most AUTOSAR specifications, the term `Network` is preferred, and therefore it is used when referring to API parameters, to the configuration, or to the protocol layout. On the other hand, the module is called Bus Mirroring, and because of this the term `Bus` is used when the mirroring direction is considered, like in “source bus” or “destination bus”.

2 Acronyms and Abbreviations

Currently, the Bus Mirroring module does not define any acronyms, abbreviations, or terms that are not defined in the [1, AUTOSAR glossary].

3 Related Documentation

3.1 Input Documents & Related Standards and Norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[3] Requirements on Bus Mirroring
AUTOSAR_CP_RS_BusMirroring

[4] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[5] System Template
AUTOSAR_CP_TPS_SystemTemplate

3.2 Related Specification

AUTOSAR provides a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for the Bus Mirroring module.

Thus, the specification SWS BSW General shall be considered as additional and required specification for the Bus Mirroring module.

4 Constraints and Assumptions

4.1 Limitations

The Bus Mirroring module cannot be used to influence the traffic on one of the buses configured as a source bus. To ensure this and to avoid loop-back of messages leading to bus overload, the generation tool shall ensure that no bus is connected to the Bus Mirroring module both as source and destination bus (see [SWS_Mirror_00001]).

The Bus Mirroring module is controlled by a diagnostic control application through the dedicated (service) API listed in Chapter 8. The control functionality is made accessible to a diagnostic tester by special diagnostic services, which are handled by the DCM and implemented by the diagnostic control application. The DCM provides the necessary security to exclude inadvertent activation of the Bus Mirroring. The Bus Mirroring module does not provide another control interface, and it does not receive control messages on the destination bus.

In general, the Bus Mirroring module does not support source buses that have a larger frame size or more additional information than the destination bus can carry, e.g. CAN XL to CAN FD, CAN FD to CAN, CAN to LIN, FlexRay to CAN or CAN FD, Ethernet to CAN, or Ethernet to FlexRay. The Bus Mirroring module does not fragment mirrored frames.

The Bus Mirroring module will only mirror traffic that is actually received or transmitted by the bus interface modules. For CAN this means that besides the transmitted frames only those data frames that pass the hardware filter will be mirrored, and that remote frames and error frames will not be mirrored. For LIN, slave-to-slave communication will not be mirrored by a LIN master. And for FlexRay, only transmitted frames and those received frames for which reception buffers are assigned (possibly as a FIFO) will be mirrored.

Another limitation of the mirroring from a FlexRay source bus concerns the reported time stamps and cycles. The `Timestamp` reported for a FlexRay frame contains the time when the corresponding job list entry was executed. The actual transmission time has to be calculated from the slot ID contained in the reported `FrameID`. The cycle contained in the reported `FrameID` is accurate only for received frames and frames transmitted in the static segment. For frames transmitted in the dynamic segment, the reported cycle can be inaccurate because it can happen that a frame cannot be transmitted in the expected cycle, it is then deferred to the next suitable cycle.

A re-serialization of received serialized frames shall not be done by the Bus Mirroring module, because that would require too much resources. Instead, the serialized PDUs shall be routed directly to the destination bus.

The Bus Mirroring module will also not support the forwarding from Ethernet to Ethernet. This use case is already covered by the Port Mirroring feature of the AUTOSAR Ethernet Switch Driver.

4.2 Applicability to Car Domains

The Bus Mirroring module can be used in all kinds of vehicles that feature external CAN and/or Ethernet connectors, e.g. a Diagnostic connector.

5 Dependencies to Other Modules

The Bus Mirroring module has interfaces towards the CAN Interface (CanIf), the LIN Interface (LinIf), the FlexRay Interface (FrIf), the PDU Router (PduR), the Default Error Tracer (DET), and the diagnostic application, which accesses either the service port API via the AUTOSAR Runtime Environment (RTE) or the Complex Drivers (CDD) API of the Bus Mirroring module.

The Bus Mirroring module includes header files of CanIf, LinIf, FrIf, PduR, DET, StbM, and the RTE.

5.1 File Structure

This section explains the file structure of the Bus Mirroring module.

5.1.1 Code File Structure

For details, refer to the Section 5.1.6 “Code file structure” in [2, SWS BSW General].

5.1.2 Header File Structure

Besides the files defined in Section 5.1.7 “Header file structure” in [2, SWS BSW General], the Bus Mirroring module needs to include the files defined below.

[SWS_Mirror_00142]

Upstream requirements: SRS_Mirror_00001

⌈The Bus Mirroring module shall include the header file `CanIf.h` if at least one `MirrorSourceNetworkCan` is configured.⌋

[SWS_Mirror_00143]

Upstream requirements: SRS_Mirror_00001

⌈The Bus Mirroring module shall include the header file `LinIf.h` if at least one `MirrorSourceNetworkLin` is configured.⌋

[SWS_Mirror_00144]

Upstream requirements: SRS_Mirror_00001

⌈The Bus Mirroring module shall include the header file `FrIf.h` if at least one `MirrorSourceNetworkFlexRay` is configured.⌋

[SWS_Mirror_00147]

Upstream requirements: SRS_Mirror_00001

⌈The Bus Mirroring module shall include the header file `StbM.h` if at least one `MirrorDestNetworkFlexRay`, `MirrorDestNetworkCanXL`, `MirrorDestNetworkIp`, or `MirrorDestNetworkCdd` is configured, or if at least one `MirrorDestNetworkCanFD` is configured where `MirrorDestProtocolType` is not set to `MIRROR_PT_NONE`.⌋

6 Requirements Tracing

The following table references the requirements specified in [3, RS Bus Mirroring] and [4, RS BSW General] and links to the fulfillment of these.

Requirement	Description	Satisfied by
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[SWS_Mirror_00003]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/ disabling of detection and reporting of development errors.	[SWS_Mirror_00004] [SWS_Mirror_00005]
[SRS_BSW_00385]	List possible error notifications	[SWS_Mirror_00007] [SWS_Mirror_00008]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_Mirror_00004] [SWS_Mirror_00005] [SWS_Mirror_00113] [SWS_Mirror_00120] [SWS_Mirror_00137] [SWS_Mirror_00138] [SWS_Mirror_00150] [SWS_Mirror_00151] [SWS_Mirror_00153] [SWS_Mirror_00154] [SWS_Mirror_00158]
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_Mirror_00002]
[SRS_BSW_00450]	A Main function of a un-initialized module shall return immediately	[SWS_Mirror_00004]
[SRS_BSW_00452]	Classification of runtime errors	[SWS_Mirror_00008]
[SRS_BSW_00459]	It shall be possible to concurrently execute a service offered by a BSW module in different partitions	[SWS_Mirror_00166] [SWS_Mirror_00167] [SWS_Mirror_00168] [SWS_Mirror_00169]
[SRS_BSW_00461]	Modules called by generic modules shall satisfy all interfaces requested by the generic module	[SWS_Mirror_01029]
[SRS_BSW_00462]	All Standardized Autosar Interfaces shall have unique requirement Id / number	[SWS_Mirror_01033]
[SRS_BSW_00478]	Timing limits of main functions	[SWS_Mirror_00006]
[SRS_BSW_00480]	Null pointer errors shall follow a naming rule	[SWS_Mirror_00007]
[SRS_BSW_00481]	Invalid configuration set selection errors shall follow a naming rule	[SWS_Mirror_00007]
[SRS_BSW_00482]	Get version information function shall follow a naming rule	[SWS_Mirror_01005]
[SRS_BSW_00483]	BSW Modules shall handle buffer alignments internally	[SWS_Mirror_01024]
[SRS_BSW_00484]	Input parameters of scalar and enum types shall be passed as a value.	[SWS_Mirror_01006] [SWS_Mirror_01007] [SWS_Mirror_01008]
[SRS_BSW_00485]	Input parameters of structure type shall be passed as a reference to a constant structure	[SWS_Mirror_01003]
[SRS_BSW_00486]	Input parameters of array type shall be passed as a reference to the constant array base type	[SWS_Mirror_01024] [SWS_Mirror_01026] [SWS_Mirror_01027] [SWS_Mirror_01029]
[SRS_Mirror_00001]	The source and destination buses shall be configurable	[SWS_Mirror_00001] [SWS_Mirror_00142] [SWS_Mirror_00143] [SWS_Mirror_00144] [SWS_Mirror_00147]
[SRS_Mirror_00005]	The Bus Mirroring module shall provide an interface for module initialization	[SWS_Mirror_00002] [SWS_Mirror_00009] [SWS_Mirror_00013] [SWS_Mirror_00016]

△

Requirement	Description	Satisfied by
[SRS_Mirror_00006]	The Bus Mirroring module shall collect incoming frames	[SWS_Mirror_00021] [SWS_Mirror_00029] [SWS_Mirror_00038]
[SRS_Mirror_00007]	The Bus Mirroring module shall filter incoming frames	[SWS_Mirror_00017] [SWS_Mirror_00018] [SWS_Mirror_00021] [SWS_Mirror_00022] [SWS_Mirror_00023] [SWS_Mirror_00024] [SWS_Mirror_00025] [SWS_Mirror_00029] [SWS_Mirror_00030] [SWS_Mirror_00031] [SWS_Mirror_00032] [SWS_Mirror_00033] [SWS_Mirror_00038] [SWS_Mirror_00039] [SWS_Mirror_00040]
[SRS_Mirror_00008]	The Bus Mirroring module shall serialize incoming frames and bus states	[SWS_Mirror_00026] [SWS_Mirror_00034] [SWS_Mirror_00035] [SWS_Mirror_00041] [SWS_Mirror_00042] [SWS_Mirror_00043] [SWS_Mirror_00044] [SWS_Mirror_00045] [SWS_Mirror_00046] [SWS_Mirror_00047] [SWS_Mirror_00048] [SWS_Mirror_00049] [SWS_Mirror_00050] [SWS_Mirror_00055] [SWS_Mirror_00056] [SWS_Mirror_00057] [SWS_Mirror_00058] [SWS_Mirror_00059] [SWS_Mirror_00060] [SWS_Mirror_00061] [SWS_Mirror_00062] [SWS_Mirror_00063] [SWS_Mirror_00064] [SWS_Mirror_00065] [SWS_Mirror_00066] [SWS_Mirror_00067] [SWS_Mirror_00068] [SWS_Mirror_00069] [SWS_Mirror_00070] [SWS_Mirror_00071] [SWS_Mirror_00072] [SWS_Mirror_00073] [SWS_Mirror_00074] [SWS_Mirror_00075] [SWS_Mirror_00076] [SWS_Mirror_00077] [SWS_Mirror_00078] [SWS_Mirror_00079] [SWS_Mirror_00080] [SWS_Mirror_00081] [SWS_Mirror_00082] [SWS_Mirror_00083] [SWS_Mirror_00084] [SWS_Mirror_00085] [SWS_Mirror_00086] [SWS_Mirror_00087] [SWS_Mirror_00088] [SWS_Mirror_00089] [SWS_Mirror_00090] [SWS_Mirror_00091] [SWS_Mirror_00092] [SWS_Mirror_00093] [SWS_Mirror_00094] [SWS_Mirror_00095] [SWS_Mirror_00096] [SWS_Mirror_00097] [SWS_Mirror_00098] [SWS_Mirror_00099] [SWS_Mirror_00100] [SWS_Mirror_00101] [SWS_Mirror_00102] [SWS_Mirror_00103] [SWS_Mirror_00104] [SWS_Mirror_00105] [SWS_Mirror_00106] [SWS_Mirror_00107] [SWS_Mirror_00108] [SWS_Mirror_00109] [SWS_Mirror_00110] [SWS_Mirror_00111] [SWS_Mirror_00112] [SWS_Mirror_00146] [SWS_Mirror_00159] [SWS_Mirror_00170]
[SRS_Mirror_00009]	The Bus Mirroring module shall create a status frame	[SWS_Mirror_00026] [SWS_Mirror_00034] [SWS_Mirror_00035] [SWS_Mirror_00041] [SWS_Mirror_00042] [SWS_Mirror_00123] [SWS_Mirror_00124] [SWS_Mirror_00125] [SWS_Mirror_00126] [SWS_Mirror_00127] [SWS_Mirror_00128] [SWS_Mirror_00129] [SWS_Mirror_00131] [SWS_Mirror_00132] [SWS_Mirror_00133] [SWS_Mirror_00134] [SWS_Mirror_00135] [SWS_Mirror_00136] [SWS_Mirror_00146] [SWS_Mirror_00149]

▽

△

Requirement	Description	Satisfied by
[SRS_Mirror_00010]	The Bus Mirroring module shall provide an interface to control the mirroring state	[SWS_Mirror_00012] [SWS_Mirror_00014] [SWS_Mirror_00015] [SWS_Mirror_00019] [SWS_Mirror_00020] [SWS_Mirror_00027] [SWS_Mirror_00028] [SWS_Mirror_00036] [SWS_Mirror_00037] [SWS_Mirror_00138]
[SRS_Mirror_00011]	The Bus Mirroring module shall provide an interface to control the active filters	[SWS_Mirror_00138]
[SRS_Mirror_00012]	The Bus Mirroring module shall provide an interface for module shutdown	[SWS_Mirror_00003]
[SRS_Mirror_00013]	The Bus Mirroring module shall queue output frames	[SWS_Mirror_00011] [SWS_Mirror_00048] [SWS_Mirror_00049] [SWS_Mirror_00050] [SWS_Mirror_00051] [SWS_Mirror_00052] [SWS_Mirror_00053] [SWS_Mirror_00054] [SWS_Mirror_00113] [SWS_Mirror_00119] [SWS_Mirror_00120] [SWS_Mirror_00121] [SWS_Mirror_00122] [SWS_Mirror_00125] [SWS_Mirror_00126] [SWS_Mirror_00137] [SWS_Mirror_00150] [SWS_Mirror_00151] [SWS_Mirror_00152] [SWS_Mirror_00153] [SWS_Mirror_00154] [SWS_Mirror_00155] [SWS_Mirror_00156] [SWS_Mirror_00157] [SWS_Mirror_00158] [SWS_Mirror_00160] [SWS_Mirror_00161]
[SRS_Mirror_00015]	The Bus Mirroring module shall remap LIN PIDs and CAN IDs	[SWS_Mirror_00114] [SWS_Mirror_00115] [SWS_Mirror_00116] [SWS_Mirror_00117] [SWS_Mirror_00118]

Table 6.1: Requirements Tracing

7 Functional Specification

This chapter defines the behavior of the Bus Mirroring module. The API of the module is defined in Chapter 8, while the configuration is defined in Chapter 10.

7.1 Overview

The Bus Mirroring module's task is the collection of frames from several source buses, which are then forwarded to a destination bus. The forwarding is strictly unidirectional to avoid message loops and to prevent intrusion scenarios.

[SWS_Mirror_00001]

Upstream requirements: SRS_Mirror_00001

⌈The generation tool shall ensure that no `ComMChannel` is referenced both from a `MirrorSourceNetwork` and a `MirrorDestNetwork`.⌋

The following figure shows how the Bus Mirroring is integrated in the AUTOSAR BSW communication stack:

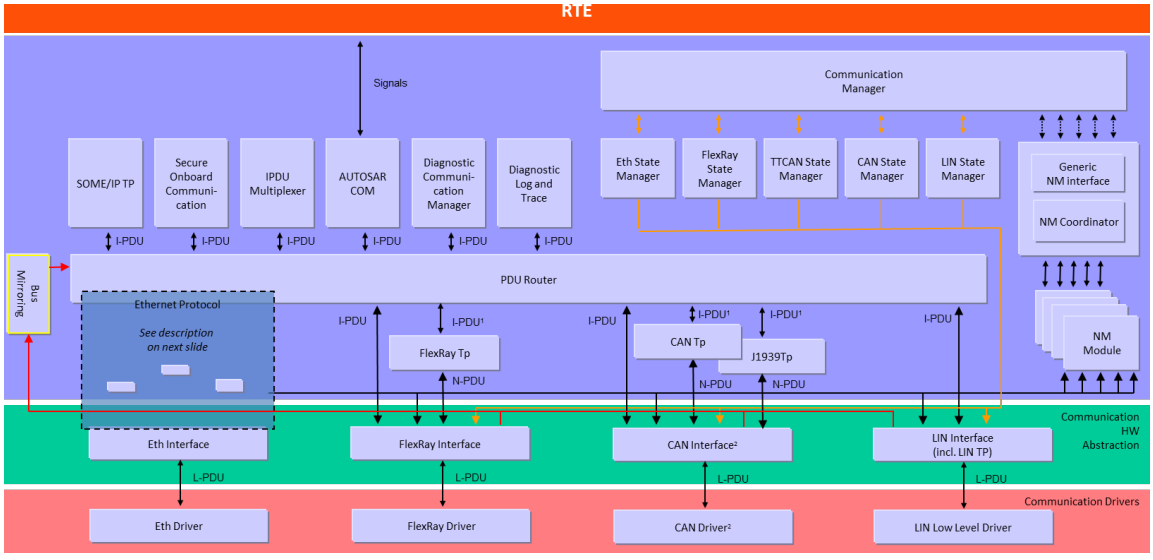

Figure 7.1: AUTOSAR BSW architecture showing the Bus Mirroring module

The following mirroring scenarios are supported by the Bus Mirroring module:

- CAN and LIN ⇒ CAN (direct)
- CAN, CAN FD, and LIN ⇒ CAN FD (direct or serialized)
- CAN, CAN FD, LIN, and FlexRay ⇒ FlexRay (serialized)
- CAN, CAN FD, LIN, and FlexRay ⇒ CAN XL (serialized)

- CAN, CAN FD, LIN, and FlexRay ⇒ IP (serialized)
- CAN, CAN FD, LIN, and FlexRay ⇒ Proprietary (CDD) (serialized)

To avoid overloading the destination bus, the messages received on each source bus are filtered. The filters are configured separately for each bus, either by configuration (see `MirrorSourceCanFilter`, `MirrorSourceLinFilter`, and `MirrorSourceFlexRayFilter`) or at runtime (see Chapter 8).

When frames are mirrored to a CAN 2.0 bus, they are sent directly with identical data. In case of CAN frames, the CAN ID is preserved, but can be remapped to avoid ID conflicts on the destination bus. LIN PIDs, on the other hand, always need to be mapped to appropriate CAN IDs. To avoid ID conflicts, mirrored frames could use ranges of extended CAN IDs.

When frames are mirrored to a CAN FD bus, the configuration parameter `MirrorDestProtocolType` determines whether they are serialized or sent directly.

When frames are mirrored to a FlexRay bus, a CAN XL bus, an IP bus (Ethernet), or a proprietary bus connected as CDD, the source frames are serialized into a larger frame using the protocol specified in Section 7.4.2. When routing to a FlexRay bus, only those FlexRay frames can be routed that are small enough to fit into the destination FlexRay frame reduced by the protocol overhead.

7.2 Module Handling

This section contains description of auxiliary functionality of the Bus Mirroring module.

7.2.1 Initialization

The Bus Mirroring module is initialized via `Mirror_Init`, and de-initialized via `Mirror_DeInit`. Except for `Mirror_GetVersionInfo` and `Mirror_Init`, the API functions of the Bus Mirroring module may only be called after the module has been properly initialized.

[SWS_Mirror_00002]

Upstream requirements: SRS_Mirror_00005, SRS_BSW_00406

⌈A call to `Mirror_Init` initializes all internal variables and sets the Bus Mirroring module to the initialized state.⌋

[SWS_Mirror_00003]

Upstream requirements: SRS_Mirror_00012, SRS_BSW_00336

⌈A call to `Mirror_DeInit` sets the Bus Mirroring module back to the uninitialized state.⌋

[SWS_Mirror_00004]

Upstream requirements: SRS_BSW_00350, SRS_BSW_00386, SRS_BSW_00450

⌈If development error reporting is enabled via `MirrorDevErrorDetect`, the Bus Mirroring module shall call `Det_ReportError` with the error code `MIRROR_E_UNINIT` when any API other than `Mirror_Init` or `Mirror_GetVersionInfo` is called in uninitialized state.⌋

[SWS_Mirror_00005]

Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈When `Mirror_Init` is called in initialized state, the Bus Mirroring module shall not re-initialize its internal variables. It shall instead call `Det_ReportError` with the error code `MIRROR_E_REINIT` if development error reporting is enabled (see `MirrorDevErrorDetect`).⌋

7.2.2 Timing Related Functionality

To be able to measure times, the Bus Mirroring module is triggered cyclically via the `Mirror_MainFunction`.

[SWS_Mirror_00006]

Upstream requirements: SRS_BSW_00478

⌈The Bus Mirroring module shall use the `Mirror_MainFunction` for timing related purposes.⌋

7.2.3 Selection of Active Source Buses

[SWS_Mirror_00013]

Upstream requirements: SRS_Mirror_00005

⌈Upon initialization, the Bus Mirroring module shall be inactive. No source bus is enabled.⌋

To start the Bus Mirroring module, one of the configured source buses (see `MirrorSourceNetwork`) has to be activated. This will start collection of frames and status information from this source bus.

[SWS_Mirror_00014]

Upstream requirements: SRS_Mirror_00010

⌈When a source bus is enabled using `Mirror_StartSourceNetwork`, frame and status acquisition from that bus shall be started, and the state of the source bus shall be reset such that it is reported directly after it has been updated for the first time.⌋

[SWS_Mirror_00015]

Upstream requirements: SRS_Mirror_00010

⌈When a source bus is disabled using `Mirror_StopSourceNetwork`, frame and status acquisition from that bus shall be stopped. Already collected frames shall still be transmitted to the destination bus.⌋

To stop the mirroring, the application may call `Mirror_Offline` at any time.

[SWS_Mirror_00012]

Upstream requirements: SRS_Mirror_00010

⌈When `Mirror_Offline` is called, all sources buses shall be deactivated, the destination bus shall be reset to the `MirrorInitialDestNetworkRef`, all statically configured filters shall be disabled, and all other filters shall be removed. Any mirrored frames still waiting for transmission shall be discarded.⌋

Source buses are also disabled when the destination network is changed (see [SWS_Mirror_00011]).

7.2.4 Switching the Destination Bus

[SWS_Mirror_00009]

Upstream requirements: SRS_Mirror_00005

⌈Upon initialization, the destination bus (`MirrorDestNetwork`) referenced by `MirrorInitialDestNetworkRef` is selected.⌋

Destination frames and status information will not be sent before the mirroring is started (see [SWS_Mirror_00014]).

[SWS_Mirror_00011]

Upstream requirements: SRS_Mirror_00013

⌈When the destination bus is changed using `Mirror_SwitchDestNetwork`, all source buses shall be disabled, all statically configured filters shall be disabled, and

all other filters shall be removed. Mirrored frames that are still waiting for transmission shall be discarded.⌋

This ensures that the selection of information sent to a destination bus has to be chosen specifically for that bus type. Otherwise, switching to a different destination bus could easily overload that bus, especially if it is another internal bus.

The destination bus is reset when the mirroring is stopped (see [SWS_Mirror_00012]).

7.2.5 Controlling Frame Filters

Frame filters can be configured statically (see `MirrorSourceCanFilter`, `MirrorSourceLinFilter`, and `MirrorSourceFlexRayFilter`) or added dynamically at run-time separately for each source bus.

[SWS_Mirror_00016]

Upstream requirements: SRS_Mirror_00005

⌈Upon initialization, all statically configured filters of the Bus Mirroring module are disabled, and no dynamic filters are available.⌋

Statically configured filters can be explicitly activated and deactivated using `Mirror_SetStaticFilterState`. Dynamic filters can be added at run-time, using one of the bus specific `Mirror_Add...Filter` services (e.g. `Mirror_AddCanMaskFilter`), and removed again by calling `Mirror_RemoveFilter` with the filter ID returned by the `Mirror_Add...Filter` service. Filters are also deactivated/removed when mirroring is stopped (see [SWS_Mirror_00012]) or when the destination network is changed (see [SWS_Mirror_00011]).

[SWS_Mirror_00017]

Upstream requirements: SRS_Mirror_00007

⌈While a filter is active (statically configured and activated by `Mirror_SetStaticFilterState` or dynamically added using one of the bus specific `Mirror_Add...Filter` services), all frames from the corresponding source bus that match the filter shall be mirrored.⌋

This means that no frames from a source bus are mirrored as long as no filters are active.

[SWS_Mirror_00018]

Upstream requirements: SRS_Mirror_00007

⌈When a statically configured filter is deactivated by `Mirror_SetStaticFilterState` or a dynamically added filter is removed by `Mirror_RemoveFilter`, frames

that have been accepted before the deactivation/removal shall still be mirrored to the destination bus.⌋

7.3 Access to Source Buses

The Bus Mirroring module supports CAN, CAN FD, LIN, and FlexRay as source buses. To acquire frames and state information of these buses, the Bus Mirroring module interacts with the corresponding bus interface modules. Reported frames are then filtered before they are mirrored to the destination bus.

[SWS_Mirror_00166]

Upstream requirements: SRS_BSW_00459

⌈The Bus Mirroring module shall call interfaces of the CAN, LIN, and FlexRay Interface modules only from within the same partition, to which the `ComMChannel` referenced by `MirrorSourceNetwork` is assigned to.⌋

7.3.1 Access to CAN and CAN FD

Throughout this section, the term CAN bus includes CAN FD buses. A CAN FD bus can transport both classic CAN and CAN FD frames. The Bus Mirroring module accesses the CAN bus through the CAN Interface module (CanIf). After the Bus Mirroring module starts the mirroring of a CAN bus, the CAN Interface module reports received and transmitted CAN and CAN FD frames to the Bus Mirroring module. The CAN bus state is polled cyclically from the `Mirror_MainFunction`.

7.3.1.1 CAN Source Bus Activation

After initialization, the CAN Interface module does not report any frames to the Bus Mirroring module.

[SWS_Mirror_00019]

Upstream requirements: SRS_Mirror_00010

⌈When `Mirror_StartSourceNetwork` is called to start a CAN source bus, the Bus Mirroring module shall call `CanIf_EnableBusMirroring` with `MirroringActive` set to TRUE to start reporting of received and transmitted CAN frames from the corresponding CAN controller.⌋

`Mirror_StartSourceNetwork` receives a `ComMChannelId` as `network`, while `CanIf_EnableBusMirroring` expects a `CanIfCtrlId` as `ControllerId`. The

translation of the one to the other can be determined at generation time by following the references from the ComMChannelId to the CanIfCtrlId through the ECU configuration.

[SWS_Mirror_00020]

Upstream requirements: SRS_Mirror_00010

⌈When Mirror_StopSourceNetwork is called to stop a CAN source bus, the Bus Mirroring module shall call CanIf_EnableBusMirroring with MirroringActive set to FALSE to stop reporting of received and transmitted CAN frames from the corresponding CAN controller.⌋

7.3.1.2 CAN Frame Acquisition

The CAN Interface module reports both received and transmitted CAN (FD) frames with a call to Mirror_ReportCanFrame. Received frames are reported from the reception interrupt or task, while transmitted frames are reported from the transmission confirmation interrupt or task.

[SWS_Mirror_00167]

Upstream requirements: SRS_BSW_00459

⌈The Bus Mirroring module shall apply appropriate mechanisms to allow calls of Mirror_ReportCanFrame from the partition to which the ComMChannel referenced by MirrorComMNetworkHandleRef is assigned to, e.g. by providing a satellite in this partition.⌋

For each reported CAN (FD) frame, the CAN Interface module provides information about the receiving CAN controller, about the CAN ID, the CAN ID type (extended or standard), and the CAN frame type (CAN FD or CAN 2.0), and the length and the actual payload of the frame.

[SWS_Mirror_00021]

Upstream requirements: SRS_Mirror_00006, SRS_Mirror_00007

⌈When Mirror_ReportCanFrame is called to report a received or transmitted CAN frame, the Bus Mirroring module shall match the canId containing the actual CAN ID, the ID type, and the frame type against all active statically configured and dynamically added filters of the corresponding source bus. If the CAN frame matches at least one filter, it is accepted by the Bus Mirroring module.⌋

When mirroring to a (serialized) CAN FD, CAN XL, FlexRay, an IP, or a proprietary destination bus, the source bus is identified by a network ID, but Mirror_ReportCanFrame reports the controllerId. The translation of the one to the other can

be determined at generation time by following the references from the CanIfCtrlId to the MirrorNetworkId through the ECU configuration via MirrorComMNetworkHandleRef.

7.3.1.3 CAN Frame Filters

[SWS_Mirror_00022]

Upstream requirements: SRS_Mirror_00007

⌈A CAN mask filter statically configured as MirrorSourceCanFilterMask matches the reported canId, if this canId masked by the MirrorSourceCanFilterCanIdMask equals the MirrorSourceCanFilterCanIdCode.⌋

[SWS_Mirror_00023]

Upstream requirements: SRS_Mirror_00007

⌈A CAN mask filter dynamically added by a call to Mirror_AddCanMaskFilter matches the reported canId, if this canId masked by the mask equals the id.⌋

[SWS_Mirror_00024]

Upstream requirements: SRS_Mirror_00007

⌈A CAN range filter statically configured as MirrorSourceCanFilterRange matches the reported canId, if the value of this canId is greater than or equal to the MirrorSourceCanFilterLower and smaller than or equal to the MirrorSourceCanFilterUpper.⌋

[SWS_Mirror_00025]

Upstream requirements: SRS_Mirror_00007

⌈A CAN range filter dynamically added by a call to Mirror_AddCanRangeFilter matches the reported canId, if the value of this canId is greater than or equal to the lowerId and smaller than or equal to the upperId.⌋

7.3.1.4 CAN Status Acquisition

[SWS_Mirror_00026]

Upstream requirements: SRS_Mirror_00008, SRS_Mirror_00009

⌈The Bus Mirroring module shall poll the status of each active CAN source bus by cyclically calling CanIf_GetControllerMode and CanIf_GetTrcvMode from the Mirror_MainFunction. If the returned ControllerModePtr is CAN_CS_STARTED

and the returned TransceiverModePtr is CANTRCV_TRCVMODE_NORMAL, the reported CAN source bus state shall be set to online, otherwise to offline. If the bus is online, the Bus Mirroring module shall call CanIf_GetControllerErrorState, and if the returned ErrorStatePtr is CAN_ERRORSTATE_PASSIVE or CAN_ERRORSTATE_BUSOFF, the reported CAN source bus state shall be set to error passive or bus-off, respectively. Additionally, if the bus is online, the Bus Mirroring module shall also call CanIf_GetControllerTxErrorCounter, and add the returned TxErrorCounterPtr to the reported CAN source bus state.⌋

The APIs CanIf_GetControllerMode and CanIf_GetControllerErrorState expect a ControllerId, and CanIf_GetTrcvMode expects a TransceiverId, but a network ID is required to report the status to the output bus. The translation of the ones to the other can be determined at generation time by following the references from the CanIfCtrlId and CanTrcvChannelId, respectively, to the MirrorNetworkId through the ECU configuration via MirrorComMNetworkHandleRef.

7.3.2 Access to LIN

The Bus Mirroring module accesses the LIN bus through the LIN Interface module (LinIf). After the Bus Mirroring module starts the mirroring of a LIN bus, the LIN Interface module reports received and transmitted LIN frames to the Bus Mirroring module. The LIN bus state is partially reported together with the LIN frames, and partially polled cyclically from the Mirror_MainFunction.

7.3.2.1 LIN Source Bus Activation

After initialization, the LIN Interface module does not report any frames to the Bus Mirroring module.

[SWS_Mirror_00027]

Upstream requirements: SRS_Mirror_00010

⌈When Mirror_StartSourceNetwork is called to start a LIN source bus, the Bus Mirroring module shall call LinIf_EnableBusMirroring with MirroringActive set to TRUE to start reporting of received and transmitted LIN frames from that bus.⌋

[SWS_Mirror_00028]

Upstream requirements: SRS_Mirror_00010

⌈When Mirror_StopSourceNetwork is called to stop a LIN source bus, the Bus Mirroring module shall call LinIf_EnableBusMirroring with MirroringActive set to FALSE to stop reporting of received and transmitted LIN frames from that bus.⌋

7.3.2.2 LIN Frame Acquisition

The LIN Interface module reports both received and transmitted LIN frames with a call to `Mirror_ReportLinFrame`. Received and transmitted frames are reported from the LIN schedule processing after the corresponding status check has been executed.

[SWS_Mirror_00168]

Upstream requirements: SRS_BSW_00459

⌈The Bus Mirroring module shall apply appropriate mechanisms to allow calls of `Mirror_ReportLinFrame` from the partition to which the `ComMChannel` referenced by `MirrorComMNetworkHandleRef` is assigned to, e.g. by providing a satellite in this partition.⌋

For each reported LIN frame, the LIN Interface module provides information about the receiving bus, about the protected ID (PID), the length, and the actual payload of the frame, and about the reception or transmission status.

[SWS_Mirror_00029]

Upstream requirements: SRS_Mirror_00006, SRS_Mirror_00007

⌈When `Mirror_ReportLinFrame` is called to report a received or transmitted LIN frame, the Bus Mirroring module shall extract the frame ID from the reported `pid` and match it against all active statically configured and dynamically added filters of the corresponding source bus. If the LIN frame matches at least one filter, it is accepted by the Bus Mirroring module.⌋

The frame ID of a LIN frame is calculated from the PID by removing the two most significant bits.

7.3.2.3 LIN Frame Filters

[SWS_Mirror_00030]

Upstream requirements: SRS_Mirror_00007

⌈A LIN mask filter statically configured as `MirrorSourceLinFilterMask` matches the reported frame ID, if this ID masked by the `MirrorSourceLinFilterLinIdMask` equals the `MirrorSourceLinFilterLinIdCode`.⌋

[SWS_Mirror_00031]

Upstream requirements: SRS_Mirror_00007

⌈A LIN mask filter dynamically added by a call to `Mirror_AddLinMaskFilter` matches the reported frame ID, if this ID masked by the `mask` equals the `id`.⌋

[SWS_Mirror_00032]

Upstream requirements: SRS_Mirror_00007

⌈A LIN range filter statically configured as `MirrorSourceLinFilterRange` matches the reported frame ID, if the value of this ID is greater than or equal to the `MirrorSourceLinFilterLower` and smaller than or equal to the `MirrorSourceLinFilterUpper`.⌋

[SWS_Mirror_00033]

Upstream requirements: SRS_Mirror_00007

⌈A LIN range filter dynamically added by a call to `Mirror_AddLinRangeFilter` matches the reported frame ID, if the value of this ID is greater than or equal to the `lowerId` and smaller than or equal to the `upperId`.⌋

7.3.2.4 LIN Status Acquisition

[SWS_Mirror_00034]

Upstream requirements: SRS_Mirror_00008, SRS_Mirror_00009

⌈The Bus Mirroring module shall evaluate the `status` reported by `Mirror_ReportLinFrame`. If it is `LIN_TX_HEADER_ERROR`, `LIN_TX_ERROR`, `LIN_RX_ERROR`, or `LIN_RX_NO_RESPONSE`, the reported LIN source bus state shall be set to header transmission error, transmission error, reception error, or no response.⌋

[SWS_Mirror_00035]

Upstream requirements: SRS_Mirror_00008, SRS_Mirror_00009

⌈The Bus Mirroring module shall poll the status of each active LIN source bus by cyclically calling `LinIf_GetTrcvMode` from the `Mirror_MainFunction`. If the returned `TransceiverModePtr` is `LINTRCV_TRCV_MODE_NORMAL`, the reported LIN source bus state shall be set to online, otherwise to offline.⌋

7.3.3 Access to FlexRay

The Bus Mirroring module accesses the FlexRay bus through the FlexRay Interface module (FrIf). After the Bus Mirroring module starts the mirroring of a FlexRay bus, the FlexRay Interface module reports received and transmitted FlexRay frames to the Bus Mirroring module. The FlexRay bus state is polled cyclically from the `Mirror_MainFunction`. A FlexRay source bus corresponds to a FlexRay cluster, which can be connected to several controllers.

7.3.3.1 FlexRay Source Bus Activation

After initialization, the FlexRay Interface module does not report any frames to the Bus Mirroring module.

[SWS_Mirror_00036]

Upstream requirements: SRS_Mirror_00010

⌈When `Mirror_StartSourceNetwork` is called to start a FlexRay source bus, the Bus Mirroring module shall call `FrIf_EnableBusMirroring` with `FrIf_MirroringActive` set to TRUE to start reporting of received and transmitted FlexRay frames from the corresponding FlexRay cluster.⌋

`Mirror_StartSourceNetwork` receives a `ComMChannelId` as `network`, while `FrIf_EnableBusMirroring` expects a `FrIfClstIdx` as `FrIf_ClstIdx`. The translation of the one to the other can be determined at generation time by following the references from the `ComMChannelId` to the the related `FrIfClstIdx` through the ECU configuration.

[SWS_Mirror_00037]

Upstream requirements: SRS_Mirror_00010

⌈When `Mirror_StopSourceNetwork` is called to stop a FlexRay source bus, the Bus Mirroring module shall call `FrIf_EnableBusMirroring` with `FrIf_MirroringActive` set to FALSE to stop reporting of received and transmitted FlexRay frames from the corresponding FlexRay cluster.⌋

7.3.3.2 FlexRay Frame Acquisition

The FlexRay Interface module reports both received and transmitted FlexRay frames with a call to `Mirror_ReportFlexRayFrame`. Received and transmitted frames are reported from the job list execution function or the transmit function of the FlexRay Interface.

[SWS_Mirror_00169]

Upstream requirements: SRS_BSW_00459

⌈The Bus Mirroring module shall apply appropriate mechanisms to allow calls of `Mirror_ReportFlexRayFrame` from the partition to which the `ComMChannel` referenced by `MirrorComMNetworkHandleRef` is assigned to, e.g. by providing a satellite in this partition.⌋

For each reported FlexRay frame, the FlexRay Interface module provides information about the receiving FlexRay controller and about the slot ID and cycle, the length and the actual payload of the frame, and information about transmission conflicts.

[SWS_Mirror_00038]

Upstream requirements: SRS_Mirror_00006, SRS_Mirror_00007

⌈When `Mirror_ReportFlexRayFrame` is called to report a received or transmitted FlexRay frame (`txConflict` is reported as FALSE), the Bus Mirroring module shall match the `slotId` and `cycle` against all active statically configured and dynamically added filters of the corresponding source bus. If the FlexRay frame matches at least one filter, it is accepted by the Bus Mirroring module.⌋

On the destination bus, the source bus is identified by a network ID, but `Mirror_ReportFlexRayFrame` reports the `controllerId`. The translation of the one to the other can be determined at generation time by following the references from the `FrIfCtrlIdx` to the `MirrorNetworkId` through the ECU configuration via `MirrorComMNetworkHandleRef`.

7.3.3.3 FlexRay Frame Filters

[SWS_Mirror_00039]

Upstream requirements: SRS_Mirror_00007

⌈A FlexRay filter statically configured as `MirrorSourceFlexRayFilter` matches the reported `slotId` and `cycle` if the `slotId` is greater than or equal to the `MirrorSourceFlexRayFilterLowerSlot` and smaller than or equal to the `MirrorSourceFlexRayFilterUpperSlot` and if the `cycle` modulo `MirrorSourceFlexRayFilterCycleRepetition` is greater than or equal to the `MirrorSourceFlexRayFilterLowerBaseCycle` and smaller than or equal to the `MirrorSourceFlexRayFilterUpperBaseCycle`.⌋

[SWS_Mirror_00040]

Upstream requirements: SRS_Mirror_00007

⌈A FlexRay filter dynamically added by a call to `Mirror_AddFlexRayFilter` matches the reported `slotId` and `cycle` if the `slotId` is greater than or equal to the `lowerSlotId` and smaller than or equal to the `upperSlotId` and if the `cycle` modulo `cycleRepetition` is greater than or equal to the `lowerBaseCycle` and smaller than or equal to the `upperBaseCycle`.⌋

7.3.3.4 FlexRay Status Acquisition

[SWS_Mirror_00041]

Upstream requirements: SRS_Mirror_00008, SRS_Mirror_00009

⌈When `Mirror_ReportFlexRayFrame` is called to report a transmission conflict (`txConflict` is reported as TRUE), the Bus Mirroring module shall match the `slotId` and `cycle` against all active statically configured and dynamically added filters. If it matches at least one filter, the reported FlexRay source bus state for that frame shall be set to transmission conflict.⌋

The callback `Mirror_ReportFlexRayFrame` reports a `controllerId` and the API `FrIf_GetPOCStatus` expects a `FrIf_CtrlIdx`, but a network ID is required to report the status to the output bus. The translation of the one to the other can be determined at generation time by following the references from the `FrIfCtrlIdx` to the `MirrorNetworkId` through the ECU configuration via `MirrorComMNetworkHandleRef`.

[SWS_Mirror_00146]

Upstream requirements: SRS_Mirror_00008, SRS_Mirror_00009

⌈When `Mirror_ReportFlexRayChannelStatus` is called to report the FlexRay channel state, the Bus Mirroring module shall compare the reported states with the previously reported states. It the states differ in Bit 1 (vSS!SyntaxError), Bit 2 (vSS!ContentError), and/or Bit 4 (vSS!Bviolation), the Bus Mirroring module shall update the reported FlexRay source bus state accordingly.⌋

The callback `Mirror_ReportFlexRayChannelStatus` reports a `clusterId` and the API `FrIf_GetState` expects a `FrIf_ClstIdx`, but a network ID is required to report the status to the output bus. The translation of the one to the other can be determined at generation time by following the references from the `FrIfClstIdx` to the `MirrorNetworkId` through the ECU configuration via `MirrorComMNetworkHandleRef`.

[SWS_Mirror_00042]

Upstream requirements: SRS_Mirror_00008, SRS_Mirror_00009

⌈The Bus Mirroring module shall poll the status of each active FlexRay source bus by cyclically calling `FrIf_GetState` from the `Mirror_MainFunction`. If the returned `FrIf_StatePtr` is `FRIF_STATE_ONLINE`, the reported FlexRay source bus state shall be set to online, otherwise to offline. If the bus is online, the Bus Mirroring module shall also call `FrIf_GetPOCStatus` for each controller connected to the FlexRay cluster. If the returned `Fr_POCStateType` is `FR_POCSTATE_NORMAL_ACTIVE` for

all controllers, the reported source bus state shall be synchronous and normal active; if Fr_POCStateType is FR_POCSTATE_NORMAL_PASSIVE for at least one controller, the reported source bus state shall be synchronous but not normal active; if Fr_POCStateType is in any other state for at least one controller, the reported source bus state shall be neither synchronous nor normal active.⌋

7.4 Serialized Mirroring

When mirroring to a serialized CAN FD destination bus, a FlexRay destination bus, a CAN XL destination bus, an IP destination bus like Ethernet, or a proprietary network connected as CDD, the Bus Mirroring module applies a protocol to pack several smaller frames into one large frame of the destination bus.

The first sub section of this section (Section 7.4.1) defines how the Bus Mirroring module places the source frames onto a destination frame using the mirroring protocol, and how the queueing is applied before transmitting a destination frames.

The second section (Section 7.4.2) shows the exact layout of the protocol and the meaning and usage of the fields in the protocol.

7.4.1 Handling of Destination Frames

This section describes how to handle the mirroring protocol, which is defined in Section 7.4.2.

7.4.1.1 Creation

[SWS_Mirror_00043]

Upstream requirements: SRS_Mirror_00008

⌈When the Bus Mirroring module is initialized or when Mirror_SwitchDestNetwork is called to activate a serialized CAN FD (MirrorDestNetworkCanFD where MirrorDestProtocolType is not set to MIRROR_PT_NONE), FlexRay (MirrorDestNetworkFlexRay), CAN XL (MirrorDestNetworkCanXL), IP (MirrorDestNetworkIp), or proprietary (MirrorDestNetworkCdd) destination bus, the Bus Mirroring module shall activate a new destination frame buffer and reset the SequenceNumber to 0.⌋

[SWS_Mirror_00044]

Upstream requirements: SRS_Mirror_00008

⌈When the first data item is added to an empty destination frame buffer (as described in [SWS_Mirror_00045], [SWS_Mirror_00046], or [SWS_Mirror_00047]) the Bus Mirroring module shall first write the header to the buffer in the layout defined by [SWS_Mirror_00055].

The `ProtocolVersion` field shall be set to 1, the `SequenceNumber` to the incremented `SequenceNumber` of the last destination frame, the `HeaderTimestamp` shall be filled with the information returned by `StbM_GetCurrentTime`, and the `DataLength` field shall be set to 0.

If the optional configuration parameter `MirrorDestTransmissionDeadline` is configured, the Bus Mirroring module shall start the transmission timeout timer.⌋

[SWS_Mirror_00045]

Upstream requirements: SRS_Mirror_00008

⌈When a source frame has been received as described in [SWS_Mirror_00021], [SWS_Mirror_00029], or [SWS_Mirror_00038], the Bus Mirroring module shall create a new data item and place it as at the end of the currently active destination frame buffer in the layout defined by [SWS_Mirror_00064], and it shall add the size of the new data item to the header field `DataLength`.

The `Timestamp` field of the new data item shall be set to the difference between the time stamp contained in the header and the current time acquired using `StbM_GetCurrentTime` expressed in multiples of $10\,\mu s$, the `FrameIDAvailable` and `PayloadAvailable` bits shall be set to 1, and the fields `NetworkType`, `NetworkID`, `FrameID`, `PayloadLength`, and `Payload` shall be set according to the received source frame.

If the reported source bus state changed since the last transmission of a source frame, the `NetworkStateAvailable` bit shall be set to 1 and the `NetworkState` field to the reported source bus state. Otherwise, the `NetworkStateAvailable` bit shall be set to 0 and the `NetworkState` field shall be omitted.⌋

[SWS_Mirror_00046]

Upstream requirements: SRS_Mirror_00008

⌈When a new FlexRay transmission conflict was reported as described in [SWS_Mirror_00041], the Bus Mirroring module shall create a new data item and place it at the end of the currently active destination frame buffer in the layout defined by [SWS_Mirror_00064], and it shall add the size of the new data item to the header field `DataLength`.

The `Timestamp` field of the data item shall be set to the difference between the time stamp contained in the header and the current time acquired using `StbM_GetCurrentTime` expressed in multiples of $10\,\mu s$, the `FrameIDAvailable`

and NetworkStateAvailable bits shall be set to 1, and the fields NetworkType, NetworkID, and FrameID shall be set according to the reported transmission conflict. The NetworkState field shall be set to the reported source bus state.

The PayloadAvailable bit shall be set to 0, and the fields PayloadLength and Payload shall be omitted.⌋

Each reported FlexRay transmission conflict invalidates a preceding FlexRay frame. The invalidated FlexRay frame could be located in another destination frame than the corresponding transmission conflict.

[SWS_Mirror_00047]

Upstream requirements: SRS_Mirror_00008

⌈When the reported source bus state has changed and if no source frame is received from the same source bus within one main function cycle, the Bus Mirroring module shall create a new data item and place it at the end of the currently active destination frame buffer in the layout defined by [SWS_Mirror_00064], and it shall add the size of the new data item to the header field DataLength.

The Timestamp field of the data item shall be set to the difference between the time stamp contained in the header and the current time acquired using StbM_GetCurrentTime expressed in multiples of $10\,\mu s$. The NetworkStateAvailable bit shall be set to 1, the fields NetworkType and NetworkID shall be set according to the reported source bus, and the NetworkState field shall be set to the reported source bus state.

Depending on the currently reported source bus state, the FrameIDAvailable shall be set to 1 or 0. In the first case, the FrameID shall be set according to the reported source bus, and in the latter case the FrameID shall be omitted.

The PayloadAvailable bit shall be set to 0, and the fields PayloadLength and Payload shall be omitted.⌋

Section 7.4.2.2.7 lists the error codes that can be reported in the NetworkState field and describes the necessity to provide the FrameID.

7.4.1.2 Queueing

[SWS_Mirror_00048]

Upstream requirements: SRS_Mirror_00008, SRS_Mirror_00013

⌈When a data item does not fit in the remaining space of the currently active destination frame buffer, the Bus Mirroring module shall place this buffer in the queue and activate a new destination frame buffer. The data item shall then be placed in the new buffer.⌋

[SWS_Mirror_00049]

Upstream requirements: SRS_Mirror_00008, SRS_Mirror_00013

⌈When the relative time stamp of a data item exceeds $655.35\,ms$, the Bus Mirroring module shall place the currently active destination frame buffer in the queue and activate a new destination frame buffer. The data item shall then be placed in the new buffer.⌋

[SWS_Mirror_00050]

Upstream requirements: SRS_Mirror_00008, SRS_Mirror_00013

⌈If the optional configuration parameter `MirrorDestTransmissionDeadline` is configured and the transmission timeout expires, the Bus Mirroring module shall place the currently active destination frame buffer in the queue and active a new destination frame buffer.⌋

The size of the queue for the serialized destination frames is determined by the configuration parameter `MirrorDestQueueSize`, the size of the queue elements by the `PduLength` of the `Pdu` referenced by `MirrorDestPduRef`.

[SWS_Mirror_00113]

Upstream requirements: SRS_Mirror_00013, SRS_BSW_00386

⌈If a destination frame cannot be placed in the queue because the queue is already full, the Bus Mirroring module shall drop that destination frame, report the runtime error `MIRROR_E_QUEUE_OVERRUN`, and shall set (to 1) the Frames Lost bit of the `NetworkState` of the next data item created in the currently active destination frame buffer.⌋

7.4.1.3 Transmission

[SWS_Mirror_00051]

Upstream requirements: SRS_Mirror_00013

⌈To initiate the transmission of a queued serialized destination frame, the Bus Mirroring module shall call `PduR_MirrorTransmit` with `PduInfoPtr->MetaDataPtr` set to the `NULL_PTR` and `PduInfoPtr->SduLength` set to the actually written part of the destination frame. If `MirrorDestPduUsesTriggerTransmit` is enabled, `PduInfoPtr->SduDataPtr` shall be set to the `NULL_PTR`, otherwise to the used part of the queued destination frame.⌋

A `NULL_PTR` for `PduInfoPtr->SduDataPtr` ensures that the destination bus interface module (`FrIf`, `CanIf`, `SoAd`, or a `CDD`) fetches the destination frame using `Mirror_TriggerTransmit`.

[SWS_Mirror_00150]

Upstream requirements: SRS_Mirror_00013, SRS_BSW_00386

⌈If the PduR_MirrorTransmit returns E_NOT_OK, the Bus Mirroring module shall immediately remove the destination frame from the queue, shall report the runtime error MIRROR_E_TRANSMIT_FAILED, and shall set (to 1) the Frames Lost bit of the NetworkState of the next data item created in the currently active destination frame buffer.⌋

[SWS_Mirror_00053]

Upstream requirements: SRS_Mirror_00013

⌈The Bus Mirroring module shall initiate the transmission of queued serialized destination frames from the Mirror_MainFunction and from the Mirror_TxConfirmation callback.⌋

This ensures that queued destination frames are transmitted as fast as possible.

To enable a suitable throughput on a FlexRay destination bus, the MirrorDestNetworkFlexRay may contain a set of MirrorDestPdus.

[SWS_Mirror_00160]

Upstream requirements: SRS_Mirror_00013

⌈If a set of MirrorDestPdus is configured for a MirrorDestNetworkFlexRay, the Bus Mirroring module shall use the PDUs of this set in arbitrary order.⌋

The SequenceNumber together with the Timestamp of the data items will ensure that a tester can sort them correctly.

[SWS_Mirror_00052]

Upstream requirements: SRS_Mirror_00013

⌈In case the active destination channel is MirrorDestNetworkCanFD, MirrorDestNetworkCanXL, MirrorDestNetworkIp or MirrorDestNetworkCdd, the Bus Mirroring module shall not transmit the next serialized destination frame before the previous destination frame has been confirmed by a call to Mirror_TxConfirmation.⌋

[SWS_Mirror_00161]

Upstream requirements: SRS_Mirror_00013

⌈In case the active destination channel is MirrorDestNetworkFlexRay, the Bus Mirroring module shall not transmit the next serialized destination frame using the same MirrorDestPdu before the previous transmission of that MirrorDestPdu has been confirmed by a call to Mirror_TxConfirmation.⌋

[SWS_Mirror_00054]

Upstream requirements: SRS_Mirror_00013

⌈When `Mirror_TriggerTransmit` is called for a serialized destination frame, the Mirror module shall copy the used part of the queued destination frame to `PduInfoPtr->SduDataPtr` and update `PduInfoPtr->SduLength` accordingly.⌋

[SWS_Mirror_00151]

Upstream requirements: SRS_Mirror_00013, SRS_BSW_00386

⌈If the `PduInfoPtr->SduLength` provided by `Mirror_TriggerTransmit` is too small for the currently transmitted serialized destination frame, the Bus Mirroring module shall remove the destination frame from the queue, shall report the runtime error `MIRROR_E_TRANSMIT_FAILED`, shall set (to 1) the Frames Lost bit of the `NetworkState` of the next data item created in the currently active serialized destination frame buffer, and shall return `E_NOT_OK` to stop this transmission.⌋

[SWS_Mirror_00152]

Upstream requirements: SRS_Mirror_00013

⌈When `Mirror_TxConfirmation` is called to report the successful or failed transmission of a serialized destination frame, the Bus Mirroring module shall remove the destination frame from the queue.⌋

[SWS_Mirror_00153]

Upstream requirements: SRS_Mirror_00013, SRS_BSW_00386

⌈If the `Mirror_TxConfirmation` reports the failed transmission of a serialized destination frame (`result` is `E_NOT_OK`), the Bus Mirroring module shall report the runtime error `MIRROR_E_TRANSMIT_FAILED`, and shall set (to 1) the Frames Lost bit of the `NetworkState` of the next data item created in the currently active destination frame buffer.⌋

7.4.2 Mirroring Protocol

The protocol that is applied by the Bus Mirroring module for IP, FlexRay, and proprietary destination buses is shown in Figure 7.2, in this example for an Ethernet destination bus.

Figure 7.2: Bus Mirroring Serialization Protocol

The protocol consists of a header (see Section 7.4.2.1) followed by several data items (see Section 7.4.2.2).

In the tables and descriptions of this section, the byte numbers increase in the same sequence as the bytes are transmitted on the destination bus, starting from 0. The bit numbers decrease, the most significant bit of a byte being bit 7 and the least significant bit 0.

7.4.2.1 Header Layout

Every destination frame starts with a header, which is shown in Figure 7.3.

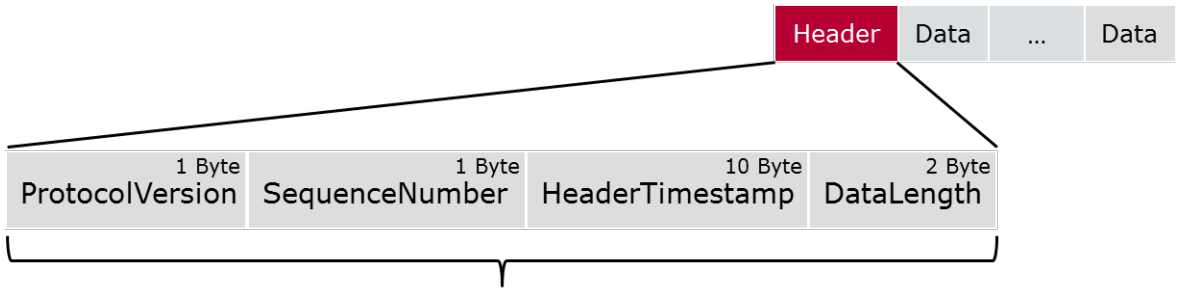

Figure 7.3: Bus Mirroring Protocol Header

[SWS_Mirror_00055]

Upstream requirements: SRS_Mirror_00008

⌈The header of a Bus Mirroring destination frame shall contain the following fields in this order:

1. `ProtocolVersion`
2. `SequenceNumber`
3. `HeaderTimestamp`
4. `DataLength`

⌋

The fields of the header are described in detail in the following subsections.

7.4.2.1.1 ProtocolVersion

[SWS_Mirror_00056]

Upstream requirements: SRS_Mirror_00008

⌈The `ProtocolVersion` shall indicate the layout of the header and the data items. The layout currently defined in this section is identified by `ProtocolVersion` 1. The

range $[2 \,..\, 127]$ is reserved for future extensions of the AUTOSAR defined protocol, the range $[128 \,..\, 255]$ is available for customer specific protocols.⌋

The protocol version allows the tester tool to interpret the protocol correctly, and to enable different layouts of the protocol.

[SWS_Mirror_00057]

Upstream requirements: SRS_Mirror_00008

⌈The width of the `ProtocolVersion` field shall be 8 bits.⌋

7.4.2.1.2 SequenceNumber

[SWS_Mirror_00058]

Upstream requirements: SRS_Mirror_00008

⌈The `SequenceNumber` shall increase with each transmission of a destination frame. After initialization or after switching the destination bus with `Mirror_SwitchDestNetwork`, it shall start from 0.⌋

The sequence number allows the tester tool to identify lost destination frames.

[SWS_Mirror_00059]

Upstream requirements: SRS_Mirror_00008

⌈The width of the `SequenceNumber` field shall be 8 bits.⌋

This means that the `SequenceNumber` will wrap around to 0 after it reached 255. A tester has to cope with this behavior and still sort the frames correctly.

7.4.2.1.3 HeaderTimestamp

[SWS_Mirror_00060]

Upstream requirements: SRS_Mirror_00008

⌈The `HeaderTimestamp` shall reflect the time when collection of data items into the destination frame started. This time shall be given as the absolute number of seconds and nanoseconds since January 1st of 1970.⌋

[SWS_Mirror_00061]

Upstream requirements: SRS_Mirror_00008

⌈The total width of the `HeaderTimestamp` field shall be 10 bytes, where the seconds take the upper 48 Bits and the nanoseconds take the lower 32 Bits. Both elements of the the `HeaderTimestamp` field shall be encoded in network byte order (MSB first).⌋

Table 7.1 shows the layout of the `HeaderTimestamp`.

HeaderTimestamp									
Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**	**Byte 9**
Seconds (48 bits, MSB first)						Nanoseconds (32 bits, MSB first)			

Table 7.1: Layout of `HeaderTimestamp`

7.4.2.1.4 DataLength

[SWS_Mirror_00062]

Upstream requirements: SRS_Mirror_00008

⌈The `DataLength` shall give the number of bytes following the header. It is the sum of the length of all data items in the destination frame.⌋

[SWS_Mirror_00063]

Upstream requirements: SRS_Mirror_00008

⌈The width of the `DataLength` field shall be 16 bits. It shall be encoded in network byte order (MSB first).⌋

7.4.2.2 Data Item Layout

Every source frame is placed in a data item, which is shown in Figure 7.4.

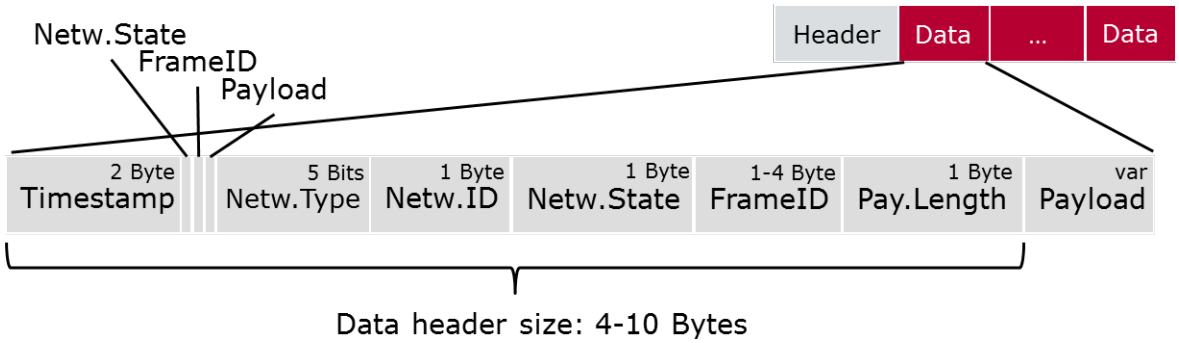

Figure 7.4: Bus Mirroring Protocol Data Item

[SWS_Mirror_00064]

Upstream requirements: SRS_Mirror_00008

⌈Data items of a Bus Mirroring destination frame shall contain the following fields in this order:

1. `Timestamp`
2. `NetworkStateAvailable`
3. `FrameIDAvailable`
4. `PayloadAvailable`
5. `NetworkType`
6. `NetworkID`
7. `NetworkState` (optional)
8. `FrameID` (optional)
9. `PayloadLength` (optional)
10. `Payload` (optional)

⌋

The fields of the data item are described in detail in the following sub sections.

7.4.2.2.1 Timestamp

[SWS_Mirror_00065]

Upstream requirements: SRS_Mirror_00008

⌈The `Timestamp` shall reflect the temporal offset of the source frame reception from the `HeaderTimestamp`, i.e. the time that passed since collection of data items into the destination frame started. It shall be given in multiples of $10\,\mu s$.⌋

[SWS_Mirror_00066]

Upstream requirements: SRS_Mirror_00008

⌈The width of the `Timestamp` field shall be 16 bits. It shall be encoded in network byte order (MSB first).⌋

7.4.2.2.2 NetworkStateAvailable

[SWS_Mirror_00067]

Upstream requirements: SRS_Mirror_00008

⌈The `NetworkStateAvailable` shall indicate whether the field `NetworkState` is present in the data item. If `NetworkStateAvailable` is 1, that field shall be present. If it is 0, that field shall be omitted.⌋

[SWS_Mirror_00068]

Upstream requirements: SRS_Mirror_00008

⌈The width of the `NetworkStateAvailable` field shall be 1 bit.⌋

7.4.2.2.3 FrameIDAvailable

[SWS_Mirror_00069]

Upstream requirements: SRS_Mirror_00008

⌈The `FrameIDAvailable` shall indicate whether the field `FrameID` is present in the data item. If `FrameIDAvailable` is 1, that field shall be present. If it is 0, that field shall be omitted.⌋

[SWS_Mirror_00070]

Upstream requirements: SRS_Mirror_00008

⌈The width of the `FrameIDAvailable` field shall be 1 bit.⌋

7.4.2.2.4 PayloadAvailable

[SWS_Mirror_00071]

Upstream requirements: SRS_Mirror_00008

⌈The `PayloadAvailable` shall indicate whether the fields `PayloadLength` and `Payload` are present in the data item. If `PayloadAvailable` is 1, these fields shall be present. If it is 0, these fields shall be omitted.⌋

[SWS_Mirror_00072]

Upstream requirements: SRS_Mirror_00008

⌈The width of the `PayloadAvailable` field shall be 1 bit.⌋

7.4.2.2.5 NetworkType

[SWS_Mirror_00073]

Upstream requirements: SRS_Mirror_00008

⌈The `NetworkType` shall indicate the type of the source bus.⌋

[SWS_Mirror_00074]

Upstream requirements: SRS_Mirror_00008

⌈The width of the `NetworkType` field shall be 5 bits, the possible values are defined in [SWS_Mirror_00170]. The range $[5 \,..\, 15]$ is reserved for future extensions of the AUTOSAR defined protocol, the range $[16 \,..\, 31]$ is available for customer specific bus types.⌋

[SWS_Mirror_00170] Values of `NetworkType`

Upstream requirements: SRS_Mirror_00008

⌈

Invalid	0
Network Type	**Numerical**
CAN or CAN FD	1
LIN	2
FlexRay	3
Ethernet	4

⌋

7.4.2.2.6 NetworkID

[SWS_Mirror_00075]

Upstream requirements: SRS_Mirror_00008

⌈The NetworkID shall identify a bus of a certain NetworkType uniquely, i.e. the same NetworkID can appear on different NetworkTypes, but not on the same NetworkType.⌋

[SWS_Mirror_00076]

Upstream requirements: SRS_Mirror_00008

⌈The width of the NetworkID field shall be 8 bits.⌋

7.4.2.2.7 NetworkState

[SWS_Mirror_00077]

Upstream requirements: SRS_Mirror_00008

⌈The NetworkState shall provide information about the source bus state. It shall only be present when the source bus state has changed since the last time it was reported, the presence shall be indicated by NetworkStateAvailable.⌋

[SWS_Mirror_00078]

Upstream requirements: SRS_Mirror_00008

⌈The width of the NetworkState field shall be 8 bits, the layout is bus specific and is defined separately for each bus as NetworkStateCAN, NetworkStateLIN, and NetworkStateFlexRay.⌋

[SWS_Mirror_00079]

Upstream requirements: SRS_Mirror_00008

⌈Bit 7 (the most significant bit) of the NetworkState shall always contain the Frames Lost state. This is a sporadic error that is not related to the source frame that is reported in the same data item, but shall not be reported in a separate data item. The Frames Lost state shall be set once to 1 after one or more source frames that passed the filters were lost because the queue of the destination bus was full or the transmission failed. Afterwards it shall be set to 0 again.⌋

[SWS_Mirror_00080]

Upstream requirements: SRS_Mirror_00008

⌈Bit 6 of the `NetworkState` shall always contain the Bus Online state. This is a continuous state that is not related to the source frame that is reported in the same data item, and may also be reported in a data item where the `FrameIDAvailable` and `PayloadAvailable` fields are set to 0. The Bus Online state shall be set to 1 when the source bus is online, i.e. when both the controller and the transceiver are able to communicate. Otherwise it shall be set to 0.⌋

7.4.2.2.7.1 NetworkStateCAN

The layout of the `NetworkState` for a CAN or CAN FD bus is shown in Table 7.3.

NetworkState							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Frames Lost	Bus Online	Error-Passive	Bus-Off	Tx error counter, divided by 8			

Table 7.3: Layout of CAN `NetworkState`

[SWS_Mirror_00081]

Upstream requirements: SRS_Mirror_00008

⌈Bit 5 of the `NetworkStateCAN` shall contain the Error-Passive state. This is a continuous state that is not related to the source frame that is reported in the same data item, and may also be reported in a data item where the `FrameIDAvailable` and `PayloadAvailable` fields are set to 0.

The Error-Passive state shall be set to 1 when the CAN controller is in the Error-Passive state, and to 0 when it is in the Error-Active or Bus-Off state.⌋

[SWS_Mirror_00082]

Upstream requirements: SRS_Mirror_00008

⌈Bit 4 of the `NetworkStateCAN` shall contain the Bus-Off state. This is a continuous state that is not related to the source frame that is reported in the same data item, and may also be reported in a data item where the `FrameIDAvailable` and `PayloadAvailable` fields are set to 0.

The Bus-Off state shall be set to 1 when the CAN controller is in the Bus-Off state, and to 0 when it is in the Error-Active or Error-Passive state.⌋

[SWS_Mirror_00083]

Upstream requirements: SRS_Mirror_00008

⌈Bits 3 – 0 of the `NetworkStateCAN` shall contain the Tx error counter of the can controller divided by 8. This is a continuous state that is not related to the source frame that is reported in the same data item, and may also be reported in a data item where the `FrameIDAvailable` and `PayloadAvailable` fields are set to 0.⌋

7.4.2.2.7.2 NetworkStateLIN

The layout of the `NetworkState` for a LIN bus is shown in Table 7.4.

NetworkState							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Frames Lost	Bus Online	*reserved*		Header Tx Error	Tx Error	Rx Error	Rx No Response

Table 7.4: Layout of LIN `NetworkState`

[SWS_Mirror_00084]

Upstream requirements: SRS_Mirror_00008

⌈Bits 5 and 4 of the `NetworkStateLIN` are currently reserved. They shall always be set to 0.⌋

[SWS_Mirror_00085]

Upstream requirements: SRS_Mirror_00008

⌈Bit 3 of the `NetworkStateLIN` shall contain the Header Tx Error state. This is an error that is related to the source frame that is reported in the same data item.

The Header Tx Error state shall be set to 1 when the LIN controller detected an error during transmission of a LIN header. Otherwise it shall be set to 0.⌋

[SWS_Mirror_00086]

Upstream requirements: SRS_Mirror_00008

⌈Bit 2 of the `NetworkStateLIN` shall contain the Tx Error state. This is an error that is related to the source frame that is reported in the same data item.

The Tx Error state shall be set to 1 when the LIN controller detected an error during transmission of a LIN frame. Otherwise it shall be set to 0.⌋

[SWS_Mirror_00087]

Upstream requirements: SRS_Mirror_00008

⌈Bit 1 of the `NetworkStateLIN` shall contain the Rx Error state. This is an error that is related to the source frame that is reported in the same data item.

The Rx Error state shall be set to 1 when the LIN controller detected an error during reception of a LIN frame. Otherwise it shall be set to 0.⌋

[SWS_Mirror_00088]

Upstream requirements: SRS_Mirror_00008

⌈Bit 0 of the `NetworkStateLIN` shall contain the Header Rx No Response state. This is an error that is related to the source frame that is reported in the same data item.

The Rx No Response state shall be set to 1 when the LIN controller did not receive the expected LIN frame after transmission of a LIN header. Otherwise it shall be set to 0.⌋

7.4.2.2.7.3 NetworkStateFlexRay

The layout of the `NetworkState` for a FlexRay bus is shown in Table 7.5.

NetworkState							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Frames Lost	Bus Online	Bus Synchronous	Normal Active	Syntax Error	Content Error	Boundary Violation	Tx Conflict

Table 7.5: Layout of FlexRay `NetworkState`

[SWS_Mirror_00089]

Upstream requirements: SRS_Mirror_00008

⌈Bit 5 of the `NetworkStateFlexRay` shall contain the Bus Synchronous state. This is a continuous state that is not related to the source frame that is reported in the same data item, and may also be reported in a data item where the `FrameIDAvailable` and `PayloadAvailable` fields are set to 0.

The Bus Synchronous state shall be set to 1 when all FlexRay controllers connected to that bus are synchronous to the network time. Otherwise it shall be set to 0.⌋

[SWS_Mirror_00090]

Upstream requirements: SRS_Mirror_00008

⌈Bit 4 of the `NetworkStateFlexRay` shall contain the Normal Active state. This is a continuous state that is not related to the source frame that is reported in the same

data item, and may also be reported in a data item where the FrameIDAvailable and PayloadAvailable fields are set to 0.

The Normal Active state shall be set to 1 when all FlexRay controllers connected to that bus are synchronous and in the normal active state. Otherwise it shall be set to 0.⌋

[SWS_Mirror_00091]

Upstream requirements: SRS_Mirror_00008

⌈Bit 3 of the NetworkStateFlexRay shall contain the Syntax Error state. This is an aggregated error flag of the FlexRay channels that is related to the channel assignment of the FrameID, but not to a source frame and its FrameID that is reported in the same data item. It may also be reported in a data item where the PayloadAvailable field is set to 0 and the FrameIDAvailable is set to 1 with the slot valid flag of the FrameID set to 0.

The Syntax Error state shall be set to 1 once after a FlexRay controller detected a syntax error. Otherwise it shall be set to 0.⌋

[SWS_Mirror_00092]

Upstream requirements: SRS_Mirror_00008

⌈Bit 2 of the NetworkStateFlexRay shall contain the Content Error state. This is an aggregated error flag of the FlexRay channels that is related to the channel assignment of the FrameID, but not to a source frame and its FrameID that is reported in the same data item. It may also be reported in a data item where the PayloadAvailable field is set to 0 and the FrameIDAvailable is set to 1 with the slot valid flag of the FrameID set to 0.

The Content Error state shall be set to 1 once after a FlexRay controller detected a content error. Otherwise it shall be set to 0.⌋

[SWS_Mirror_00093]

Upstream requirements: SRS_Mirror_00008

⌈Bit 1 of the NetworkStateFlexRay shall contain the Boundary Violation state. This is an aggregated error flag of the FlexRay channels that is related to the channel assignment of the FrameID, but not to a source frame and its FrameID that is reported in the same data item. It may also be reported in a data item where the PayloadAvailable field is set to 0 and the FrameIDAvailable is set to 1 with the slot valid flag of the FrameID set to 0.

The Boundary Violation state shall be set to 1 once after a FlexRay controller detected a boundary violation. Otherwise it shall be set to 0.⌋

[SWS_Mirror_00094]

Upstream requirements: SRS_Mirror_00008

⌈Bit 0 of the NetworkStateFlexRay shall contain the Tx Conflict state. This is an error that is related to the previous source frame that was reported with the same FrameID and is always reported in a data item where the FrameIDAvailable field is set to 1 and the PayloadAvailable field is set to 0.

The Tx Conflict state shall be set to 1 when a FlexRay controller detected a transmission conflict. Otherwise it shall be set to 0.⌋

7.4.2.2.8 FrameID

[SWS_Mirror_00095]

Upstream requirements: SRS_Mirror_00008

⌈The FrameID shall provide the identification of the source frame. This identification shall be unique for one source bus identified by NetworkType and NetworkID. The FrameID may be omitted when reporting a source bus state change, the presence shall be indicated by FrameIDAvailable.⌋

[SWS_Mirror_00096]

Upstream requirements: SRS_Mirror_00008

⌈The width and layout of the FrameID field is bus specific and is defined separately for each bus as FrameIDCAN, FrameIDLIN, and FrameIDFlexRay.⌋

7.4.2.2.8.1 FrameIDCAN

The layout of the FrameID for a CAN or CAN FD bus is shown in Table 7.6.

FrameID						
Byte 0				Byte 1	Byte 2	Byte 3
Bit 7	Bit 6	Bit 5	Bits 4 .. 0			
Ext.ID/ Std.ID	FD/ 2.0	*res.*	CAN ID (Bits 28 .. 24)	CAN ID (Bits 23 .. 16)	CAN ID (Bits 15 .. 8)	CAN ID (Bits 7 .. 0)

Table 7.6: Layout of CAN FrameID

The layout of the FrameIDCAN corresponds to the Can_IdType provided by Mirror_ReportCanFrame.

[SWS_Mirror_00097]

Upstream requirements: SRS_Mirror_00008

⌈The width of the `FrameIDCAN` field shall be 4 bytes.⌋

[SWS_Mirror_00098]

Upstream requirements: SRS_Mirror_00008

⌈Bit 7 of Byte 0 of the `FrameIDCAN` shall be set to 1 for an Extended CAN ID and to 0 for a Standard CAN ID.⌋

[SWS_Mirror_00099]

Upstream requirements: SRS_Mirror_00008

⌈Bit 6 of Byte 0 of the `FrameIDCAN` shall be set to 1 for a CAN FD frame and to 0 for a CAN 2.0 frame.⌋

[SWS_Mirror_00100]

Upstream requirements: SRS_Mirror_00008

⌈Bit 5 of Byte 0 of the `FrameIDCAN` is currently reserved. It shall always be set 0.⌋

[SWS_Mirror_00101]

Upstream requirements: SRS_Mirror_00008

⌈Bits 4 – 0 of Byte 0 and Bytes 1 – 3 of the `FrameIDCAN` shall contain the CAN ID of the reported CAN frame in network byte order (MSB first).⌋

7.4.2.2.8.2 FrameIDLIN

The layout of the `FrameID` for a LIN bus is shown in Table 7.7.

FrameID
Byte 0
LIN PID

Table 7.7: Layout of LIN `FrameID`

[SWS_Mirror_00102]

Upstream requirements: SRS_Mirror_00008

⌈The width of the `FrameIDLIN` field shall be 1 byte.⌋

[SWS_Mirror_00103]

Upstream requirements: SRS_Mirror_00008

⌈Byte 0 of the `FrameIDLIN` shall contain the LIN PID of the reported LIN frame.⌋

7.4.2.2.8.3 FrameIDFlexRay

The layout of the `FrameID` for a FlexRay bus is shown in Table 7.8.

FrameID						
Byte 0					Byte 1	Byte 2
Bit 7	Bit 6	Bit 5 .. 4	Bit 3	Bits 2 .. 0		
Channel B	Channel A	*reserved*	Slot Valid	Slot ID (Bits 10 .. 8)	Slot ID (Bits 7 .. 0)	Cycle

Table 7.8: Layout of FlexRay `FrameID`

[SWS_Mirror_00104]

Upstream requirements: SRS_Mirror_00008

⌈The width of the `FrameIDFlexRay` field shall be 3 bytes.⌋

[SWS_Mirror_00105]

Upstream requirements: SRS_Mirror_00008

⌈Bits 7 – 6 of Byte 0 of the `FrameIDFlexRay` shall contain the channel assignment of the reported FlexRay frame. Bit 7 shall be set to 1 if the reported FlexRay frame is available on channel B of the FlexRay controller, otherwise it shall be set to 0. Bit 6 shall be set to 1 if the reported FlexRay frame is available on channel A of the FlexRay controller, otherwise it shall be set to 0. A reported FlexRay frame is either assigned exclusively to channel A or B or to both channels.⌋

This layout of the channel assignment corresponds to the `Fr_ChannelType` reported by `Mirror_ReportFlexRayFrame`.

[SWS_Mirror_00106]

Upstream requirements: SRS_Mirror_00008

⌈Bits 5 – 4 of Byte 0 of the `FrameIDFlexRay` are currently reserved. They shall always be set 0.⌋

[SWS_Mirror_00159]

Upstream requirements: SRS_Mirror_00008

⌈Bit 3 of Byte 0 of the `FrameIDFlexRay` shall contain a flag indicating whether the reported slot ID and cycle are valid (flag is 1) or unused (flag is 0). It shall only be set to 0 when an aggregated error of the FlexRay channels is reported independently of a source frame or transmission conflict. Otherwise it shall always be set to 1.⌋

[SWS_Mirror_00107]

Upstream requirements: SRS_Mirror_00008

⌈Bits 2 – 0 of Byte 0 and Byte 1 of the `FrameIDFlexRay` shall contain the slot ID of the reported FlexRay frame in network byte order (MSB first).⌋

[SWS_Mirror_00108]

Upstream requirements: SRS_Mirror_00008

⌈Byte 2 of the `FrameIDFlexRay` shall contain the cycle in which the reported FlexRay frame was sent or received.⌋

Please note: For received frames and for frames sent in the static segment, the cycle is always reliable. For frames sent in the dynamic segment, the actual cycle cannot be known in advance, because the frame might not be transmitted in the planned cycle.

7.4.2.2.9 PayloadLength

[SWS_Mirror_00109]

Upstream requirements: SRS_Mirror_00008

⌈The `PayloadLength` shall provide the length of the payload of the source frame. It may be omitted when reporting a source bus state change, the presence shall be indicated by `PayloadAvailable`.⌋

[SWS_Mirror_00110]

Upstream requirements: SRS_Mirror_00008

⌈The width of the `PayloadLength` field shall be 8 bits.⌋

7.4.2.2.10 Payload

[SWS_Mirror_00111]

Upstream requirements: SRS_Mirror_00008

⌈The `Payload` shall provide the actual payload of the source frame. It may be omitted when reporting a source bus state change, the presence shall be indicated by `PayloadAvailable`.⌋

[SWS_Mirror_00112]

Upstream requirements: SRS_Mirror_00008

⌈The width of the `Payload` field shall correspond to the reported source frame. The maximum values are 8 bytes for LIN and CAN 2.0, 64 bytes for CAN FD, and 254 for FlexRay.⌋

7.5 Direct Mirroring

When mirroring to a CAN destination bus or a direct CAN FD bus (`MirrorDestNetworkCanFD.MirrorDestProtocolType` == `MIRROR_PT_NONE`), the Bus Mirroring module sends received CAN (FD) and LIN frames directly to the destination bus, though possibly with a changed CAN ID to avoid conflicts with regular messages on the destination bus.

This section defines how the Bus Mirroring module translates CAN IDs and queues the source frames and how it creates and queues status frames before transmitting them on the destination bus.

Again, throughout this section, the term CAN bus includes CAN FD buses.

7.5.1 Handling of Source Frames

This section describes how to process and transmit the source frames that were received from the CAN and LIN bus as described in Section 7.3.1.2 and Section 7.3.2.2, respectively.

7.5.1.1 ID Mapping

Usually, CAN source frames can be transmitted unchanged on the destination bus, while the PIDs of LIN source frames have to be mapped to a range of CAN ID.

But sometimes, it is hard to find a consecutive sequence of unused CAN IDs for mapping of the LIN PIDs, or the same CAN ID is also used by frames that are usually transmitted on the destination CAN bus.

In these cases, certain CAN IDs and LIN PIDs have to be remapped to special CAN IDs.

7.5.1.1.1 ID Mapping on CAN

[SWS_Mirror_00114]

Upstream requirements: SRS_Mirror_00015

⌈If the `canId` of a CAN source frame matches the `MirrorSourceCanSingleIdMappingSourceCanId` of a `MirrorSourceCanSingleIdMapping`, the destination frame shall be transmitted with the `MirrorSourceCanSingleIdMappingDestCanId` of that mapping.⌋

[SWS_Mirror_00115]

Upstream requirements: SRS_Mirror_00015

⌈If the `canId` of a CAN source frame masked by the `MirrorSourceCanMaskBasedIdMappingSourceCanIdMask` of a `MirrorSourceCanMaskBasedIdMapping` matches the `MirrorSourceCanMaskBasedIdMappingSourceCanIdCode` of that mapping, the CAN destination frame shall be transmitted with the masked `canId` added to the `MirrorSourceCanMaskBasedIdMappingDestBaseId`.⌋

[SWS_Mirror_00116]

Upstream requirements: SRS_Mirror_00015

⌈If the `canId` of a CAN source frame matches neither a `MirrorSourceCanSingleIdMapping` nor a `MirrorSourceCanMaskBasedIdMapping`, the CAN destination frame shall be transmitted with the original `canId`, i.e. identical CAN ID, ID type (Extended or Standard), and frame type (CAN FD or CAN 2.0).⌋

7.5.1.1.2 ID Mapping on LIN

[SWS_Mirror_00117]

Upstream requirements: SRS_Mirror_00015

⌈If the frame ID extracted from the `pid` of a LIN source frame matches the `MirrorSourceLinToCanIdMappingLinId` of a `MirrorSourceLinToCanIdMapping`, the CAN destination frame shall be transmitted with the `MirrorSourceLinToCanIdMappingCanId` of that mapping.⌋

[SWS_Mirror_00118]

Upstream requirements: SRS_Mirror_00015

⌈If the frame ID extracted from the `pid` of a LIN source frame matches no `MirrorSourceLinToCanIdMapping`, the CAN destination frame shall be transmitted with the LIN frame ID added to the `MirrorSourceLinToCanBaseId`.⌋

7.5.1.2 Queuing

[SWS_Mirror_00119]

Upstream requirements: SRS_Mirror_00013

⌈The Bus Mirroring module shall place all CAN destination frames in the queue.⌋

The size of the queue for the CAN destination frames is determined by the configuration parameter `MirrorDestQueueSize`, the size of the queue elements by the `PduLength` of the `Pdu` referenced by `MirrorDestPduRef`.

[SWS_Mirror_00120]

Upstream requirements: SRS_Mirror_00013, SRS_BSW_00386

⌈If a destination frame cannot be placed in the queue because the queue is already full, the Bus Mirroring module shall drop that destination frame, report the runtime error `MIRROR_E_QUEUE_OVERRUN`, and set (to 1) the Frames Lost bit of the `NetworkState` in the next status frame.⌋

The handling of status frames is defined in Section 7.5.2.

7.5.1.3 Transmission

To be able to transmit arbitrary CAN IDs with arbitrary type (Extended / Standard) in CAN frames of arbitrary type (CAN 2.0 / CAN FD), the Bus Mirroring module uses a `MirrorDestPdu` with MetaData and open CanIdMask (see [SWS_Mirror_CONSTR_00001]).

[SWS_Mirror_00121]

Upstream requirements: SRS_Mirror_00013

⌈To initiate the transmission of a queued CAN destination frame, the Bus Mirroring module shall call `PduR_MirrorTransmit` with `PduInfoPtr->MetaDataPtr` set to

MetaData containing the CAN ID of the destination frame and PduInfoPtr->SduLength set to the length of the destination frame. If MirrorDestPduUsesTriggerTransmit is enabled, PduInfoPtr->SduDataPtr shall be set to the NULL_PTR, otherwise to the payload of the source frame.⌋

A NULL_PTR for PduInfoPtr->SduDataPtr ensures that the destination bus interface module (CanIf) fetches the destination frame using Mirror_TriggerTransmit.

[SWS_Mirror_00154]

Upstream requirements: SRS_Mirror_00013, SRS_BSW_00386

⌈If the PduR_MirrorTransmit returns E_NOT_OK, the Bus Mirroring module shall immediately remove the destination frame from the queue, shall report the runtime error MIRROR_E_TRANSMIT_FAILED, and shall set (to 1) the Frames Lost bit of the NetworkState of the next status frame.⌋

[SWS_Mirror_00155]

Upstream requirements: SRS_Mirror_00013

⌈The Bus Mirroring module shall initiate the transmission of queued CAN destination frames from the Mirror_MainFunction and from the Mirror_TxConfirmation callback.⌋

This ensures that queued destination frames are transmitted as fast as possible.

[SWS_Mirror_00156]

Upstream requirements: SRS_Mirror_00013

⌈The Bus Mirroring module shall not transmit the next CAN destination frame before the previous destination frame has been confirmed by a call to Mirror_TxConfirmation.⌋

[SWS_Mirror_00122]

Upstream requirements: SRS_Mirror_00013

⌈When Mirror_TriggerTransmit is called for a CAN destination frame, the Mirror module shall copy the payload of the source frame to PduInfoPtr->SduDataPtr and update PduInfoPtr->SduLength accordingly.⌋

On the CAN bus, it is not possible that Mirror_TriggerTransmit provides a PduInfoPtr->SduLength that is too small for the destination frame, because the destination frame has by configuration a size of 8 bytes for CAN 2.0 or 64 bytes for CAN FD, and the CanIf will always provide the hardware buffer size, which is also 8 bytes for CAN 2.0 and 64 bytes for CAN FD.

[SWS_Mirror_00157]

Upstream requirements: SRS_Mirror_00013

⌈When `Mirror_TxConfirmation` is called to report the successful or failed transmission of a CAN destination frame, the Bus Mirroring module shall remove the destination frame from the queue.⌋

[SWS_Mirror_00158]

Upstream requirements: SRS_Mirror_00013, SRS_BSW_00386

⌈If the `Mirror_TxConfirmation` reports the failed transmission of a CAN destination frame (`result` is `E_NOT_OK`), the Bus Mirroring module shall report the runtime error `MIRROR_E_TRANSMIT_FAILED`, and shall set (to 1) the Frames Lost bit of the `NetworkState` of the next status frame.⌋

7.5.2 Creation of Status Frames

[SWS_Mirror_00123]

Upstream requirements: SRS_Mirror_00009

⌈If `MirrorStatusCanId` is configured and when one or more source bus states have changed, the Bus Mirroring module shall allocate a new status frame buffer and write the header in the layout defined by [SWS_Mirror_00127].

The `SHProtocolVersion` field shall be set to 1.⌋

[SWS_Mirror_00124]

Upstream requirements: SRS_Mirror_00009

⌈If `MirrorStatusCanId` is configured, the Bus Mirroring module shall create a new status item for each source bus where the reported state has changed and place it at the end of the currently active status frame buffer in the layout defined by [SWS_Mirror_00129].

The fields `SINetworkType` and `SINetworkID` shall be set according to the reported source bus, the `SINetworkState` field shall be set to the reported source bus state.

Depending on the currently reported source bus state, the `SIFrameIDAvailable` shall be set to 1 or 0. In the first case, the `SIFrameID` shall be set according to the reported source bus, and in the latter case the `SIFrameID` shall be omitted.⌋

Section 7.4.2.2.7 lists the error codes that can be reported in the `SINetworkState` field and describes the necessity to provide the `SIFrameID`.

[SWS_Mirror_00125]

Upstream requirements: SRS_Mirror_00009, SRS_Mirror_00013

⌈When a status item does not fit in the remaining space of the currently active status frame buffer, the Bus Mirroring module shall place this buffer in the queue with the CAN ID configured in `MirrorStatusCanId` and activate a new status frame buffer.⌋

[SWS_Mirror_00126]

Upstream requirements: SRS_Mirror_00009, SRS_Mirror_00013

⌈When status items have been written for all source buses where the reported state has changed, the Bus Mirroring module shall place the currently active status frame buffer in the queue with the CAN ID configured in `MirrorStatusCanId`.⌋

7.5.3 Status Protocol

The protocol that is applied by the Bus Mirroring module for transmission of status frames on CAN consists of a header (see Section 7.5.3.1) followed by several data items (see Section 7.5.3.2).

In the tables and descriptions of this section, the byte numbers increase in the same sequence as the bytes are transmitted on the destination bus, starting from 0. The bit numbers decrease, the most significant bit of a byte being bit 7 and the least significant bit 0.

7.5.3.1 Status Header Layout

Every status frame starts with a header, which is shown in Figure 7.5.

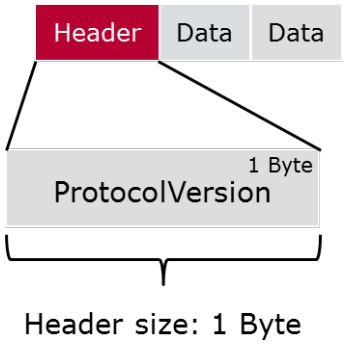

Figure 7.5: Status Frame Header

[SWS_Mirror_00127]

Upstream requirements: SRS_Mirror_00009

⌈The header of a Bus Mirroring status frame shall contain the `SHProtocolVersion`.⌋

7.5.3.1.1 SHProtocolVersion

[SWS_Mirror_00128]

Upstream requirements: SRS_Mirror_00009

⌈The `SHProtocolVersion` shall be identical to the `ProtocolVersion` of a serialized destination frame.⌋

The `ProtocolVersion` is defined in Section 7.4.2.1.1.

7.5.3.2 Status Item Layout

Every source bus state is placed in a status item, which is shown in Figure 7.6.

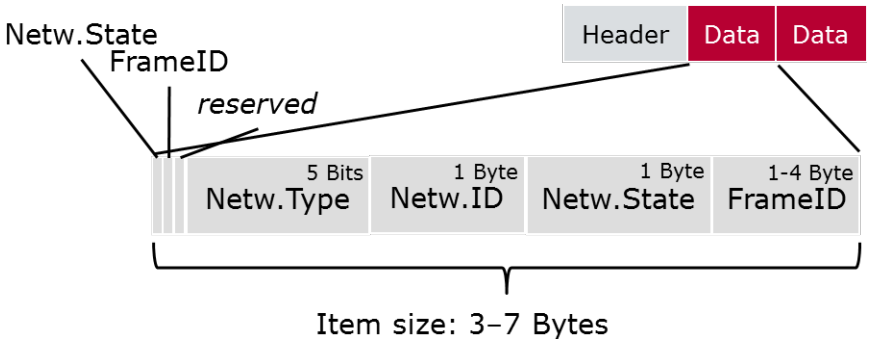

Figure 7.6: Status Frame Item

[SWS_Mirror_00129]

Upstream requirements: SRS_Mirror_00009

⌈Status items of a Bus Mirroring status frame shall contain the following fields in this order:

1. `SINetworkStateAvailable`
2. `SIFrameIDAvailable`
3. *reserved*
4. `SINetworkType`
5. `SINetworkID`
6. `SINetworkState`
7. `SIFrameID` (optional)

⌋

[SWS_Mirror_00132]

Upstream requirements: SRS_Mirror_00009

⌈Bit 5 of Byte 0 of the status item is currently reserved and shall always be set to 0.⌋

The fields of the status item are described in detail in the following sub sections.

7.5.3.2.1 SINetworkStateAvailable

[SWS_Mirror_00149]

Upstream requirements: SRS_Mirror_00009

⌈The layout and semantics of the `SINetworkStateAvailable` shall be identical to the `NetworkStateAvailable` used in a serialized data item. It shall always be set to 1.⌋

The `NetworkStateAvailable` is defined in Section 7.4.2.2.2. The receiver of a Bus Mirroring status frame can use the `SINetworkStateAvailable` to check for a valid status item: If this bit is 0, the remainder of the frame can be ignored, it is probably just padding (see also [SWS_Mirror_CONSTR_00002]).

7.5.3.2.2 SIFrameIDAvailable

[SWS_Mirror_00131]

Upstream requirements: SRS_Mirror_00009

⌈The layout and semantics of the `SIFrameIDAvailable` shall be identical to the `FrameIDAvailable` used in a serialized data item.⌋

The `FrameIDAvailable` is defined in Section 7.4.2.2.3.

7.5.3.2.3 SINetworkType

[SWS_Mirror_00133]

Upstream requirements: SRS_Mirror_00009

⌈The layout and semantics of the `SINetworkType` shall be identical to the `NetworkType` used in a serialized data item.⌋

The `SINetworkType` is defined in Section 7.4.2.2.5.

7.5.3.2.4 SINetworkID

[SWS_Mirror_00134]

Upstream requirements: SRS_Mirror_00009

⌈The layout and semantics of the `SINetworkID` shall be identical to the `NetworkID` used in a serialized data item.⌋

The `NetworkID` is defined in Section 7.4.2.2.6.

7.5.3.2.5 SINetworkState

[SWS_Mirror_00135]

Upstream requirements: SRS_Mirror_00009

⌈The layout and semantics of the `SINetworkState` shall be identical to the `NetworkState` used in a serialized data item.⌋

The `NetworkState` is defined in Section 7.4.2.2.7.

7.5.3.2.6 SIFrameID

[SWS_Mirror_00136]

Upstream requirements: SRS_Mirror_00009

⌈The layout and semantics of the `SIFrameID` shall be identical to the `FrameID` used in a serialized data item.⌋

The `FrameID` is defined in Section 7.4.2.2.8.

7.6 Error Classification

Section 7.2 “Error Handling” of the document “General Specification of Basic Software Modules” [2] describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, this section specifies particular errors arranged in the respective sub sections below.

7.6.1 Development Errors

[SWS_Mirror_00007] Definiton of development errors in module Mirror

Upstream requirements: SRS_BSW_00385, SRS_BSW_00480, SRS_BSW_00481

⌈

Type of error	*Related error code*	*Error value*
An API was called while the module was uninitialized	MIRROR_E_UNINIT	0x01
The init API was called twice	MIRROR_E_REINIT	0x02
Mirror_Init was called with an invalid configuration pointer	MIRROR_E_INIT_FAILED	0x03
An API service was called with a NULL pointer	MIRROR_E_PARAM_POINTER	0x10
An API service was called with a wrong ID	MIRROR_E_INVALID_PDU_SDU_ID	0x11
An API service was called with wrong network handle	MIRROR_E_INVALID_NETWORK_ID	0x12

⌋

7.6.2 Runtime Errors

[SWS_Mirror_00008] Definiton of runtime errors in module Mirror

Upstream requirements: SRS_BSW_00385, SRS_BSW_00452

⌈

Type of error	*Related error code*	*Error value*
A message could not be stored in the queue	MIRROR_E_QUEUE_OVERRUN	0x40
A message could not be transmitted	MIRROR_E_TRANSMIT_FAILED	0x41

⌋

7.6.3 Production Errors

The Bus Mirroring module does not define production errors.

7.6.4 Extended Production Errors

The Bus Mirroring module does not define extended production errors.

8 API Specification

8.1 API Parameter Checking

The Bus Mirroring module reports the development error MIRROR_E_PARAM_POINTER when a NULL_PTR is not accepted as an argument to a service or callback function. The exact behavior is specified in [SWS_BSW_00050] and [SWS_BSW_00212].

[SWS_Mirror_00137]

Upstream requirements: SRS_Mirror_00013, SRS_BSW_00386

⌈If development error detection is enabled by MirrorDevErrorDetect, the Bus Mirroring module shall check the TxPduId of the callback functions Mirror_TxConfirmation and Mirror_TriggerTransmit against MirrorDestPduId, and shall report the development error MIRROR_E_INVALID_PDU_SDU_ID when an unknown ID is provided by the call.⌋

[SWS_Mirror_00138]

Upstream requirements: SRS_Mirror_00010, SRS_Mirror_00011, SRS_BSW_00386

⌈If development error detection is enabled by MirrorDevErrorDetect, the Bus Mirroring module shall check the NetworkHandleType parameters of its service functions against the ComMChannelId referenced via MirrorComMNetworkHandleRef, and shall report the development error MIRROR_E_INVALID_NETWORK_ID when an unknown network handle is provided by the call.⌋

8.2 Imported Types

In this section, all types used by the Bus Mirroring module are listed together with the defining module:

[SWS_Mirror_01100] Definition of imported datatypes of module Mirror ⌈

Module	*Header File*	*Imported Type*
Can	Can_GeneralTypes.h	Can_ControllerStateType
	Can_GeneralTypes.h	Can_ErrorStateType
	Can_GeneralTypes.h	Can_IdType
CanTrcv	Can_GeneralTypes.h	CanTrcv_TrcvModeType
Comtype	ComStack_Types.h	NetworkHandleType
	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType

▽

△

Module	Header File	Imported Type
	ComStack_Types.h	PduLengthType
Fr	Fr_GeneralTypes.h	Fr_ChannelType
	Fr_GeneralTypes.h	Fr_ErrorModeType
	Fr_GeneralTypes.h	Fr_POCStateType
	Fr_GeneralTypes.h	Fr_POCStatusType
	Fr_GeneralTypes.h	Fr_SlotModeType
	Fr_GeneralTypes.h	Fr_StartupStateType
	Fr_GeneralTypes.h	Fr_WakeupStatusType
FrIf	FrIf.h	FrIf_StateType
Lin	Lin_GeneralTypes.h	Lin_FramePidType
	Lin_GeneralTypes.h	Lin_StatusType
LinTrcv	Lin_GeneralTypes.h	LinTrcv_TrcvModeType
StbM	Rte_StbM_Type.h	StbM_SynchronizedTimeBaseType
	Rte_StbM_Type.h	StbM_TimeBaseStatusType
	Rte_StbM_Type.h	StbM_TimeStampType
	Rte_StbM_Type.h	StbM_TimeTupleType
	Rte_StbM_Type.h	StbM_UserDataType
	StbM.h	StbM_VirtualLocalTimeType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

⌋

8.3 Type Definitions

8.3.1 Mirror_ConfigType

[SWS_Mirror_01002] Definition of datatype Mirror_ConfigType ⌈

Name	Mirror_ConfigType	
Kind	Structure	
Elements	Implementation specific.	
	Type	–
	Comment	–
Description	This is the base type for the configuration of the Bus Mirroring module. A pointer to an instance of this structure will be used in the initialization of the Bus Mirroring module. The content of this structure is defined in chapter 10 Configuration specification.	
Available via	Mirror.h	

⌋

8.3.2 MIRROR_INVALID_NETWORK

[SWS_Mirror_00165] Definition of NetworkHandleType-extension for module Mirror ⌈

Range	MIRROR_INVALID_NETWORK	0xFF	Invalid network ID.
Description	This type represents a special value of NetworkHandleType, representing an invalid network handle.		
Available via	Mirror.h		

⌋

8.4 Function Definitions

This is a list of functions provided for upper layer modules.

8.4.1 Generic Functions

8.4.1.1 Mirror_Init

[SWS_Mirror_01003] Definition of API function Mirror_Init

Upstream requirements: SRS_BSW_00485

⌈

Service Name	Mirror_Init	
Syntax	`void Mirror_Init ( const Mirror_ConfigType* configPtr )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	configPtr	Pointer to selected configuration structure
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function initializes the Bus Mirroring module. In configurations, in which Mirror is assigned to more than one partition (i.e. Mirror_Main Functions are mapped to partitions), Mirror may provide one init function per partition.	
Available via	Mirror.h	

⌋

8.4.1.2 Mirror_DeInit

[SWS_Mirror_01004] Definition of API function Mirror_DeInit ⌈

Service Name	Mirror_DeInit

Syntax	`void Mirror_DeInit (`
	`  void`
	`)`

Service ID [hex]	0x02
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	This function resets the Bus Mirroring module to the uninitialized state.
Available via	Mirror.h

⌋

8.4.1.3 Mirror_GetVersionInfo

[SWS_Mirror_01005] Definition of API function Mirror_GetVersionInfo

Upstream requirements: SRS_BSW_00482

⌈

Service Name	Mirror_GetVersionInfo	

Syntax	`void Mirror_GetVersionInfo (`
	`  Std_VersionInfoType* versionInfo`
	`)`	

Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	versionInfo	Pointer to where to store the version information of this module.
Return value	None	
Description	Returns the version information of this module.	
Available via	Mirror.h	

⌋

8.4.2 Filter Handling

8.4.2.1 Mirror_GetStaticFilterState

[SWS_Mirror_01006] Definition of API function Mirror_GetStaticFilterState

Upstream requirements: SRS_BSW_00484

⌈

Service Name	Mirror_GetStaticFilterState	
Syntax	`Std_ReturnType Mirror_GetStaticFilterState ( NetworkHandleType network, uint8 filterId, boolean* isActive )`	
Service ID [hex]	0x23	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	network	ComM channel that corresponds to the source bus to which the filter is attached.
	filterId	ID of the filter.
Parameters (inout)	None	
Parameters (out)	isActive	Pointer to where to store the current filter state.
Return value	Std_ReturnType	`E_OK`: Filter state copied to isActive. `E_NOT_OK`: Function was called with invalid parameters.
Description	Returns the state of a pre-configured filter.	
Available via	Mirror.h	

⌋

8.4.2.2 Mirror_SetStaticFilterState

[SWS_Mirror_01007] Definition of API function Mirror_SetStaticFilterState

Upstream requirements: SRS_BSW_00484

⌈

Service Name	Mirror_SetStaticFilterState
Syntax	`Std_ReturnType Mirror_SetStaticFilterState ( NetworkHandleType network, uint8 filterId, boolean isActive )`
Service ID [hex]	0x14
Sync/Async	Synchronous
Reentrancy	Reentrant for different networks. Non reentrant for the same network.

▽

△

Parameters (in)	network	ComM channel that corresponds to the source bus to which the filter is attached.
	filterId	ID of the filter.
	isActive	TRUE: Activate filter FALSE: Deactivate filter
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Filter state updated from isActive. `E_NOT_OK`: Function was called with invalid parameters.
Description	Sets the state of a pre-configured filter.	
Available via	Mirror.h	

⌋

8.4.2.3 Mirror_AddCanRangeFilter

[SWS_Mirror_01008] Definition of API function Mirror_AddCanRangeFilter

Upstream requirements: SRS_BSW_00484

⌈

Service Name	Mirror_AddCanRangeFilter	
Syntax	`Std_ReturnType Mirror_AddCanRangeFilter ( NetworkHandleType network, uint8* filterId, Can_IdType lowerId, Can_IdType upperId )`	
Service ID [hex]	0x15	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different networks. Non reentrant for the same network.	
Parameters (in)	network	ComM channel that corresponds to the CAN bus to which the filter shall be attached.
	lowerId	Lower CAN ID of the range.
	upperId	Upper CAN ID of the range.
Parameters (inout)	None	
Parameters (out)	filterId	ID of the newly created filter.
Return value	Std_ReturnType	`E_OK`: New filter created. `E_NOT_OK`: Creation of filter failed because of invalid parameters or because no filter on the given network was free.
Description	Creates a CAN ID range filter.	
Available via	Mirror.h	

⌋

8.4.2.4 Mirror_AddCanMaskFilter

[SWS_Mirror_01009] Definition of API function Mirror_AddCanMaskFilter ⌈

Service Name	Mirror_AddCanMaskFilter	
Syntax	`Std_ReturnType Mirror_AddCanMaskFilter ( NetworkHandleType network, uint8* filterId, Can_IdType id, Can_IdType mask )`	
Service ID [hex]	0x16	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different networks. Non reentrant for the same network.	
Parameters (in)	network	ComM channel that corresponds to the CAN bus to which the filter shall be attached.
	id	CAN ID used to match a received or transmitted CAN ID.
	mask	Mask that defines the bits of 'id' that are relevant for comparison with the actual CAN ID.
Parameters (inout)	None	
Parameters (out)	filterId	ID of the newly created filter.
Return value	Std_ReturnType	`E_OK`: New filter created. `E_NOT_OK`: Creation of filter failed because of invalid parameters or because no filter on the given network was free.
Description	Creates a CAN ID mask filter.	
Available via	Mirror.h	

⌋

8.4.2.5 Mirror_AddLinRangeFilter

[SWS_Mirror_01010] Definition of API function Mirror_AddLinRangeFilter ⌈

Service Name	Mirror_AddLinRangeFilter	
Syntax	`Std_ReturnType Mirror_AddLinRangeFilter ( NetworkHandleType network, uint8* filterId, uint8 lowerId, uint8 upperId )`	
Service ID [hex]	0x17	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different networks. Non reentrant for the same network.	
Parameters (in)	network	ComM channel that corresponds to the LIN bus to which the filter shall be attached.
	lowerId	Lower frame ID of the range.
	upperId	Upper frame ID of the range.
Parameters (inout)	None	

▽

△

Parameters (out)	filterId	ID of the newly created filter.
Return value	Std_ReturnType	`E_OK`: New filter created. `E_NOT_OK`: Creation of filter failed because of invalid parameters or because no filter on the given network was free.
Description	Creates a LIN frame ID range filter.	
Available via	Mirror.h	

⌋

8.4.2.6 Mirror_AddLinMaskFilter

[SWS_Mirror_01011] Definition of API function Mirror_AddLinMaskFilter ⌈

Service Name	Mirror_AddLinMaskFilter	
Syntax	`Std_ReturnType Mirror_AddLinMaskFilter ( NetworkHandleType network, uint8* filterId, uint8 id, uint8 mask )`	
Service ID [hex]	0x18	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different networks. Non reentrant for the same network.	
Parameters (in)	network	ComM channel that corresponds to the LIN bus to which the filter shall be attached.
	id	Frame ID used to match a received or transmitted frame ID.
	mask	Mask that defines the bits of 'id' that are relevant for comparison with the actual frame ID.
Parameters (inout)	None	
Parameters (out)	filterId	ID of the newly created filter.
Return value	Std_ReturnType	`E_OK`: New filter created. `E_NOT_OK`: Creation of filter failed because of invalid parameters or because no filter on the given network was free.
Description	Creates a LIN frame ID mask filter.	
Available via	Mirror.h	

⌋

8.4.2.7 Mirror_AddFlexRayFilter

[SWS_Mirror_01012] Definition of API function Mirror_AddFlexRayFilter ⌈

Service Name	Mirror_AddFlexRayFilter	
Syntax	`Std_ReturnType Mirror_AddFlexRayFilter ( NetworkHandleType network, uint8* filterId, uint16 lowerSlotId, uint16 upperSlotId, uint8 lowerBaseCycle, uint8 upperBaseCycle, uint8 cycleRepetition, Fr_ChannelType frChannel )`	
Service ID [hex]	0x19	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different networks. Non reentrant for the same network.	
Parameters (in)	network	ComM channel that corresponds to the FlexRay bus to which the filter shall be attached.
	lowerSlotId	Lower slot ID of a range of slot IDs.
	upperSlotId	Upper slot ID of a range of slot IDs.
	lowerBaseCycle	Lower base cycle of a range of cycles.
	upperBaseCycle	Upper base cycle of a range of cycles.
	cycleRepetition	Repetition pattern of selected cycles (2^n).
	frChannel	FlexRay channel assignment.
Parameters (inout)	None	
Parameters (out)	filterId	ID of the newly created filter.
Return value	Std_ReturnType	`E_OK`: New filter created. `E_NOT_OK`: Creation of filter failed because of invalid parameters or because no filter on the given network was free.
Description	Creates a FlexRay filter.	
Available via	Mirror.h	

⌋

8.4.2.8 Mirror_RemoveFilter

[SWS_Mirror_01013] Definition of API function Mirror_RemoveFilter ⌈

Service Name	Mirror_RemoveFilter
Syntax	`Std_ReturnType Mirror_RemoveFilter ( NetworkHandleType network, uint8 filterId )`
Service ID [hex]	0x1a
Sync/Async	Synchronous

▽

△

Reentrancy	Reentrant for different networks. Non reentrant for the same network.	
Parameters (in)	network	ComM channel that corresponds to the source bus to which the filter is attached.
	filterId	ID of the filter.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Filter was removed. `E_NOT_OK`: Function was called with invalid parameters.
Description	Removes a CAN, LIN, or FlexRay filter that was added at runtime.	
Available via	Mirror.h	

⌋

8.4.3 State Handling

8.4.3.1 Mirror_IsMirrorActive

[SWS_Mirror_01014] Definition of API function Mirror_IsMirrorActive ⌈

Service Name	Mirror_IsMirrorActive	

Syntax	`boolean Mirror_IsMirrorActive (`
	`  void`
	`)`	

Service ID [hex]	0x20	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	boolean	TRUE: Bus Mirroring module is active FALSE: Bus Mirroring module is inactive
Description	Returns the global mirroring state.	
Available via	Mirror.h	

⌋

8.4.3.2 Mirror_Offline

[SWS_Mirror_01015] Definition of API function Mirror_Offline ⌈

Service Name	Mirror_Offline

Syntax	`void Mirror_Offline (`
	`  void`
	`)`

Service ID [hex]	0x13
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Completely disables any mirroring activities. Source buses are reset to disabled, queued messages are purged, and the destination bus is reset to the default destination. Pre-configured filters are disabled, and filters added at runtime are removed.
Available via	Mirror.h

⌋

8.4.3.3 Mirror_GetDestNetwork

[SWS_Mirror_01016] Definition of API function Mirror_GetDestNetwork ⌈

Service Name	Mirror_GetDestNetwork	

Syntax	`NetworkHandleType Mirror_GetDestNetwork (`
	`  void`
	`)`	

Service ID [hex]	0x21	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	NetworkHandleType	ComM channel that corresponds to the currently active destination network.
Description	Returns the currently selected destination bus.	
Available via	Mirror.h	

⌋

8.4.3.4 Mirror_SwitchDestNetwork

[SWS_Mirror_01017] Definition of API function Mirror_SwitchDestNetwork ⌈

Service Name	Mirror_SwitchDestNetwork	
Syntax	`Std_ReturnType Mirror_SwitchDestNetwork ( NetworkHandleType network )`	
Service ID [hex]	0x12	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	network	ComM channel corresponding to the destination bus that shall be enabled.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Destination bus was changed. `E_NOT_OK`: Function was called with invalid parameters.
Description	Changes the destination bus to the given ComM channel. The previously active destination bus and all source buses are disabled.	
Available via	Mirror.h	

⌋

8.4.3.5 Mirror_IsSourceNetworkStarted

[SWS_Mirror_01018] Definition of API function Mirror_IsSourceNetworkStarted ⌈

Service Name	Mirror_IsSourceNetworkStarted	
Syntax	`boolean Mirror_IsSourceNetworkStarted ( NetworkHandleType network )`	
Service ID [hex]	0x22	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	network	ComM channel corresponding to the source bus that shall be checked.
Parameters (inout)	None	
Parameters (out)	None	
Return value	boolean	TRUE: Source bus is active. FALSE: Source bus is inactive.
Description	Returns the state of a source bus.	
Available via	Mirror.h	

⌋

8.4.3.6 Mirror_StartSourceNetwork

[SWS_Mirror_01019] Definition of API function Mirror_StartSourceNetwork ⌈

Service Name	Mirror_StartSourceNetwork	
Syntax	`Std_ReturnType Mirror_StartSourceNetwork ( NetworkHandleType network )`	
Service ID [hex]	0x10	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different networks. Non reentrant for the same network.	
Parameters (in)	network	ComM channel corresponding to the source bus that shall be started.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Source bus was activated. `E_NOT_OK`: Function was called with invalid parameters.
Description	Activates a source bus.	
Available via	Mirror.h	

⌋

8.4.3.7 Mirror_StopSourceNetwork

[SWS_Mirror_01020] Definition of API function Mirror_StopSourceNetwork ⌈

Service Name	Mirror_StopSourceNetwork	
Syntax	`Std_ReturnType Mirror_StopSourceNetwork ( NetworkHandleType network )`	
Service ID [hex]	0x11	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different networks. Non reentrant for the same network.	
Parameters (in)	network	ComM channel corresponding to the source bus that shall be stopped.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Source bus was deactivated. `E_NOT_OK`: Function was called with invalid parameters.
Description	Deactivates a source bus.	
Available via	Mirror.h	

⌋

8.4.4 Support Functions

8.4.4.1 Mirror_GetNetworkType

[SWS_Mirror_01021] Definition of API function Mirror_GetNetworkType ⌈

Service Name	Mirror_GetNetworkType	
Syntax	`Mirror_NetworkType Mirror_GetNetworkType ( NetworkHandleType network )`	
Service ID [hex]	0x24	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	network	ComM channel corresponding to one of the buses configured as source or destination bus.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Mirror_NetworkType	Network type of the bus identified by 'network', or MIRROR_NT_INVALID if the bus is not configured for Mirror.
Description	Returns the network type of the given network.	
Available via	Mirror.h	

⌋

8.4.4.2 Mirror_GetNetworkId

[SWS_Mirror_01022] Definition of API function Mirror_GetNetworkId ⌈

Service Name	Mirror_GetNetworkId	
Syntax	`uint8 Mirror_GetNetworkId ( NetworkHandleType network )`	
Service ID [hex]	0x25	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	network	ComM channel corresponding to one of the buses configured as source or destination bus.
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint8	Network ID of the bus identified by 'network', or 0xFF if the bus is not configured for Mirror.
Description	Returns the network ID of the given network.	
Available via	Mirror.h	

⌋

8.4.4.3 Mirror_GetNetworkHandle

[SWS_Mirror_01023] Definition of API function Mirror_GetNetworkHandle ⌈

Service Name	Mirror_GetNetworkHandle	
Syntax	`NetworkHandleType Mirror_GetNetworkHandle ( Mirror_NetworkType networkType, uint8 networkId )`	
Service ID [hex]	0x26	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	networkType	Network type of the bus to be identified.
	networkId	Network ID of the bus to be identified.
Parameters (inout)	None	
Parameters (out)	None	
Return value	NetworkHandleType	ComM channel that corresponds to the bus identified by the given network type and network ID. `MIRROR_INVALID_NETWORK`, if no configured network corresponds to the given combination of networkType and networkId.
Description	Returns the network handle (ComMChannel) of the bus identified by the given network type and network ID, or MIRROR_INVALID_NETWORK.	
Available via	Mirror.h	

⌋

8.5 Callback Notifications

This is a list of functions provided for other modules.

8.5.1 Mirror_ReportCanFrame

[SWS_Mirror_01024] Definition of callback function Mirror_ReportCanFrame

Upstream requirements: SRS_BSW_00483, SRS_BSW_00486

⌈

Service Name	Mirror_ReportCanFrame
Syntax	`void Mirror_ReportCanFrame ( uint8 controllerId, Can_IdType canId, uint8 length, const uint8* payload )`
Service ID [hex]	0x50

▽

△

Sync/Async	Synchronous	
Reentrancy	Reentrant for different controllerIds. Non reentrant for the same controllerId.	
Parameters (in)	controllerId	ID of the CAN controller that received or transmitted the frame.
	canId	CAN ID of the CAN frame.
	length	Length of the CAN frame.
	payload	Content of the CAN frame.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Reports a received or transmitted CAN frame. All received CAN frames that pass the hardware acceptance filter are reported, independent of the software filter configuration. Transmitted CAN frames are reported when the transmission is confirmed.	
Available via	Mirror.h	

⌋

8.5.2 Mirror_ReportLinFrame

[SWS_Mirror_01027] Definition of callback function Mirror_ReportLinFrame

Upstream requirements: SRS_BSW_00486

⌈

Service Name	Mirror_ReportLinFrame	

Syntax	`void Mirror_ReportLinFrame (`
	`  NetworkHandleType network,`
	`  Lin_FramePidType pid,`
	`  const PduInfoType* pdu,`
	`  Lin_StatusType status`
	`)`	

Service ID [hex]	0x51	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different networks. Non reentrant for the same network.	
Parameters (in)	network	ComM channel associated with the LIN channel on which the frame was received or transmitted.
	pid	Protected ID of the LIN frame.
	pdu	Content of the LIN frame.
	status	Rx/Tx status of the frame access through the LIN driver.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Reports a received or transmitted LIN frame.	
Available via	Mirror.h	

⌋

8.5.3 Mirror_ReportFlexRayFrame

[SWS_Mirror_01026] Definition of callback function Mirror_ReportFlexRayFrame

Upstream requirements: SRS_BSW_00486

⌈

Service Name	Mirror_ReportFlexRayFrame	
Syntax	`void Mirror_ReportFlexRayFrame ( uint8 controllerId, uint16 slotId, uint8 cycle, Fr_ChannelType frChannel, const PduInfoType* frame, boolean txConflict )`	
Service ID [hex]	0x52	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different controllerIds. Non reentrant for the same controllerId.	
Parameters (in)	controllerId	FlexRay controller that received/transmitted the frame.
	slotId	ID of the slot in which the received/transmitted frame is located.
	cycle	Cycle in which the reception/transmission takes place.
	frChannel	FlexRay channel(s) on which the reception/transmission takes place.
	frame	Content of the FlexRay frame, or NULL when a txConflict is reported.
	txConflict	TRUE in case a txConflict has been detected, FALSE otherwise.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Reports a received or transmitted FlexRay frame or a Tx conflict.	
Available via	Mirror.h	

⌋

8.5.4 Mirror_ReportFlexRayChannelStatus

[SWS_Mirror_01025] Definition of callback function Mirror_ReportFlexRayChannelStatus ⌈

Service Name	Mirror_ReportFlexRayChannelStatus
Syntax	`void Mirror_ReportFlexRayChannelStatus ( uint8 clusterId, uint16 channelAStatus, uint16 channelBStatus )`
Service ID [hex]	0x53

▽

△

Sync/Async	Synchronous	
Reentrancy	Reentrant for different clusterIds. Non reentrant for the same clusterId.	
Parameters (in)	clusterId	FlexRay cluster for which the status is reported.
	channelAStatus	Status of FlexRay channel A.
	channelBStatus	Status of FlexRay channel B.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Reports the aggregated channel status for FlexRay channels A and B of a cluster. The status is encoded as specified in SWS_Fr_00558.	
Available via	Mirror.h	

⌋

8.5.5 Mirror_TxConfirmation

[SWS_Mirror_01028] Definition of callback function Mirror_TxConfirmation ⌈

Service Name	Mirror_TxConfirmation	
Syntax	`void Mirror_TxConfirmation ( PduIdType TxPduId, Std_ReturnType result )`	
Service ID [hex]	0x40	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	ID of the PDU that has been transmitted.
	result	E_OK: The PDU was transmitted. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	
Available via	Mirror.h	

⌋

8.5.6 Mirror_TriggerTransmit

[SWS_Mirror_01029] Definition of callback function Mirror_TriggerTransmit

Upstream requirements: SRS_BSW_00461, SRS_BSW_00486

⌈

Service Name	Mirror_TriggerTransmit	
Syntax	`Std_ReturnType Mirror_TriggerTransmit ( PduIdType TxPduId, PduInfoType* PduInfoPtr )`	
Service ID [hex]	0x41	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	ID of the SDU that is requested to be transmitted.
Parameters (inout)	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLengh. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: SDU has been copied and SduLength indicates the number of copied bytes. `E_NOT_OK`: No SDU data has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description	Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by PduInfoPtr->SduLength. If it fits, it shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->SduLength. If not, it returns E_NOT_OK without changing PduInfoPtr.	
Available via	Mirror.h	

⌋

8.6 Scheduled Functions

This function is directly called by Basic Software Scheduler (SchM).

8.6.1 Mirror_MainFunction

[SWS_Mirror_01030] Definition of scheduled function Mirror_MainFunction ⌈

Service Name	Mirror_MainFunction
Syntax	`void Mirror_MainFunction ( void )`

▽

△

Service ID [hex]	0x04
Description	Main function of the Bus Mirroring module. Used for scheduling purposes and timeout supervision. Per configured MirrorMainFunction instance one Mirror_MainFunction_<shortName> shall be implemented. Hereby <shortName> is the short name of the MirrorMainFunction configuration container in the ECU configuration.
Available via	SchM_Mirror.h

⌋

8.7 Expected Interfaces

In this section, all interfaces required from other modules are listed.

8.7.1 Mandatory Interfaces

This section defines all interfaces that are required to fulfill the core functionality of the module.

[SWS_Mirror_01101] Definition of mandatory interfaces required by module Mirror ⌈

API Function	***Header File***	***Description***
PduR_MirrorTransmit	PduR_Mirror.h	Requests transmission of a PDU.

⌋

8.7.2 Optional Interfaces

This section defines all interfaces that are required to fulfill an optional functionality of the module.

[SWS_Mirror_01102] Definition of optional interfaces requested by module Mirror
⌈

API Function	*Header File*	*Description*
CanIf_EnableBusMirroring	CanIf.h	Enables or disables mirroring for a CAN controller.
CanIf_GetControllerErrorState	CanIf.h	This service calls the corresponding CAN Driver service for obtaining the error state of the CAN controller.
CanIf_GetControllerMode	CanIf.h	This service calls the corresponding CAN Driver service for obtaining the current status of the CAN controller.
CanIf_GetControllerTxErrorCounter	CanIf.h	This service calls the corresponding CAN Driver service for obtaining the Tx error counter of the CAN controller.
CanIf_GetTrcvMode	CanIf.h	This function invokes CanTrcv_GetOpMode and updates the parameter TransceiverModePtr with the value OpMode provided by CanTrcv.
Det_ReportError	Det.h	Service to report development errors.
FrIf_EnableBusMirroring	FrIf.h	Enables or disables mirroring for all FlexRay controllers connected to the addressed FlexRay cluster.
FrIf_GetPOCStatus	FrIf.h	Wraps the FlexRay Driver API function Fr_Get POCStatus().
FrIf_GetState	FrIf.h	Get current FrIf state.
LinIf_EnableBusMirroring	LinIf.h	Enables or disables mirroring for a LIN channel.
LinIf_GetTrcvMode	LinIf.h	Returns the actual state of a LIN Transceiver Driver.
StbM_GetCurrentTime	StbM.h	Returns a time tuple (Local time, Global time and Timebase status) and user data details Note: This API shall be called with locked interrupts / within an Exclusive Area to prevent interruption (i.e., the risk that the time stamp is outdated on return of the function call).

⌋

8.8 Service Interfaces

8.8.1 Implementation Data Types

8.8.1.1 Mirror_NetworkType

[SWS_Mirror_01000] Definition of ImplementationDataType Mirror_NetworkType
⌈

Name	Mirror_NetworkType		
Kind	Type		
Derived from	uint8		
Range	MIRROR_NT_INVALID	0x00	Invalid network
	MIRROR_NT_CAN	0x01	CAN network

▽

△

	MIRROR_NT_LIN	0x02	LIN network
	MIRROR_NT_FLEXRAY	0x03	FlexRay network
	MIRROR_NT_ETHERNET	0x04	Ethernet network
	MIRROR_NT_ PROPRIETARY	0x05	Proprietary network
	MIRROR_NT_CAN_XL	0x06	CAN XL network
Description	This type represents the bus types that are supported as source or destination buses for the Bus Mirroring module. The invalid type is used as a return value if a function cannot return a valid type.		
Variation	–		
Available via	Rte_Mirror_Type.h		

⌋

8.8.2 Client-Server Interfaces

8.8.2.1 MirrorControl

[SWS_Mirror_01033] Definition of ClientServerInterface MirrorControl

Upstream requirements: SRS_BSW_00462

⌈

Name	MirrorControl		
Comment	Provides access to the control functions of the Bus Mirroring module.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	AddCanMaskFilter	
Comment	Creates a CAN ID mask filter.	
Mapped to API	Mirror_AddCanMaskFilter	
Variation	–	
Parameters	network	
	Type	NetworkHandleType
	Direction	IN
	Comment	ComM channel that corresponds to the CAN bus to which the filter shall be attached.
	Variation	–
	filterId	
	Type	uint8*
	Direction	OUT
	Comment	ID of the newly created filter.
	Variation	–

▽

△

	id	
	Type	Can_IdType
	Direction	IN
	Comment	CAN ID used to match a received or transmitted CAN ID.
	Variation	–
	mask	
	Type	Can_IdType
	Direction	IN
	Comment	Mask that defines the bits of 'id' that are relevant for comparison with the actual CAN ID.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	AddCanRangeFilter	
Comment	Creates a CAN ID range filter.	
Mapped to API	Mirror_AddCanRangeFilter	
Variation	–	
Parameters	network	
	Type	NetworkHandleType
	Direction	IN
	Comment	ComM channel that corresponds to the CAN bus to which the filter shall be attached.
	Variation	–
	filterId	
	Type	uint8*
	Direction	OUT
	Comment	ID of the newly created filter.
	Variation	–
	lowerId	
	Type	Can_IdType
	Direction	IN
	Comment	Lower CAN ID of the range.
	Variation	–
	upperId	
	Type	Can_IdType
	Direction	IN
	Comment	Upper CAN ID of the range.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	AddFlexRayFilter	
Comment	Creates a FlexRay filter.	
Mapped to API	Mirror_AddFlexRayFilter	
Variation	–	
Parameters	network	
	Type	NetworkHandleType

▽

△

	Direction	IN
	Comment	ComM channel that corresponds to the FlexRay bus to which the filter shall be attached.
	Variation	–
	filterId	
	Type	uint8*
	Direction	OUT
	Comment	ID of the newly created filter.
	Variation	–
	lowerSlotId	
	Type	uint16
	Direction	IN
	Comment	Lower slot ID of a range of slot IDs.
	Variation	–
	upperSlotId	
	Type	uint16
	Direction	IN
	Comment	Upper slot ID of a range of slot IDs.
	Variation	–
	lowerBaseCycle	
	Type	uint8
	Direction	IN
	Comment	Lower base cycle of a range of cycles.
	Variation	–
	upperBaseCycle	
	Type	uint8
	Direction	IN
	Comment	Upper base cycle of a range of cycles.
	Variation	–
	cycleRepetition	
	Type	uint8
	Direction	IN
	Comment	Repetition pattern of selected cycles (2^n).
	Variation	–
	frChannel	
	Type	Fr_ChannelType
	Direction	IN
	Comment	FlexRay channel assignment.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	AddLinMaskFilter
Comment	Creates a LIN frame ID mask filter.
Mapped to API	Mirror_AddLinMaskFilter
Variation	–

▽

△

Parameters	network	
	Type	NetworkHandleType
	Direction	IN
	Comment	ComM channel that corresponds to the LIN bus to which the filter shall be attached.
	Variation	–
	filterId	
	Type	uint8*
	Direction	OUT
	Comment	ID of the newly created filter.
	Variation	–
	id	
	Type	uint8
	Direction	IN
	Comment	Frame ID used to match a received or transmitted frame ID.
	Variation	–
	mask	
	Type	uint8
	Direction	IN
	Comment	Mask that defines the bits of 'id' that are relevant for comparison with the actual frame ID.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	AddLinRangeFilter	
Comment	Creates a LIN frame ID range filter.	
Mapped to API	Mirror_AddLinRangeFilter	
Variation	–	
Parameters	network	
	Type	NetworkHandleType
	Direction	IN
	Comment	ComM channel that corresponds to the LIN bus to which the filter shall be attached.
	Variation	–
	filterId	
	Type	uint8*
	Direction	OUT
	Comment	ID of the newly created filter.
	Variation	–
	lowerId	
	Type	uint8
	Direction	IN
	Comment	Lower frame ID of the range.
	Variation	–
	upperId	
	Type	uint8
	Direction	IN

▽

△

	Comment	Upper frame ID of the range.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	GetDestNetwork	
Comment	Returns the currently selected destination bus.	
Mapped to API	Mirror_GetDestNetwork	
Variation	–	
Parameters	network	
	Type	NetworkHandleType
	Direction	OUT
	Comment	ComM channel that corresponds to the currently active destination network.
	Variation	–
Possible Errors	E_OK	

Operation	GetNetworkHandle	
Comment	Returns the network handle (ComMChannel) of the bus identified by the given network type and network ID.	
Mapped to API	Mirror_GetNetworkHandle	
Variation	–	
Parameters	networkType	
	Type	Mirror_NetworkType
	Direction	IN
	Comment	Network type of the bus to be identified.
	Variation	–
	networkId	
	Type	uint8
	Direction	IN
	Comment	Network ID of the bus to be identified.
	Variation	–
	network	
	Type	NetworkHandleType
	Direction	OUT
	Comment	ComM channel that corresponds to the bus identified by the given network type and network ID.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	GetNetworkId	
Comment	Returns the network ID of the given network.	
Mapped to API	Mirror_GetNetworkId	
Variation	–	
Parameters	network	
	Type	NetworkHandleType
	Direction	IN

▽

△

	Comment	ComM channel corresponding to one of the buses configured as source or destination bus.
	Variation	–
	networkId	
	Type	uint8
	Direction	OUT
	Comment	Network ID of the bus identified by 'network'.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	GetNetworkType	
Comment	Returns the network type of the given network.	
Mapped to API	Mirror_GetNetworkType	
Variation	–	
Parameters	network	
	Type	NetworkHandleType
	Direction	IN
	Comment	ComM channel corresponding to one of the buses configured as source or destination bus.
	Variation	–
	networkType	
	Type	Mirror_NetworkType
	Direction	OUT
	Comment	Network type of the bus identified by 'network'.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	GetStaticFilterState	
Comment	Returns the state of a pre-configured filter.	
Mapped to API	Mirror_GetStaticFilterState	
Variation	–	
Parameters	network	
	Type	NetworkHandleType
	Direction	IN
	Comment	ComM channel that corresponds to the source bus to which the filter is attached.
	Variation	–
	filterId	
	Type	uint8
	Direction	IN
	Comment	ID of the filter.
	Variation	–
	isActive	
	Type	boolean*
	Direction	OUT
	Comment	Pointer to where to store the current filter state.

▽

△

	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	IsMirrorActive	
Comment	Returns the global mirroring state.	
Mapped to API	Mirror_IsMirrorActive	
Variation	–	
Parameters	mirrorActive	
	Type	boolean
	Direction	OUT
	Comment	Global mirroring state.
	Variation	–
Possible Errors	E_OK	

Operation	IsSourceNetworkStarted	
Comment	Returns the state of a source bus.	
Mapped to API	Mirror_IsSourceNetworkStarted	
Variation	–	
Parameters	network	
	Type	NetworkHandleType
	Direction	IN
	Comment	ComM channel corresponding to the source bus that shall be checked.
	Variation	–
	sourceNetworkStarted	
	Type	boolean
	Direction	OUT
	Comment	State of a source bus. TRUE: Source bus is active. FALSE: Source bus is inactive.
	Variation	–
Possible Errors	E_OK	

Operation	Offline
Comment	Completely disables any mirroring activities. Source buses are reset to disabled, queued messages are purged, and the destination bus is reset to the default destination. Pre-configured filters are disabled, and filters added at runtime are removed.
Mapped to API	Mirror_Offline
Variation	–
Possible Errors	E_OK

Operation	RemoveFilter	
Comment	Removes a CAN, LIN, or FlexRay filter that was added at runtime.	
Mapped to API	Mirror_RemoveFilter	
Variation	–	
Parameters	network	
	Type	NetworkHandleType
	Direction	IN

▽

△

	Comment	ComM channel that corresponds to the source bus to which the filter is attached.
	Variation	–
	filterId	
	Type	uint8
	Direction	IN
	Comment	ID of the filter.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	SetStaticFilterState	
Comment	Sets the state of a pre-configured filter.	
Mapped to API	Mirror_SetStaticFilterState	
Variation	–	
Parameters	network	
	Type	NetworkHandleType
	Direction	IN
	Comment	ComM channel that corresponds to the source bus to which the filter is attached.
	Variation	–
	filderId	
	Type	uint8
	Direction	IN
	Comment	ID of the filter.
	Variation	–
	isActive	
	Type	boolean
	Direction	IN
	Comment	TRUE: Activate filter FALSE: Deactivate filter
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	StartSourceNetwork	
Comment	Activates a source bus.	
Mapped to API	Mirror_StartSourceNetwork	
Variation	–	
Parameters	network	
	Type	NetworkHandleType
	Direction	IN
	Comment	ComM channel corresponding to the source bus that shall be started.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	StopSourceNetwork	
Comment	Deactivates a source bus.	
Mapped to API	Mirror_StopSourceNetwork	
Variation	–	
Parameters	network	
	Type	NetworkHandleType
	Direction	IN
	Comment	ComM channel corresponding to the source bus that shall be stopped.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	SwitchDestNetwork	
Comment	Changes the destination bus to the given ComM channel. The previously active destination bus and all source buses are disabled.	
Mapped to API	Mirror_SwitchDestNetwork	
Variation	–	
Parameters	network	
	Type	NetworkHandleType
	Direction	IN
	Comment	ComM channel corresponding to the destination bus that shall be enabled.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋

8.8.3 Provided Ports

8.8.3.1 MirrorControl

[SWS_Mirror_01031] Definition of Port MirrorControl provided by module Mirror
⌈

Name	MirrorControl		
Kind	ProvidedPort	***Interface***	MirrorControl
Description	Provided port for the interface MirrorControl.		
Variation	–		

⌋

9 Sequence Diagrams

Currently, no sequence diagrams are available.

10 Configuration Specification

In general, this chapter defines configuration parameters and their clustering into containers. For general information about the definition of containers and parameters, refer to the Section 10.1 “Introduction to configuration specification” in [2, SWS BSW General].

Section 10.1 specifies the structure (containers) and the parameters of the Bus Mirroring module.

Section 10.2 lists constraints on the configuration of the Bus Mirroring module.

Section 10.3 specifies published information of the Bus Mirroring module.

10.1 Containers and Configuration Parameters

The following sections summarize all configuration parameters of the Bus Mirroring module. The detailed meaning of the parameters is described in Chapter 7 and Chapter 8.

Some of these containers and parameters are derived from classes and attributes of the [5, TPS System Template], which also contains the rules for these derivations.

The following pictures show an overview of the configuration parameters available for the Bus Mirroring module:

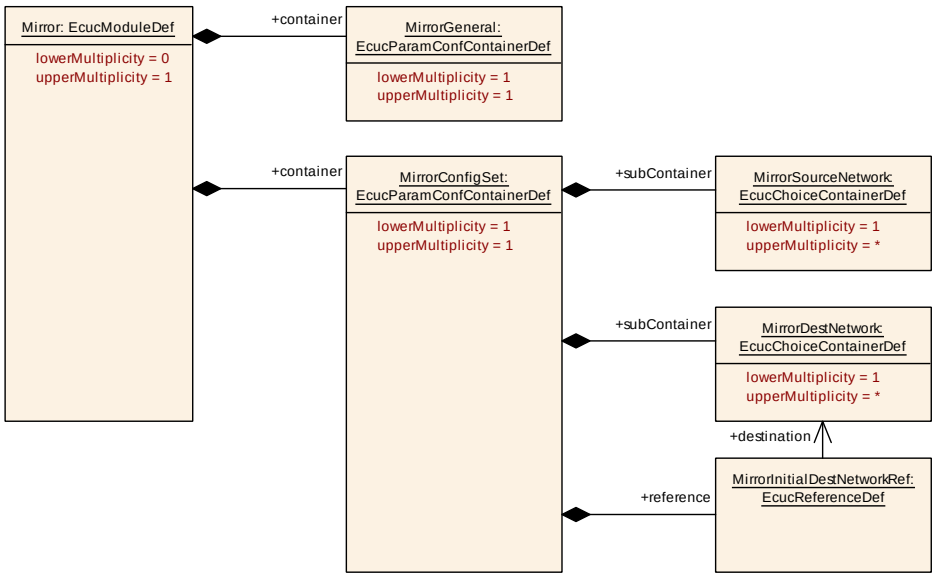

Figure 10.1: Configuration container Mirror with sub-container MirrorConfigSet

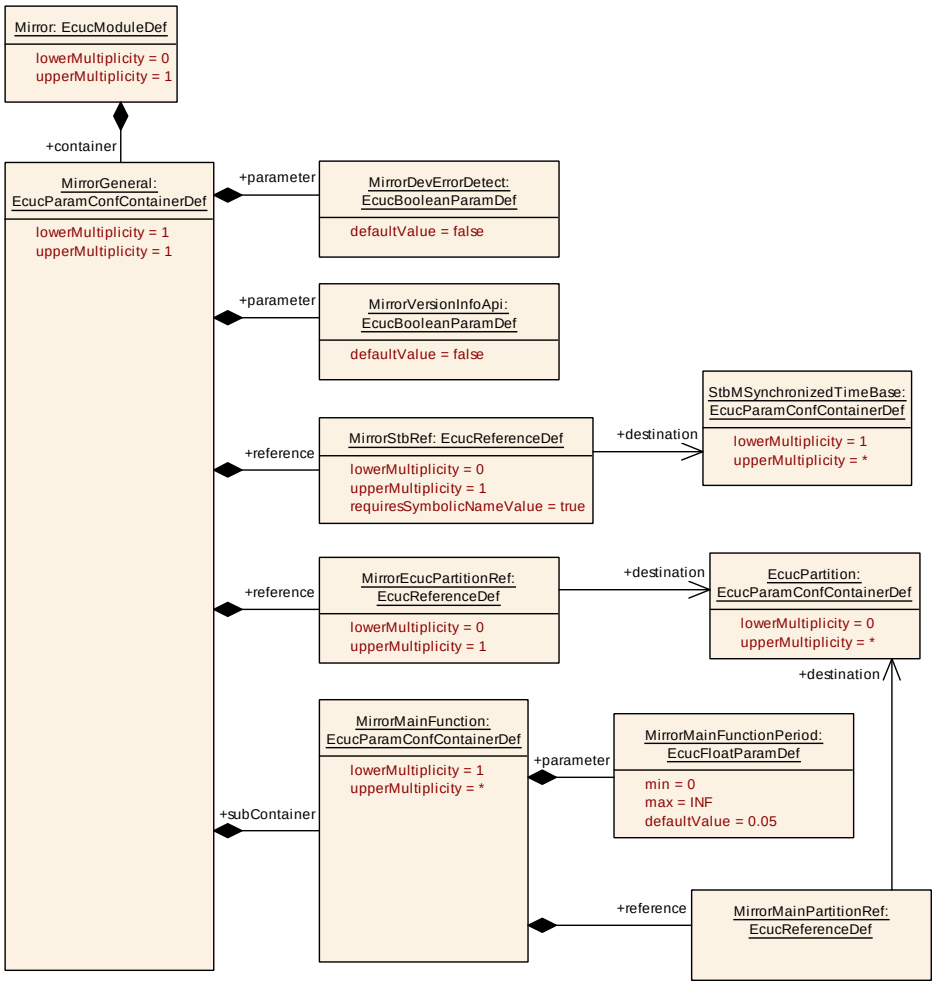

Figure 10.2: Configuration container MirrorGeneral

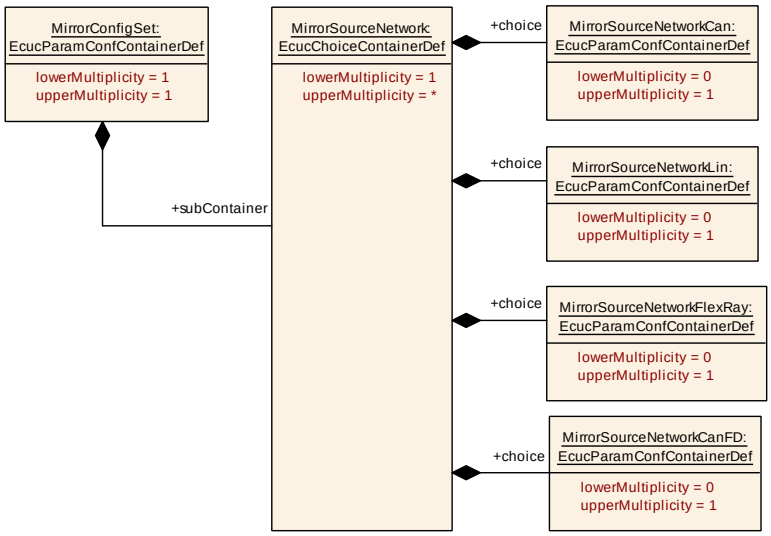

Figure 10.3: Configuration container MirrorSourceNetwork

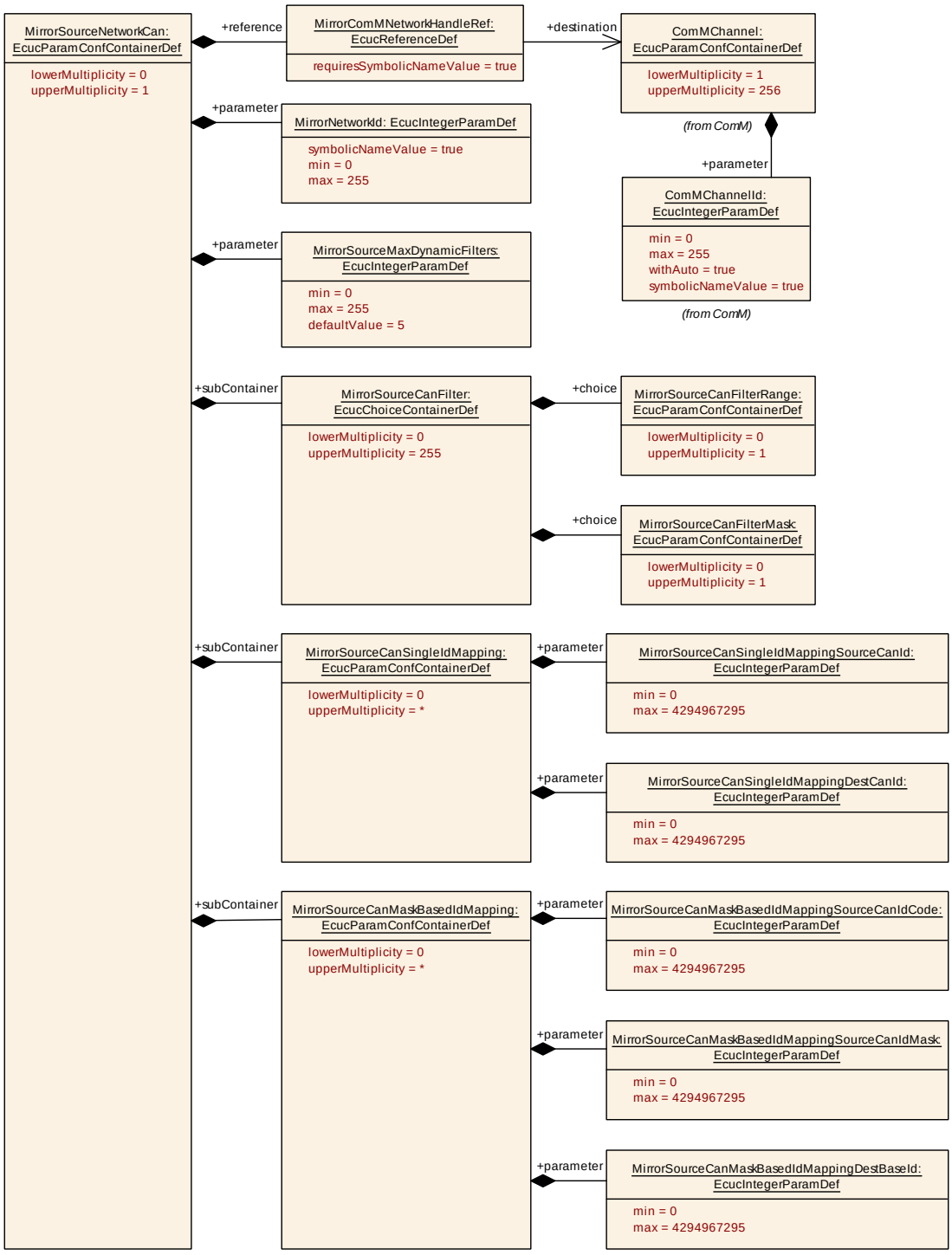

Figure 10.4: Configuration container MirrorSourceNetworkCan

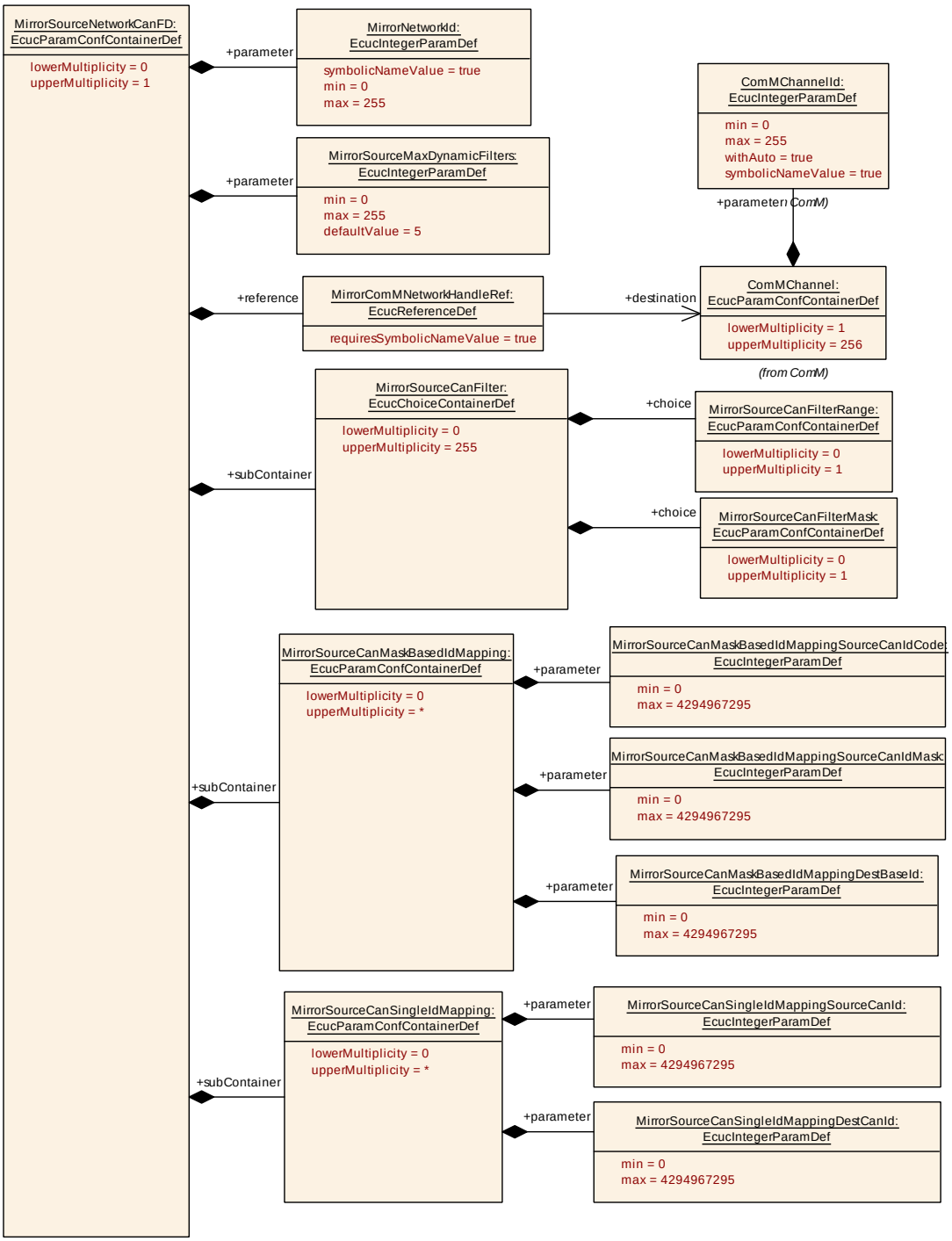

Figure 10.5: Configuration container MirrorSourceNetworkCanFD

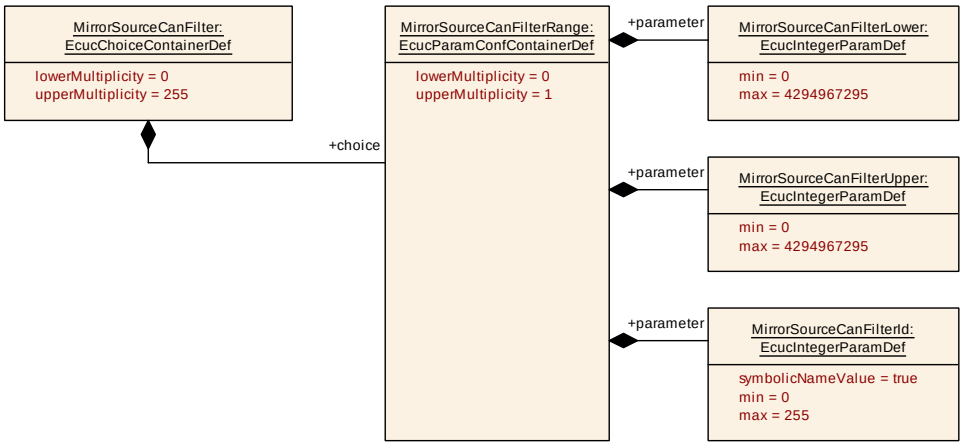

Figure 10.6: Configuration container MirrorSourceNetworkCanFilterRange

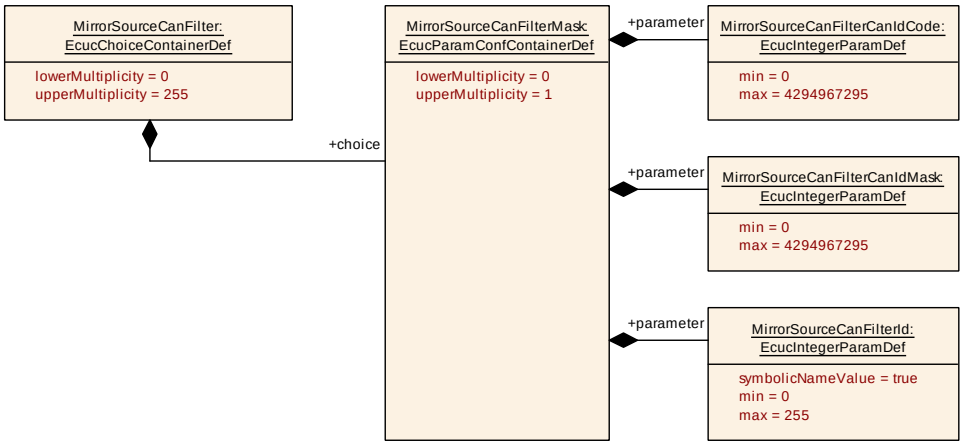

Figure 10.7: Configuration container MirrorSourceNetworkCanFilterMask

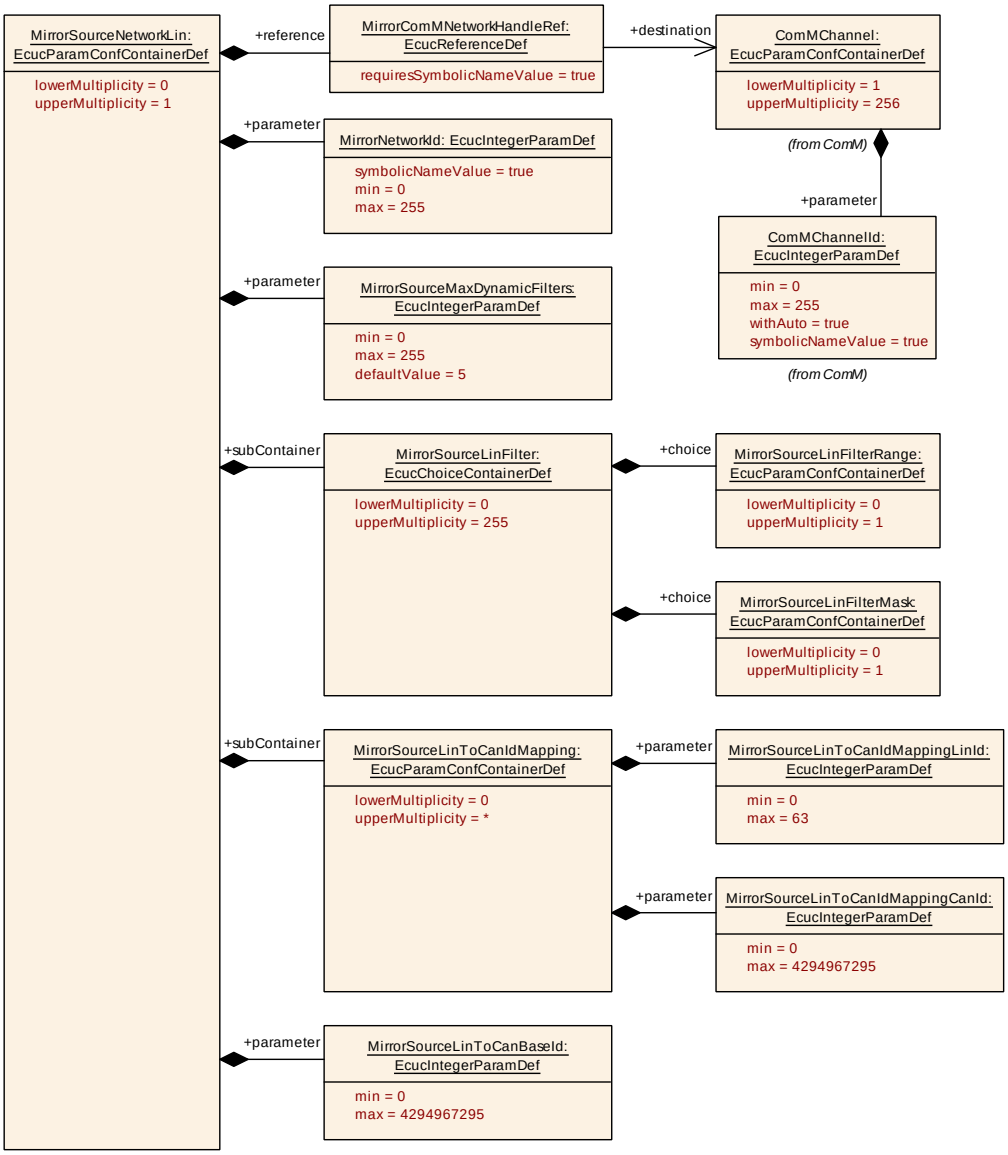

Figure 10.8: Configuration container MirrorSourceNetworkLin

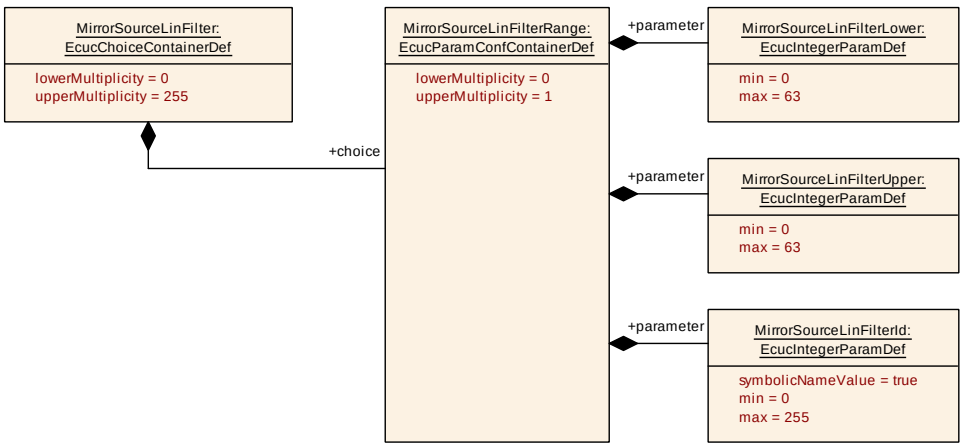

Figure 10.9: Configuration container MirrorSourceNetworkLinFilterRange

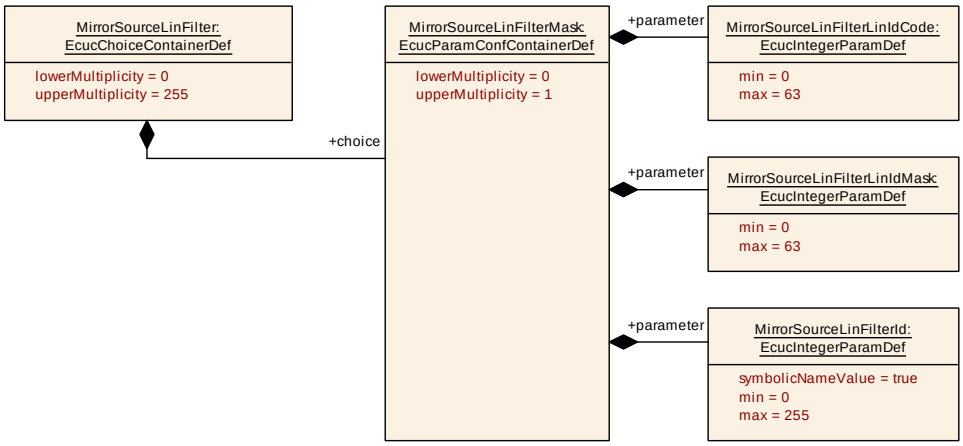

Figure 10.10: Configuration container MirrorSourceNetworkLinFilterMask

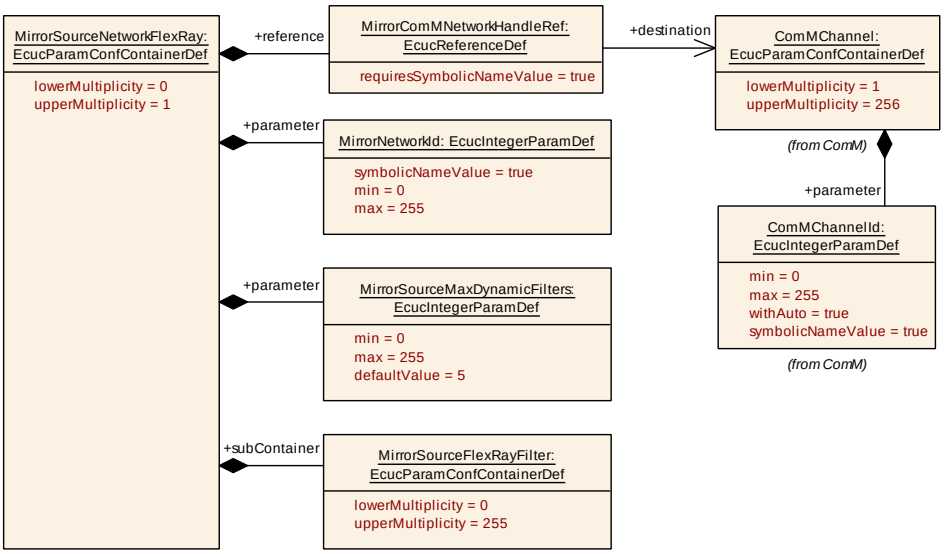

Figure 10.11: Configuration container MirrorSourceNetworkFlexRay

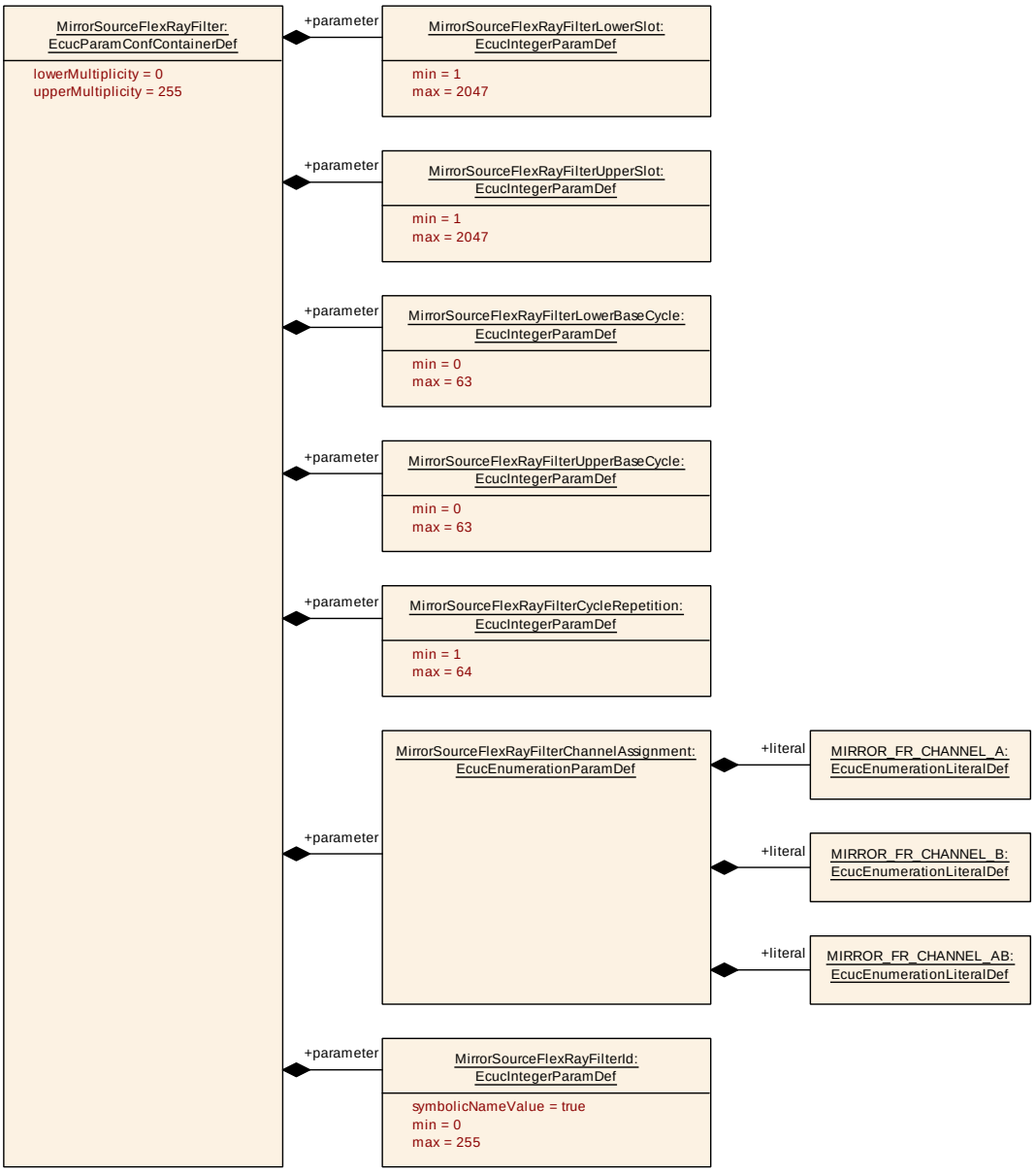

Figure 10.12: Configuration container MirrorSourceNetworkFlexRayFilter

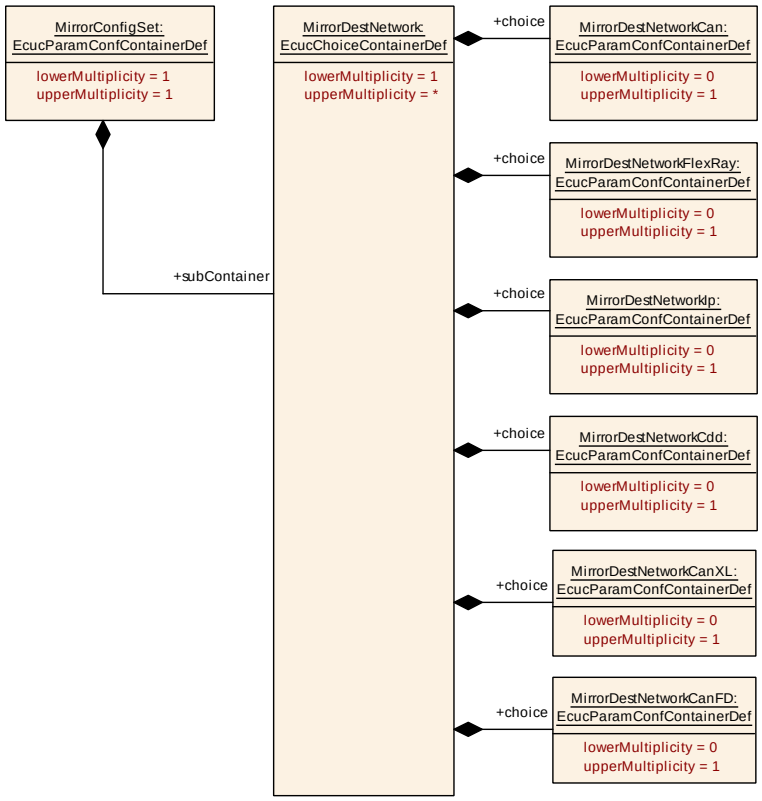

Figure 10.13: Configuration container MirrorDestNetwork

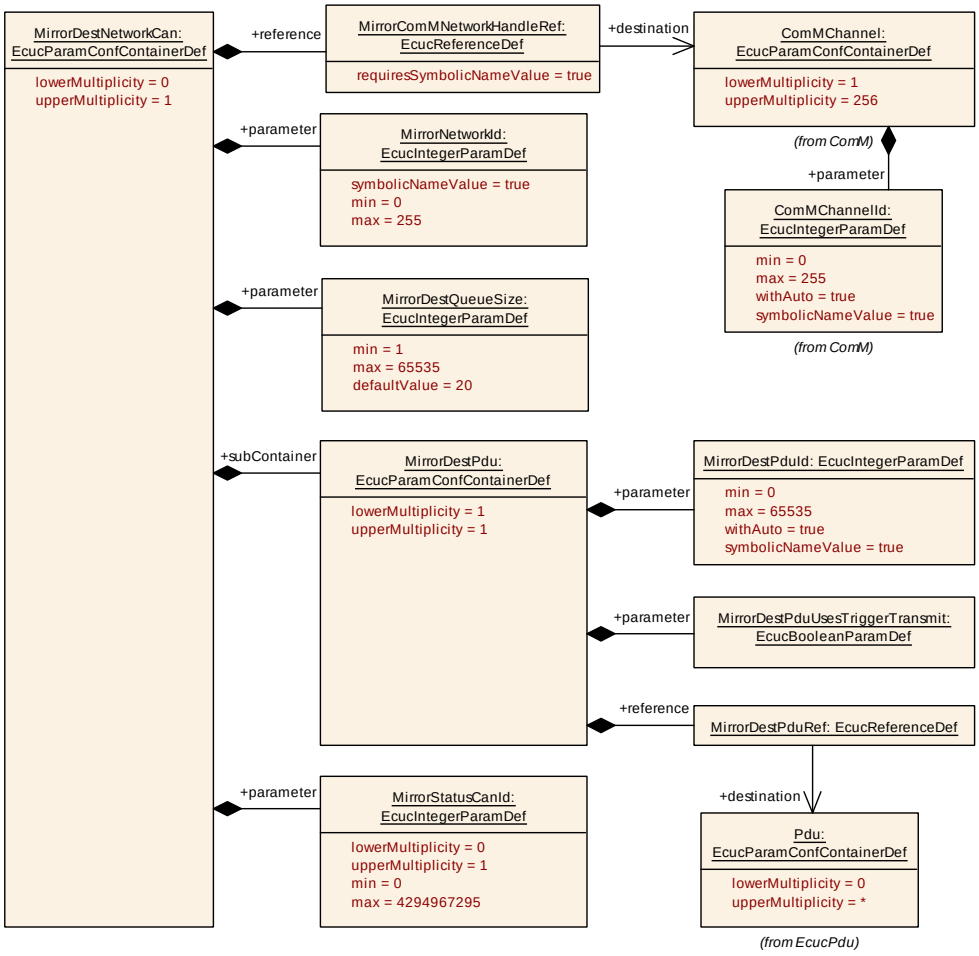

Figure 10.14: Configuration container MirrorDestNetworkCan

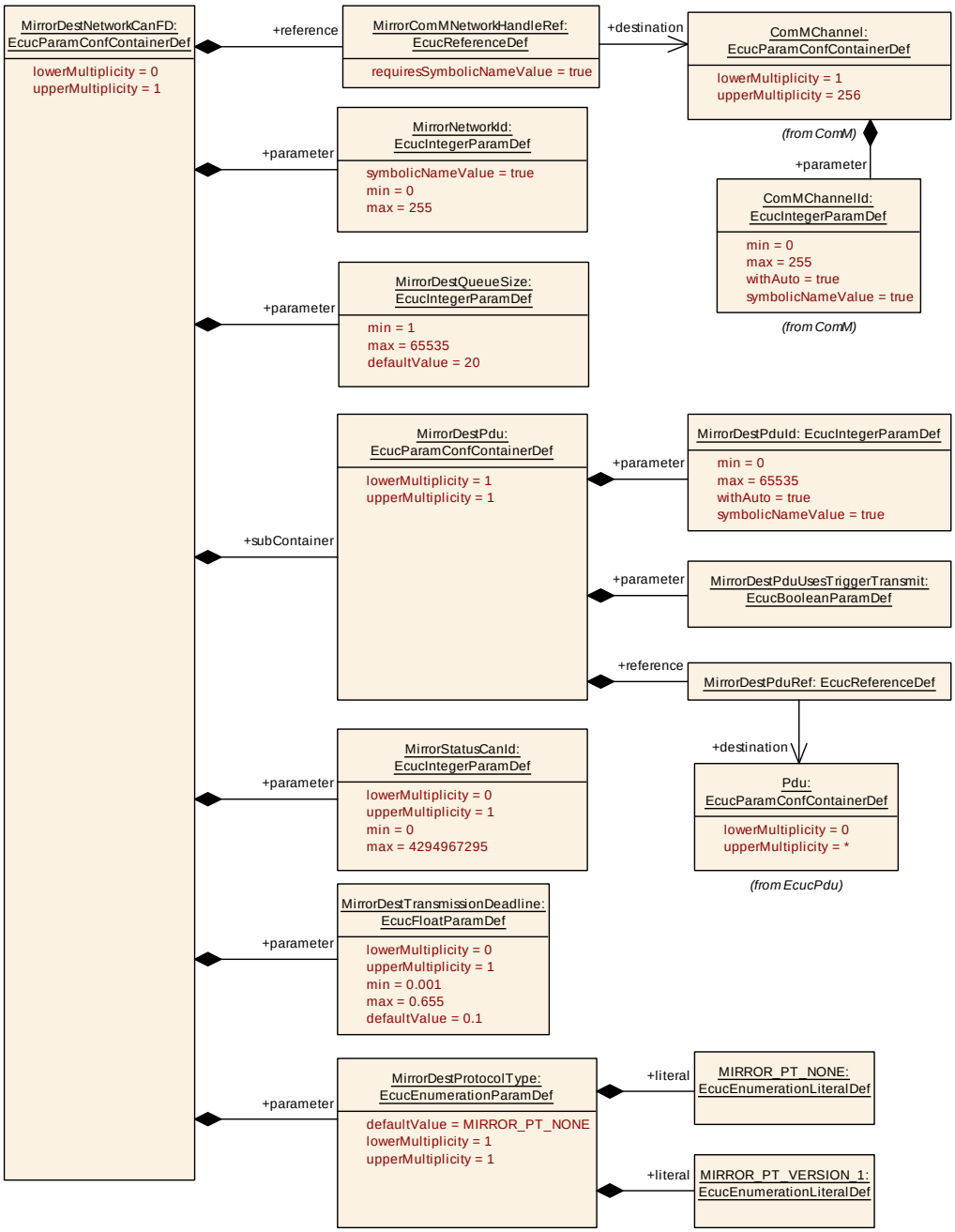

Figure 10.15: Configuration container MirrorDestNetworkCanFD

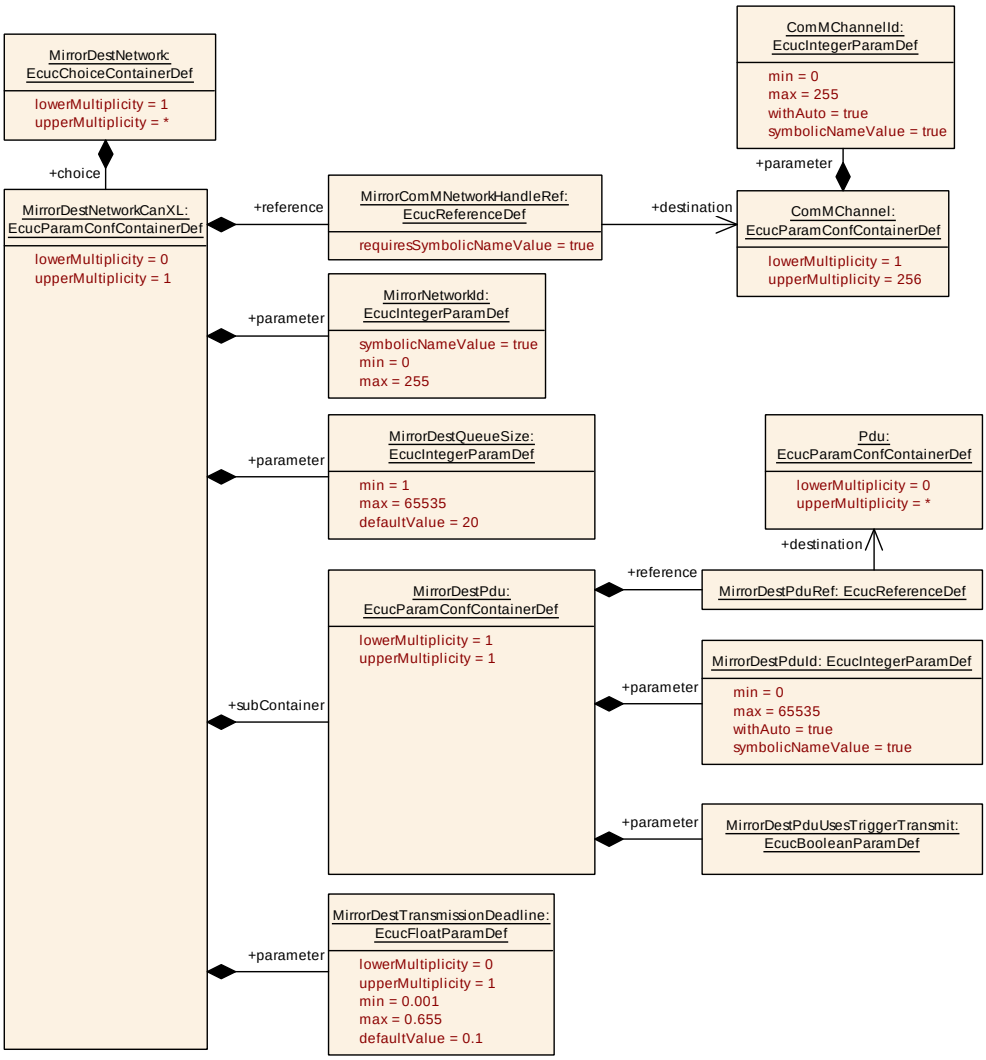

Figure 10.16: Configuration container MirrorDestNetworkCanXL

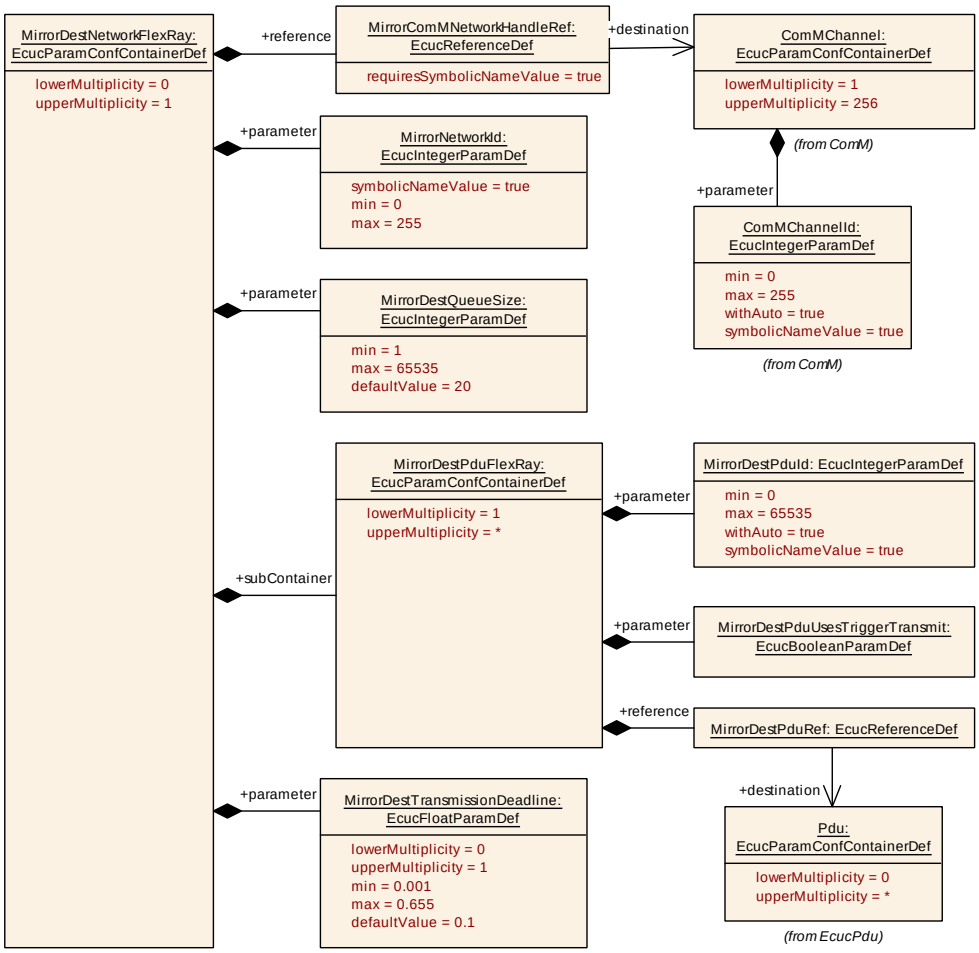

Figure 10.17: Configuration container MirrorDestNetworkFlexRay

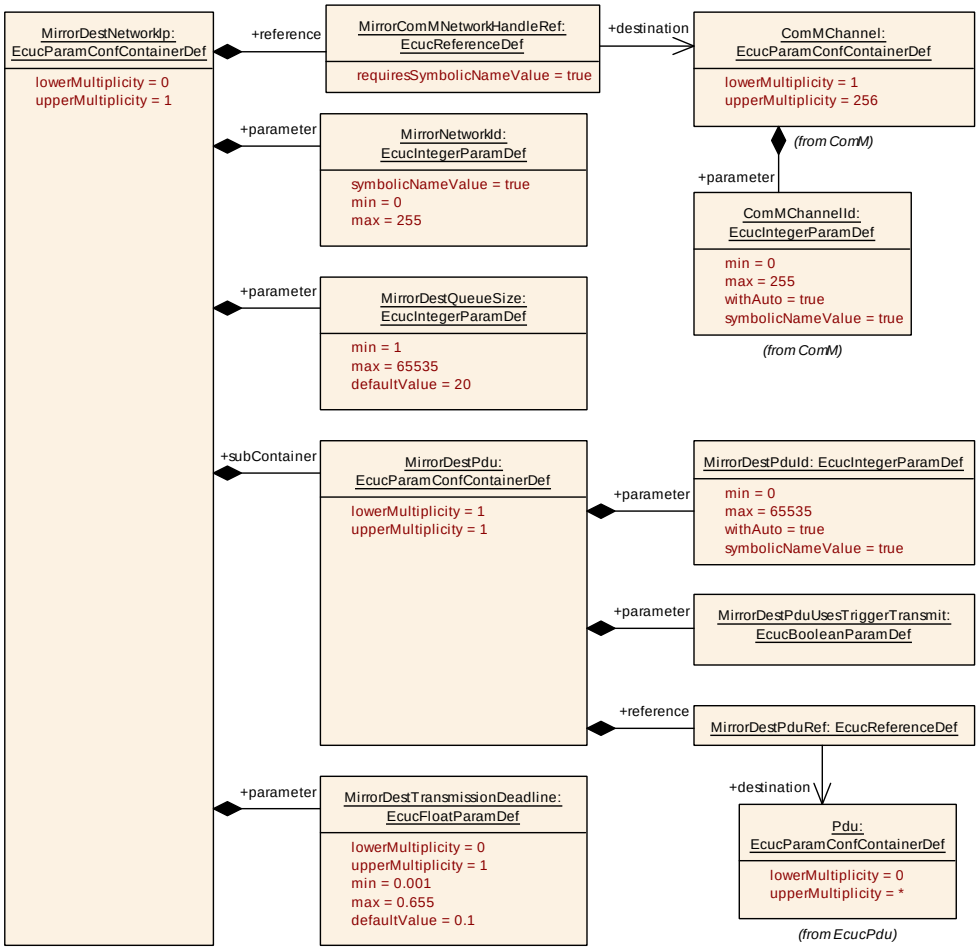

Figure 10.18: Configuration container MirrorDestNetworkIp

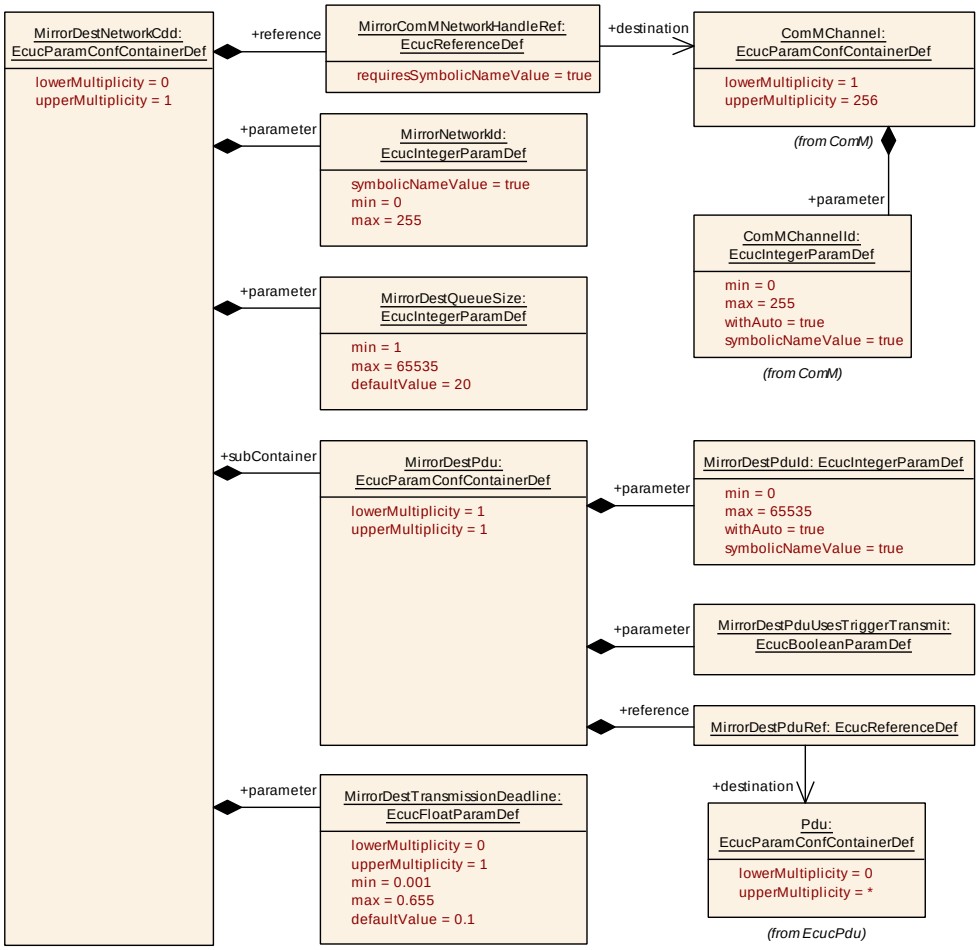

Figure 10.19: Configuration container MirrorDestNetworkCdd

10.1.1 Mirror

[ECUC_Mirror_00001] Definition of EcucModuleDef Mirror ⌈

Module Name	Mirror
Description	Configuration of the Bus Mirroring module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorConfigSet	1	Contains the configuration parameters and sub containers of the Bus Mirroring module.
MirrorGeneral	1	Contains the general configuration parameters of the module.

⌋

10.1.2 MirrorGeneral

[ECUC_Mirror_00002] Definition of EcucParamConfContainerDef MirrorGeneral
⌈

Container Name	MirrorGeneral
Parent Container	Mirror
Description	Contains the general configuration parameters of the module.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorDevErrorDetect	1	[ECUC_Mirror_00003]
MirrorVersionInfoApi	1	[ECUC_Mirror_00005]
MirrorEcucPartitionRef	0..1	[ECUC_Mirror_00067]
MirrorStbRef	0..1	[ECUC_Mirror_00065]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorMainFunction	1..*	Each element of this container defines one instance of Mirror_MainFunction.

⌋

[ECUC_Mirror_00003] Definition of EcucBooleanParamDef MirrorDevErrorDetect
⌈

Parameter Name	MirrorDevErrorDetect		
Parent Container	MirrorGeneral		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00005] Definition of EcucBooleanParamDef MirrorVersionInfoApi ⌈

Parameter Name	MirrorVersionInfoApi		
Parent Container	MirrorGeneral		
Description	Pre-processor switch for enabling version info API support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00067] Definition of EcucReferenceDef MirrorEcucPartitionRef ⌈

Parameter Name	MirrorEcucPartitionRef		
Parent Container	MirrorGeneral		
Description	Reference to EcucPartition, where BusMirroring module is assigned to.		
Multiplicity	0..1		
Type	Reference to EcucPartition		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00065] Definition of EcucReferenceDef MirrorStbRef ⌈

Parameter Name	MirrorStbRef		
Parent Container	MirrorGeneral		
Description	Reference to the StbM time base to use for acquiring the time stamps used in the mirroring protocol. This reference is not required if all destination buses are CAN.		
Multiplicity	0..1		
Type	Symbolic name reference to StbMSynchronizedTimeBase		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants

▽

△

	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.1.3 MirrorMainFunction

[ECUC_Mirror_00068] Definition of EcucParamConfContainerDef MirrorMainFunction ⌈

Container Name	MirrorMainFunction		
Parent Container	MirrorGeneral		
Description	Each element of this container defines one instance of Mirror_MainFunction.		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorMainFunctionPeriod	1	[ECUC_Mirror_00070]
MirrorMainPartitionRef	1	[ECUC_Mirror_00069]

No Included Containers

⌋

[ECUC_Mirror_00070] Definition of EcucFloatParamDef MirrorMainFunctionPeriod ⌈

Parameter Name	MirrorMainFunctionPeriod		
Parent Container	MirrorMainFunction		
Description	Execution cycle of the respective Mirror_MainFunction instance in seconds.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	0.05		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

▽

△

Scope / Dependency	scope: local

⌋

[ECUC_Mirror_00069] Definition of EcucReferenceDef MirrorMainPartitionRef ⌈

Parameter Name	MirrorMainPartitionRef		
Parent Container	MirrorMainFunction		
Description	Reference to EcucPartition, where the according Mirror_MainFunction instance is assigned to.		
Multiplicity	1		
Type	Reference to EcucPartition		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.1.4 MirrorConfigSet

[ECUC_Mirror_00008] Definition of EcucParamConfContainerDef MirrorConfigSet ⌈

Container Name	MirrorConfigSet
Parent Container	Mirror
Description	Contains the configuration parameters and sub containers of the Bus Mirroring module.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorInitialDestNetworkRef	1	[ECUC_Mirror_00007]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorDestNetwork	1..*	Destination bus to which frames are sent by the Bus Mirroring module.
MirrorSourceNetwork	1..*	Source bus from which frames are received by the Bus Mirroring module.

⌋

[ECUC_Mirror_00007] Definition of EcucReferenceDef MirrorInitialDestNetwork Ref ⌈

Parameter Name	MirrorInitialDestNetworkRef		
Parent Container	MirrorConfigSet		
Description	Reference to the destination bus that is selected after initialization of the Bus Mirroring module.		
Multiplicity	1		
Type	Reference to MirrorDestNetwork		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.1.5 MirrorSourceNetwork

[ECUC_Mirror_00009] Definition of EcucChoiceContainerDef MirrorSourceNetwork ⌈

Choice Container Name	MirrorSourceNetwork		
Parent Container	MirrorConfigSet		
Description	Source bus from which frames are received by the Bus Mirroring module.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

No Included Parameters

Container Choices		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorSourceNetworkCan	0..1	Source bus representing a CAN network.
MirrorSourceNetworkCanFD	0..1	Source bus representing a CAN FD network.
MirrorSourceNetworkFlexRay	0..1	Source bus representing a FlexRay network.
MirrorSourceNetworkLin	0..1	Source bus representing a LIN network.

⌋

10.1.6 MirrorSourceNetworkCan

[ECUC_Mirror_00010] Definition of EcucParamConfContainerDef MirrorSourceNetworkCan ⌈

Container Name	MirrorSourceNetworkCan		
Parent Container	MirrorSourceNetwork		
Description	Source bus representing a CAN network.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorNetworkId	1	[ECUC_Mirror_00012]
MirrorSourceMaxDynamicFilters	1	[ECUC_Mirror_00013]
MirrorComMNetworkHandleRef	1	[ECUC_Mirror_00064]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorSourceCanFilter	0..255	Pre-configured filter for CAN frames.
MirrorSourceCanMaskBasedId Mapping	0..*	Rule for remapping a set of CAN IDs.
MirrorSourceCanSingleIdMapping	0..*	Rule for remapping a single CAN ID.

⌋

For parameter table [ECUC_Mirror_00012] MirrorNetworkId, see definition below container MirrorDestNetworkCan.

[ECUC_Mirror_00013] Definition of EcucIntegerParamDef MirrorSourceMaxDynamicFilters ⌈

Parameter Name	MirrorSourceMaxDynamicFilters		
Parent Container	MirrorSourceNetworkCan, MirrorSourceNetworkCanFD, MirrorSourceNetworkFlexRay, MirrorSourceNetworkLin		
Description	Maximum number of filters that can be dynamically added using Mirror_AddXxxFilter().		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	5		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE

▽

△

	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

For parameter table [ECUC_Mirror_00064] MirrorComMNetworkHandleRef, see definition below container MirrorDestNetworkCan.

10.1.7 MirrorSourceNetworkCanFD

[ECUC_Mirror_00072] Definition of EcucParamConfContainerDef MirrorSourceNetworkCanFD ⌈

Container Name	MirrorSourceNetworkCanFD		
Parent Container	MirrorSourceNetwork		
Description	Source bus representing a CAN FD network.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorNetworkId	1	[ECUC_Mirror_00012]
MirrorSourceMaxDynamicFilters	1	[ECUC_Mirror_00013]
MirrorComMNetworkHandleRef	1	[ECUC_Mirror_00064]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorSourceCanFilter	0..255	Pre-configured filter for CAN frames.
MirrorSourceCanMaskBasedId Mapping	0..*	Rule for remapping a set of CAN IDs.
MirrorSourceCanSingleIdMapping	0..*	Rule for remapping a single CAN ID.

⌋

For parameter table [ECUC_Mirror_00012] MirrorNetworkId, see definition below container MirrorDestNetworkCan.

For parameter table [ECUC_Mirror_00013] MirrorSourceMaxDynamicFilters, see definition below container MirrorSourceNetworkCan.

For parameter table [ECUC_Mirror_00064] MirrorComMNetworkHandleRef, see definition below container MirrorDestNetworkCan.

10.1.8 MirrorSourceCanFilter

[ECUC_Mirror_00014] Definition of EcucChoiceContainerDef MirrorSourceCanFilter ⌈

Choice Container Name	MirrorSourceCanFilter		
Parent Container	MirrorSourceNetworkCan, MirrorSourceNetworkCanFD		
Description	Pre-configured filter for CAN frames.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

No Included Parameters

Container Choices		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorSourceCanFilterMask	0..1	Pre-configured mask based filter for CAN frames.
MirrorSourceCanFilterRange	0..1	Pre-configured range filter for CAN frames.

⌋

10.1.9 MirrorSourceCanFilterRange

[ECUC_Mirror_00015] Definition of EcucParamConfContainerDef MirrorSourceCanFilterRange ⌈

Container Name	MirrorSourceCanFilterRange		
Parent Container	MirrorSourceCanFilter		
Description	Pre-configured range filter for CAN frames.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorSourceCanFilterId	1	[ECUC_Mirror_00018]
MirrorSourceCanFilterLower	1	[ECUC_Mirror_00016]
MirrorSourceCanFilterUpper	1	[ECUC_Mirror_00017]

No Included Containers

⌋

For parameter table [ECUC_Mirror_00018] MirrorSourceCanFilterId, see definition below container MirrorSourceCanFilterMask.

[ECUC_Mirror_00016] Definition of EcucIntegerParamDef MirrorSourceCanFilterLower ⌈

Parameter Name	MirrorSourceCanFilterLower		
Parent Container	MirrorSourceCanFilterRange		
Description	Lowest CAN ID that is accepted by the filter.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00017] Definition of EcucIntegerParamDef MirrorSourceCanFilterUpper ⌈

Parameter Name	MirrorSourceCanFilterUpper		
Parent Container	MirrorSourceCanFilterRange		
Description	Highest CAN ID that is accepted by the filter.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

▽

△

Scope / Dependency	scope: local

⌋

10.1.10 MirrorSourceCanFilterMask

[ECUC_Mirror_00019] Definition of EcucParamConfContainerDef MirrorSourceCanFilterMask ⌈

Container Name	MirrorSourceCanFilterMask		
Parent Container	MirrorSourceCanFilter		
Description	Pre-configured mask based filter for CAN frames.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorSourceCanFilterCanIdCode	1	[ECUC_Mirror_00020]
MirrorSourceCanFilterCanIdMask	1	[ECUC_Mirror_00021]
MirrorSourceCanFilterId	1	[ECUC_Mirror_00018]

No Included Containers

⌋

[ECUC_Mirror_00020] Definition of EcucIntegerParamDef MirrorSourceCanFilterCanIdCode ⌈

Parameter Name	MirrorSourceCanFilterCanIdCode		
Parent Container	MirrorSourceCanFilterMask		
Description	Value to match masked CAN IDs.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

▽

△

Scope / Dependency	scope: local

⌋

[ECUC_Mirror_00021] Definition of EcucIntegerParamDef MirrorSourceCanFilterCanIdMask ⌈

Parameter Name	MirrorSourceCanFilterCanIdMask		
Parent Container	MirrorSourceCanFilterMask		
Description	Mask applied to CAN IDs before comparison.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00018] Definition of EcucIntegerParamDef MirrorSourceCanFilterId ⌈

Parameter Name	MirrorSourceCanFilterId		
Parent Container	MirrorSourceCanFilterMask, MirrorSourceCanFilterRange		
Description	Unique identifier of the pre-configured CAN filter.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

⌋

10.1.11 MirrorSourceCanSingleIdMapping

[ECUC_Mirror_00022] Definition of EcucParamConfContainerDef MirrorSourceCanSingleIdMapping ⌈

Container Name	MirrorSourceCanSingleIdMapping		
Parent Container	MirrorSourceNetworkCan, MirrorSourceNetworkCanFD		
Description	Rule for remapping a single CAN ID.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
MirrorSourceCanSingleIdMappingDestCanId	1	[ECUC_Mirror_00024]
MirrorSourceCanSingleIdMappingSourceCanId	1	[ECUC_Mirror_00023]

No Included Containers

⌋

[ECUC_Mirror_00024] Definition of EcucIntegerParamDef MirrorSourceCanSingleIdMappingDestCanId ⌈

Parameter Name	MirrorSourceCanSingleIdMappingDestCanId		
Parent Container	MirrorSourceCanSingleIdMapping		
Description	Mapped CAN ID.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00023] Definition of EcucIntegerParamDef MirrorSourceCanSingleIdMappingSourceCanId ⌈

Parameter Name	MirrorSourceCanSingleIdMappingSourceCanId	
Parent Container	MirrorSourceCanSingleIdMapping	
Description	Original CAN ID.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	

▽

△

Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.1.12 MirrorSourceCanMaskBasedIdMapping

[ECUC_Mirror_00025] Definition of EcucParamConfContainerDef MirrorSourceCanMaskBasedIdMapping ⌈

Container Name	MirrorSourceCanMaskBasedIdMapping		
Parent Container	MirrorSourceNetworkCan, MirrorSourceNetworkCanFD		
Description	Rule for remapping a set of CAN IDs.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorSourceCanMaskBasedIdMappingDestBaseId	1	[ECUC_Mirror_00028]
MirrorSourceCanMaskBasedIdMappingSourceCanIdCode	1	[ECUC_Mirror_00026]
MirrorSourceCanMaskBasedIdMappingSourceCanIdMask	1	[ECUC_Mirror_00027]

No Included Containers

⌋

[ECUC_Mirror_00028] Definition of EcucIntegerParamDef MirrorSourceCanMaskBasedIdMappingDestBaseId ⌈

Parameter Name	MirrorSourceCanMaskBasedIdMappingDestBaseId
Parent Container	MirrorSourceCanMaskBasedIdMapping
Description	Base ID merged with the masked parts of the original CAN ID to form the mapped CAN ID.
Multiplicity	1
Type	EcucIntegerParamDef

▽

△

Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00026] Definition of EcucIntegerParamDef MirrorSourceCanMaskBasedIdMappingSourceCanIdCode ⌈

Parameter Name	MirrorSourceCanMaskBasedIdMappingSourceCanIdCode		
Parent Container	MirrorSourceCanMaskBasedIdMapping		
Description	Value to match masked original CAN IDs.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00027] Definition of EcucIntegerParamDef MirrorSourceCanMaskBasedIdMappingSourceCanIdMask ⌈

Parameter Name	MirrorSourceCanMaskBasedIdMappingSourceCanIdMask		
Parent Container	MirrorSourceCanMaskBasedIdMapping		
Description	Mask applied to original CAN IDs before comparison.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.1.13 MirrorSourceNetworkLin

[ECUC_Mirror_00029] Definition of EcucParamConfContainerDef MirrorSourceNetworkLin ⌈

Container Name	MirrorSourceNetworkLin		
Parent Container	MirrorSourceNetwork		
Description	Source bus representing a LIN network.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorNetworkId	1	[ECUC_Mirror_00012]
MirrorSourceLinToCanBaseId	1	[ECUC_Mirror_00041]
MirrorSourceMaxDynamicFilters	1	[ECUC_Mirror_00013]
MirrorComMNetworkHandleRef	1	[ECUC_Mirror_00064]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorSourceLinFilter	0..255	Pre-configured filter for LIN frames.
MirrorSourceLinToCanIdMapping	0..*	Rule for mapping a LIN frame ID to a special CAN ID.

⌋

For parameter table [ECUC_Mirror_00012] MirrorNetworkId, see definition below container MirrorDestNetworkCan.

[ECUC_Mirror_00041] Definition of EcucIntegerParamDef MirrorSourceLinToCanBaseId ⌈

Parameter Name	MirrorSourceLinToCanBaseId		
Parent Container	MirrorSourceNetworkLin		
Description	Base ID merged with the LIN frame ID to form the CAN ID.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

▽

△

Scope / Dependency	scope: local

⌋

For parameter table [ECUC_Mirror_00013] MirrorSourceMaxDynamicFilters, see definition below container MirrorSourceNetworkCan.

For parameter table [ECUC_Mirror_00064] MirrorComMNetworkHandleRef, see definition below container MirrorDestNetworkCan.

10.1.14 MirrorSourceLinFilter

[ECUC_Mirror_00030] Definition of EcucChoiceContainerDef MirrorSourceLinFilter ⌈

Choice Container Name	MirrorSourceLinFilter		
Parent Container	MirrorSourceNetworkLin		
Description	Pre-configured filter for LIN frames.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

No Included Parameters

Container Choices		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorSourceLinFilterMask	0..1	Pre-configured mask based filter for LIN frames.
MirrorSourceLinFilterRange	0..1	Pre-configured range filter for LIN frames.

⌋

10.1.15 MirrorSourceLinFilterRange

[ECUC_Mirror_00031] Definition of EcucParamConfContainerDef MirrorSourceLinFilterRange ⌈

Container Name	MirrorSourceLinFilterRange
Parent Container	MirrorSourceLinFilter
Description	Pre-configured range filter for LIN frames.

▽

△

Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorSourceLinFilterId	1	[ECUC_Mirror_00034]
MirrorSourceLinFilterLower	1	[ECUC_Mirror_00032]
MirrorSourceLinFilterUpper	1	[ECUC_Mirror_00033]

No Included Containers

⌋

For parameter table [ECUC_Mirror_00034] MirrorSourceLinFilterId, see definition below container MirrorSourceLinFilterMask.

[ECUC_Mirror_00032] Definition of EcucIntegerParamDef MirrorSourceLinFilterLower ⌈

Parameter Name	MirrorSourceLinFilterLower		
Parent Container	MirrorSourceLinFilterRange		
Description	Lowest frame ID that is accepted by the filter.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 63		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00033] Definition of EcucIntegerParamDef MirrorSourceLinFilterUpper ⌈

Parameter Name	MirrorSourceLinFilterUpper
Parent Container	MirrorSourceLinFilterRange
Description	Highest frame ID that is accepted by the filter.
Multiplicity	1

▽

△

Type	EcucIntegerParamDef		
Range	0 .. 63		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.1.16 MirrorSourceLinFilterMask

[ECUC_Mirror_00035] Definition of EcucParamConfContainerDef MirrorSourceLinFilterMask ⌈

Container Name	MirrorSourceLinFilterMask		
Parent Container	MirrorSourceLinFilter		
Description	Pre-configured mask based filter for LIN frames.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorSourceLinFilterId	1	[ECUC_Mirror_00034]
MirrorSourceLinFilterLinIdCode	1	[ECUC_Mirror_00036]
MirrorSourceLinFilterLinIdMask	1	[ECUC_Mirror_00037]

No Included Containers

⌋

[ECUC_Mirror_00034] Definition of EcucIntegerParamDef MirrorSourceLinFilterId ⌈

Parameter Name	MirrorSourceLinFilterId
Parent Container	MirrorSourceLinFilterMask, MirrorSourceLinFilterRange
Description	Unique identifier of the pre-configured LIN filter.
Multiplicity	1

▽

△

Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

⌋

[ECUC_Mirror_00036] Definition of EcucIntegerParamDef MirrorSourceLinFilterLinIdCode ⌈

Parameter Name	MirrorSourceLinFilterLinIdCode		
Parent Container	MirrorSourceLinFilterMask		
Description	Value to match masked frame IDs.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 63		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00037] Definition of EcucIntegerParamDef MirrorSourceLinFilterLinIdMask ⌈

Parameter Name	MirrorSourceLinFilterLinIdMask		
Parent Container	MirrorSourceLinFilterMask		
Description	Mask applied to frame IDs before comparison.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 63		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.1.17 MirrorSourceLinToCanIdMapping

[ECUC_Mirror_00038] Definition of EcucParamConfContainerDef MirrorSourceLinToCanIdMapping ⌈

Container Name	MirrorSourceLinToCanIdMapping		
Parent Container	MirrorSourceNetworkLin		
Description	Rule for mapping a LIN frame ID to a special CAN ID.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorSourceLinToCanIdMappingCanId	1	[ECUC_Mirror_00040]
MirrorSourceLinToCanIdMappingLinId	1	[ECUC_Mirror_00039]

No Included Containers

⌋

[ECUC_Mirror_00040] Definition of EcucIntegerParamDef MirrorSourceLinToCanIdMappingCanId ⌈

Parameter Name	MirrorSourceLinToCanIdMappingCanId		
Parent Container	MirrorSourceLinToCanIdMapping		
Description	CAN ID which lies outside of the range mapping.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00039] Definition of EcucIntegerParamDef MirrorSourceLinToCanIdMappingLinId ⌈

Parameter Name	MirrorSourceLinToCanIdMappingLinId		
Parent Container	MirrorSourceLinToCanIdMapping		
Description	Frame ID which is excluded from the range mapping.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 63		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.1.18 MirrorSourceNetworkFlexRay

[ECUC_Mirror_00042] Definition of EcucParamConfContainerDef MirrorSourceNetworkFlexRay ⌈

Container Name	MirrorSourceNetworkFlexRay		
Parent Container	MirrorSourceNetwork		
Description	Source bus representing a FlexRay network.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorNetworkId	1	[ECUC_Mirror_00012]
MirrorSourceMaxDynamicFilters	1	[ECUC_Mirror_00013]
MirrorComMNetworkHandleRef	1	[ECUC_Mirror_00064]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorSourceFlexRayFilter	0..255	Pre-configured filter for FlexRay frames.

⌋

For parameter table [ECUC_Mirror_00012] MirrorNetworkId, see definition below container MirrorDestNetworkCan.

For parameter table [ECUC_Mirror_00013] MirrorSourceMaxDynamicFilters, see definition below container MirrorSourceNetworkCan.

For parameter table [ECUC_Mirror_00064] MirrorComMNetworkHandleRef, see definition below container MirrorDestNetworkCan.

10.1.19 MirrorSourceFlexRayFilter

[ECUC_Mirror_00043] Definition of EcucParamConfContainerDef MirrorSourceFlexRayFilter ⌈

Container Name	MirrorSourceFlexRayFilter		
Parent Container	MirrorSourceNetworkFlexRay		
Description	Pre-configured filter for FlexRay frames.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorSourceFlexRayFilterChannelAssignment	1	[ECUC_Mirror_00049]
MirrorSourceFlexRayFilterCycleRepetition	1	[ECUC_Mirror_00048]
MirrorSourceFlexRayFilterId	1	[ECUC_Mirror_00050]
MirrorSourceFlexRayFilterLowerBaseCycle	1	[ECUC_Mirror_00046]
MirrorSourceFlexRayFilterLowerSlot	1	[ECUC_Mirror_00044]
MirrorSourceFlexRayFilterUpperBaseCycle	1	[ECUC_Mirror_00047]
MirrorSourceFlexRayFilterUpperSlot	1	[ECUC_Mirror_00045]

No Included Containers

⌋

[ECUC_Mirror_00049] Definition of EcucEnumerationParamDef MirrorSourceFlexRayFilterChannelAssignment ⌈

Parameter Name	MirrorSourceFlexRayFilterChannelAssignment	
Parent Container	MirrorSourceFlexRayFilter	
Description	FlexRay channels accepted by the filter.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	MIRROR_FR_CHANNEL_A	FlexRay channel A only.
	MIRROR_FR_CHANNEL_AB	FlexRay channel A and B.

▽

△

	MIRROR_FR_CHANNEL_B	FlexRay channel B only.	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00048] Definition of EcucIntegerParamDef MirrorSourceFlexRay FilterCycleRepetition ⌈

Parameter Name	MirrorSourceFlexRayFilterCycleRepetition		
Parent Container	MirrorSourceFlexRayFilter		
Description	Cycle repetition of accepted cycles.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 64		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00050] Definition of EcucIntegerParamDef MirrorSourceFlexRay FilterId ⌈

Parameter Name	MirrorSourceFlexRayFilterId		
Parent Container	MirrorSourceFlexRayFilter		
Description	Unique identifier of the pre-configured FlexRay filter.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

⌋

[ECUC_Mirror_00046] Definition of EcucIntegerParamDef MirrorSourceFlexRayFilterLowerBaseCycle ⌈

Parameter Name	MirrorSourceFlexRayFilterLowerBaseCycle		
Parent Container	MirrorSourceFlexRayFilter		
Description	Lowest base cycle number that is accepted by the filter.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 63		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00044] Definition of EcucIntegerParamDef MirrorSourceFlexRayFilterLowerSlot ⌈

Parameter Name	MirrorSourceFlexRayFilterLowerSlot		
Parent Container	MirrorSourceFlexRayFilter		
Description	Lowest slot ID that is accepted by the filter.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 2047		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00047] Definition of EcucIntegerParamDef MirrorSourceFlexRayFilterUpperBaseCycle ⌈

Parameter Name	MirrorSourceFlexRayFilterUpperBaseCycle		
Parent Container	MirrorSourceFlexRayFilter		
Description	Highest base cycle number that is accepted by the filter.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 63		
Default value	–		

▽

△

Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00045] Definition of EcucIntegerParamDef MirrorSourceFlexRay FilterUpperSlot ⌈

Parameter Name	MirrorSourceFlexRayFilterUpperSlot		
Parent Container	MirrorSourceFlexRayFilter		
Description	Highest slot ID that is accepted by the filter.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 2047		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.1.20 MirrorDestNetwork

[ECUC_Mirror_00051] Definition of EcucChoiceContainerDef MirrorDestNetwork ⌈

Choice Container Name	MirrorDestNetwork		
Parent Container	MirrorConfigSet		
Description	Destination bus to which frames are sent by the Bus Mirroring module.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

No Included Parameters

Container Choices		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorDestNetworkCan	0..1	Destination bus representing a CAN network.
MirrorDestNetworkCanFD	0..1	Destination bus representing a CAN FD network. Mirroring can be configured to either direct (MirrorDestProtocolType == MIRROR_PT_NONE) or serialized (MirrorDestProtocolType != MIRROR_PT_NONE).
MirrorDestNetworkCanXL	0..1	Destination bus representing a CAN XL network.
MirrorDestNetworkCdd	0..1	Destination bus representing a user defined network.
MirrorDestNetworkFlexRay	0..1	Destination bus representing a FlexRay network.
MirrorDestNetworkIp	0..1	Destination bus representing an IP network.

⌋

10.1.21 MirrorDestNetworkCan

[ECUC_Mirror_00052] Definition of EcucParamConfContainerDef MirrorDestNetworkCan ⌈

Container Name	MirrorDestNetworkCan		
Parent Container	MirrorDestNetwork		
Description	Destination bus representing a CAN network.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorDestQueueSize	1	[ECUC_Mirror_00054]
MirrorNetworkId	1	[ECUC_Mirror_00012]
MirrorStatusCanId	0..1	[ECUC_Mirror_00061]
MirrorComMNetworkHandleRef	1	[ECUC_Mirror_00064]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorDestPdu	1	I-PDU used for transmission of the mirrored frames on the destination bus.

⌋

[ECUC_Mirror_00054] Definition of EcucIntegerParamDef MirrorDestQueueSize ⌈

Parameter Name	MirrorDestQueueSize		
Parent Container	MirrorDestNetworkCan, MirrorDestNetworkCanFD, MirrorDestNetworkCanXL, MirrorDestNetworkCdd, MirrorDestNetworkFlexRay, MirrorDestNetworkIp		
Description	Number of frames that can be stored in the output queue for the destination bus.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default value	20		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00012] Definition of EcucIntegerParamDef MirrorNetworkId ⌈

Parameter Name	MirrorNetworkId		
Parent Container	MirrorDestNetworkCan, MirrorDestNetworkCanFD, MirrorDestNetworkCanXL, MirrorDestNetworkCdd, MirrorDestNetworkFlexRay, MirrorDestNetworkIp, MirrorSourceNetworkCan, MirrorSourceNetworkCanFD, MirrorSourceNetworkFlexRay, MirrorSourceNetworkLin		
Description	Network ID of the bus.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

⌋

[ECUC_Mirror_00061] Definition of EcucIntegerParamDef MirrorStatusCanId ⌈

Parameter Name	MirrorStatusCanId
Parent Container	MirrorDestNetworkCan, MirrorDestNetworkCanFD
Description	CAN ID of the CAN status frame. If configured, a status frame will be sent on the CAN destination bus that contains the state of all active source buses.
Multiplicity	0..1
Type	EcucIntegerParamDef

▽

△

Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00064] Definition of EcucReferenceDef MirrorComMNetworkHandleRef ⌈

Parameter Name	MirrorComMNetworkHandleRef		
Parent Container	MirrorDestNetworkCan, MirrorDestNetworkCanFD, MirrorDestNetworkCanXL, MirrorDestNetworkCdd, MirrorDestNetworkFlexRay, MirrorDestNetworkIp, MirrorSourceNetworkCan, MirrorSourceNetworkCanFD, MirrorSourceNetworkFlexRay, MirrorSourceNetworkLin		
Description	Reference to the ComMChannel that represents the bus.		
Multiplicity	1		
Type	Symbolic name reference to ComMChannel		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.1.22 MirrorDestPdu

[ECUC_Mirror_00055] Definition of EcucParamConfContainerDef MirrorDestPdu ⌈

Container Name	MirrorDestPdu		
Parent Container	MirrorDestNetworkCan, MirrorDestNetworkCanFD, MirrorDestNetworkCanXL, MirrorDestNetworkCdd, MirrorDestNetworkIp		
Description	I-PDU used for transmission of the mirrored frames on the destination bus.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorDestPduId	1	[ECUC_Mirror_00057]
MirrorDestPduUsesTriggerTransmit	1	[ECUC_Mirror_00063]
MirrorDestPduRef	1	[ECUC_Mirror_00056]

No Included Containers

⌋

[ECUC_Mirror_00057] Definition of EcucIntegerParamDef MirrorDestPduId ⌈

Parameter Name	MirrorDestPduId		
Parent Container	MirrorDestPdu, MirrorDestPduFlexRay		
Description	I-PDU identifier used for TxConfirmation from PduR.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU withAuto = true		

⌋

[ECUC_Mirror_00063] Definition of EcucBooleanParamDef MirrorDestPduUses TriggerTransmit ⌈

Parameter Name	MirrorDestPduUsesTriggerTransmit		
Parent Container	MirrorDestPdu, MirrorDestPduFlexRay		
Description	Switches transmission via TriggerTransmit. • true: The I-PDU is transmitted using TriggerTransmit. • false: The I-PDU is transmitted directly with the Transmit call.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mirror_00056] Definition of EcucReferenceDef MirrorDestPduRef ⌈

Parameter Name	MirrorDestPduRef		
Parent Container	MirrorDestPdu, MirrorDestPduFlexRay		
Description	Reference to the Pdu object representing the I-PDU.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.1.23 MirrorDestNetworkCanFD

[ECUC_Mirror_00073] Definition of EcucParamConfContainerDef MirrorDestNetworkCanFD ⌈

Container Name	MirrorDestNetworkCanFD		
Parent Container	MirrorDestNetwork		
Description	Destination bus representing a CAN FD network. Mirroring can be configured to either direct (MirrorDestProtocolType == MIRROR_PT_NONE) or serialized (MirrorDest ProtocolType != MIRROR_PT_NONE).		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorDestProtocolType	1	[ECUC_Mirror_00074]
MirrorDestQueueSize	1	[ECUC_Mirror_00054]
MirrorDestTransmissionDeadline	0..1	[ECUC_Mirror_00059]
MirrorNetworkId	1	[ECUC_Mirror_00012]
MirrorStatusCanId	0..1	[ECUC_Mirror_00061]
MirrorComMNetworkHandleRef	1	[ECUC_Mirror_00064]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorDestPdu	1	I-PDU used for transmission of the mirrored frames on the destination bus.

⌋

[ECUC_Mirror_00074] Definition of EcucEnumerationParamDef MirrorDestProtocolType ⌈

Parameter Name	MirrorDestProtocolType		
Parent Container	MirrorDestNetworkCanFD		
Description	Protocol type to use for the transmission on the destination bus. If set to MIRROR_PT_NONE, source frames are mirrored as single frames to the destination. Otherwise, they are serialized according to the chosen protocol		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	MIRROR_PT_NONE	Source Frames are mirrored as single frames to the destination.	
	MIRROR_PT_VERSION_1	Source frames are serialized and collected into destination frames.	
Default value	MIRROR_PT_NONE		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

⌋

For parameter table [ECUC_Mirror_00054] MirrorDestQueueSize, see definition below container MirrorDestNetworkCan.

[ECUC_Mirror_00059] Definition of EcucFloatParamDef MirrorDestTransmissionDeadline ⌈

Parameter Name	MirrorDestTransmissionDeadline		
Parent Container	MirrorDestNetworkCanFD, MirrorDestNetworkCanXL, MirrorDestNetworkCdd, MirrorDestNetworkFlexRay, MirrorDestNetworkIp		
Description	Time in seconds after which the collection of source frames into the destination frame stopped and the frame is sent at the latest. If omitted, destination frames are only sent when full or when the time stamp overflows after 655.35ms.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0.001 .. 0.655]		
Default value	0.1		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

▽

△

Scope / Dependency	scope: local dependency: For MirrorDestNetworkCanFD, this parameter is only relevant when (MirrorDestProtocolType != MIRROR_PT_NONE).

⌋

For parameter table [ECUC_Mirror_00012] MirrorNetworkId, see definition below container MirrorDestNetworkCan.

For parameter table [ECUC_Mirror_00061] MirrorStatusCanId, see definition below container MirrorDestNetworkCan.

For parameter table [ECUC_Mirror_00064] MirrorComMNetworkHandleRef, see definition below container MirrorDestNetworkCan.

10.1.24 MirrorDestNetworkCanXL

[ECUC_Mirror_00071] Definition of EcucParamConfContainerDef MirrorDestNetworkCanXL ⌈

Container Name	MirrorDestNetworkCanXL		
Parent Container	MirrorDestNetwork		
Description	Destination bus representing a CAN XL network.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorDestQueueSize	1	[ECUC_Mirror_00054]
MirrorDestTransmissionDeadline	0..1	[ECUC_Mirror_00059]
MirrorNetworkId	1	[ECUC_Mirror_00012]
MirrorComMNetworkHandleRef	1	[ECUC_Mirror_00064]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorDestPdu	1	I-PDU used for transmission of the mirrored frames on the destination bus.

⌋

For parameter table [ECUC_Mirror_00054] MirrorDestQueueSize, see definition below container MirrorDestNetworkCan.

For parameter table [ECUC_Mirror_00059] MirrorDestTransmissionDeadline, see definition below container MirrorDestNetworkCanFD.

For parameter table [ECUC_Mirror_00012] MirrorNetworkId, see definition below container MirrorDestNetworkCan.

For parameter table [ECUC_Mirror_00064] MirrorComMNetworkHandleRef, see definition below container MirrorDestNetworkCan.

10.1.25 MirrorDestNetworkFlexRay

[ECUC_Mirror_00058] Definition of EcucParamConfContainerDef MirrorDestNetworkFlexRay ⌈

Container Name	MirrorDestNetworkFlexRay		
Parent Container	MirrorDestNetwork		
Description	Destination bus representing a FlexRay network.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorDestQueueSize	1	[ECUC_Mirror_00054]
MirrorDestTransmissionDeadline	0..1	[ECUC_Mirror_00059]
MirrorNetworkId	1	[ECUC_Mirror_00012]
MirrorComMNetworkHandleRef	1	[ECUC_Mirror_00064]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorDestPduFlexRay	1..*	I-PDU used for transmission of the mirrored frames on the destination bus. For FlexRay, an arbitrary number of I-PDUs can be configured.

⌋

For parameter table [ECUC_Mirror_00054] MirrorDestQueueSize, see definition below container MirrorDestNetworkCan.

For parameter table [ECUC_Mirror_00059] MirrorDestTransmissionDeadline, see definition below container MirrorDestNetworkCanFD.

For parameter table [ECUC_Mirror_00012] MirrorNetworkId, see definition below container MirrorDestNetworkCan.

For parameter table [ECUC_Mirror_00064] MirrorComMNetworkHandleRef, see definition below container MirrorDestNetworkCan.

10.1.26 MirrorDestPduFlexRay

[ECUC_Mirror_00066] Definition of EcucParamConfContainerDef MirrorDestPduFlexRay ⌈

Container Name	MirrorDestPduFlexRay		
Parent Container	MirrorDestNetworkFlexRay		
Description	I-PDU used for transmission of the mirrored frames on the destination bus. For Flex Ray, an arbitrary number of I-PDUs can be configured.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorDestPduId	1	[ECUC_Mirror_00057]
MirrorDestPduUsesTriggerTransmit	1	[ECUC_Mirror_00063]
MirrorDestPduRef	1	[ECUC_Mirror_00056]

No Included Containers

⌋

For parameter table [ECUC_Mirror_00057] MirrorDestPduId, see definition below container MirrorDestPdu.

For parameter table [ECUC_Mirror_00063] MirrorDestPduUsesTriggerTransmit, see definition below container MirrorDestPdu.

For parameter table [ECUC_Mirror_00056] MirrorDestPduRef, see definition below container MirrorDestPdu.

10.1.27 MirrorDestNetworkIp

[ECUC_Mirror_00060] Definition of EcucParamConfContainerDef MirrorDestNetworkIp ⌈

Container Name	MirrorDestNetworkIp		
Parent Container	MirrorDestNetwork		
Description	Destination bus representing an IP network.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
MirrorDestQueueSize	1	[ECUC_Mirror_00054]
MirrorDestTransmissionDeadline	0..1	[ECUC_Mirror_00059]
MirrorNetworkId	1	[ECUC_Mirror_00012]
MirrorComMNetworkHandleRef	1	[ECUC_Mirror_00064]

Included Containers		
Container Name	Multiplicity	Scope / Dependency
MirrorDestPdu	1	I-PDU used for transmission of the mirrored frames on the destination bus.

⌋

For parameter table [ECUC_Mirror_00054] MirrorDestQueueSize, see definition below container MirrorDestNetworkCan.

For parameter table [ECUC_Mirror_00059] MirrorDestTransmissionDeadline, see definition below container MirrorDestNetworkCanFD.

For parameter table [ECUC_Mirror_00012] MirrorNetworkId, see definition below container MirrorDestNetworkCan.

For parameter table [ECUC_Mirror_00064] MirrorComMNetworkHandleRef, see definition below container MirrorDestNetworkCan.

10.1.28 MirrorDestNetworkCdd

[ECUC_Mirror_00062] Definition of EcucParamConfContainerDef MirrorDestNetworkCdd ⌈

Container Name	MirrorDestNetworkCdd
Parent Container	MirrorDestNetwork
Description	Destination bus representing a user defined network.
Post-Build Variant Multiplicity	true

▽

△

Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MirrorDestQueueSize	1	[ECUC_Mirror_00054]
MirrorDestTransmissionDeadline	0..1	[ECUC_Mirror_00059]
MirrorNetworkId	1	[ECUC_Mirror_00012]
MirrorComMNetworkHandleRef	1	[ECUC_Mirror_00064]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MirrorDestPdu	1	I-PDU used for transmission of the mirrored frames on the destination bus.

⌋

For parameter table [ECUC_Mirror_00054] MirrorDestQueueSize, see definition below container MirrorDestNetworkCan.

For parameter table [ECUC_Mirror_00059] MirrorDestTransmissionDeadline, see definition below container MirrorDestNetworkCanFD.

For parameter table [ECUC_Mirror_00012] MirrorNetworkId, see definition below container MirrorDestNetworkCan.

For parameter table [ECUC_Mirror_00064] MirrorComMNetworkHandleRef, see definition below container MirrorDestNetworkCan.

10.2 Configuration Constraints

This section lists configuration constraints for the the `MirrorDestPdu`s of the supported destination buses.

10.2.1 CAN / CAN FD Destination Bus

[SWS_Mirror_CONSTR_00001] ⌈The `MirrorDestPdu` of a `MirrorDestNetworkCan` or a direct `MirrorDestNetworkCanFD` (`MirrorDestProtocolType` == `MIRROR_PT_NONE`) requires a `MetaDataItem` of `MetaDataItemType` `CAN_ID_32`. The `CanIfTxPduCanIdMask` of the corresponding `CanIfTxPduCfg` shall be `0`.⌋

This way, the Bus Mirroring module can transmit CAN (FD) destination frames with any CAN ID.

[SWS_Mirror_CONSTR_00002] ⌈The `CanFdPaddingValue` that is used to transmit the PDU referenced by `MirrorDestPduRef` for a CAN FD destination bus shall be set to 0 to ensure that the `NetworkStateAvailable` of a CAN status item is 0 if the status item has not been written by the Bus Mirroring module but lies in a padded region of the status frame.⌋

10.2.2 FlexRay Destination Bus

To avoid padding, the `MirrorDestPdu` used for a FlexRay destination bus shall be placed on dynamic frames.

[SWS_Mirror_CONSTR_00004] ⌈`FrIfAllowDynamicLSduLength` shall be set to true for all `FrIfFrameStructure`s that contain `FrIfTxPdu`s referenced by a `MirrorDestPdu` of a `MirrorDestNetworkFlexRay`.⌋

According to [SWS_FrIf_05244], a FlexRay PDU with dynamic length must be placed at the end of a FlexRay frame, or must be the only PDU within the frame.

10.2.3 Mirroring of Serialized Frames

In principal, when a serialized frame is received by an ECU that features Bus Mirroring, it would be nice to merge it into the stream of serialized messages created by the Bus Mirroring module. But as declared Section 4.1, this would mean that the Bus Mirroring module would have to first de-serialize the received message and then re-serialize the elements of the message, which would be quite complicated and expensive regarding run-time, and it would require an extended configuration because the mirroring could not discern serialized frames from other frames that accidentally could be interpreted as serialized frames.

Note that this scenario can only happen on a FlexRay source bus, because IP/Ethernet and proprietary networks cannot be configured as source buses.

If a `MirrorSourceFlexRayFilter` accepts the serialized frames, they will therefore be packed as a single frame into the serialized destination frame, resulting in a nested serialization. To avoid such a nested serialization, it should be avoided that serialized frames are accepted by the Bus Mirroring module by setting the FlexRay frame filters accordingly.

[SWS_Mirror_CONSTR_00003] ⌈The configured MirrorSourceFlexRayFilters shall be configured such that they do not include serialized frames transmitted on the source bus.⌋

Instead, a direct routing of the serialized frame should be configured using PduR, resulting in additional PDUs which could carry serialized frames on the destination bus.

10.3 Published Information

For details, refer to the Section 10.3 “Published Information” in [2, SWS BSW General].

A Not Applicable Requirements

[SWS_Mirror_NA]

Upstream requirements: SRS_Mirror_00002, SRS_Mirror_00003, SRS_Mirror_00004, SRS_Mirror_00014, SRS_Mirror_00016, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00375, SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00394, SRS_BSW_00395, SRS_BSW_00396, SRS_BSW_00399, SRS_BSW_00403, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00419, SRS_BSW_00422, SRS_BSW_00425, SRS_BSW_00432, SRS_BSW_00458, SRS_BSW_00466, SRS_BSW_00469, SRS_BSW_00470, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00490, SRS_BSW_00492

⌈These requirements are not applicable to this specification.⌋

B Change History of AUTOSAR Traceable Items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

B.1 Traceable Item History of this Document According to AUTOSAR Release R24-11

B.1.1 Added Specification Items in R24-11

[ECUC_Mirror_00072] [ECUC_Mirror_00073] [ECUC_Mirror_00074]

B.1.2 Changed Specification Items in R24-11

[ECUC_Mirror_00009] [ECUC_Mirror_00012] [ECUC_Mirror_00013] [ECUC_Mirror_00014] [ECUC_Mirror_00022] [ECUC_Mirror_00025] [ECUC_Mirror_00051] [ECUC_Mirror_00054] [ECUC_Mirror_00055] [ECUC_Mirror_00059] [ECUC_Mirror_00061] [ECUC_Mirror_00064] [SWS_Mirror_00043] [SWS_Mirror_00052] [SWS_Mirror_00147] [SWS_Mirror_00170]

B.1.3 Deleted Specification Items in R24-11

none

B.1.4 Added Constraints in R24-11

none

B.1.5 Changed Constraints in R24-11

[SWS_Mirror_CONSTR_00001]

B.1.6 Deleted Constraints in R24-11

none

B.2 Traceable Item History of this Document According to AUTOSAR Release R23-11

B.2.1 Added Specification Items in R23-11

[SWS_Mirror_00170]

B.2.2 Changed Specification Items in R23-11

[SWS_Mirror_00061] [SWS_Mirror_00074] [SWS_Mirror_01000]

B.2.3 Deleted Specification Items in R23-11

none

B.2.4 Added Constraints in R23-11

none

B.2.5 Changed Constraints in R23-11

none

B.2.6 Deleted Constraints in R23-11

none

B.3 Traceable Item History of this Document According to AUTOSAR Release R22-11

B.3.1 Added Specification Items in R22-11

[SWS_Mirror_NA]

B.3.2 Changed Specification Items in R22-11

[SWS_Mirror_00043] [SWS_Mirror_00052] [SWS_Mirror_00147] [SWS_Mirror_-01000] [SWS_Mirror_01033] [SWS_Mirror_01100]

B.3.3 Deleted Specification Items in R22-11

none

B.4 Traceable Item History of this Document According to AUTOSAR Release R21-11

B.4.1 Added Specification Items in R21-11

none

B.4.2 Changed Specification Items in R21-11

none

B.4.3 Deleted Specification Items in R21-11

none

B.5 Traceable Item History of this Document According to AUTOSAR Release R20-11

B.5.1 Added Specification Items in R20-11

none

B.5.2 Changed Specification Items in R20-11

[SWS_Mirror_00022] [SWS_Mirror_00030] [SWS_Mirror_00114] [SWS_Mirror_-00115] [SWS_Mirror_00116] [SWS_Mirror_00118]

B.5.3 Deleted Specification Items in R20-11

none

B.6 Traceable Item History of this Document According to AUTOSAR Release R19-11

B.6.1 Added Specification Items in R19-11

[SWS_Mirror_00166] [SWS_Mirror_00167] [SWS_Mirror_00168] [SWS_Mirror_00169]

B.6.2 Changed Specification Items in R19-11

[SWS_Mirror_00047] [SWS_Mirror_00097] [SWS_Mirror_00098] [SWS_Mirror_00099] [SWS_Mirror_00100] [SWS_Mirror_00101] [SWS_Mirror_00102] [SWS_Mirror_00103] [SWS_Mirror_00104] [SWS_Mirror_00105] [SWS_Mirror_00106] [SWS_Mirror_00107] [SWS_Mirror_00108] [SWS_Mirror_00124] [SWS_Mirror_00127] [SWS_Mirror_00128] [SWS_Mirror_00129] [SWS_Mirror_00131] [SWS_Mirror_00133] [SWS_Mirror_00134] [SWS_Mirror_00135] [SWS_Mirror_00136] [SWS_Mirror_00149] [SWS_Mirror_00159]

B.6.3 Deleted Specification Items in R19-11

none

B.7 Traceable Item History of this Document According to AUTOSAR Release 4.4.0

B.7.1 Added Specification Items in 4.4.0

[SWS_Mirror_00001] [SWS_Mirror_00002] [SWS_Mirror_00003] [SWS_Mirror_00004] [SWS_Mirror_00005] [SWS_Mirror_00006] [SWS_Mirror_00007] [SWS_Mirror_00008] [SWS_Mirror_00009] [SWS_Mirror_00011] [SWS_Mirror_00012] [SWS_Mirror_00013] [SWS_Mirror_00014] [SWS_Mirror_00015] [SWS_Mirror_00016] [SWS_Mirror_00017] [SWS_Mirror_00018] [SWS_Mirror_00019] [SWS_Mirror_00020] [SWS_Mirror_00021] [SWS_Mirror_00022] [SWS_Mirror_00023] [SWS_Mirror_00024] [SWS_Mirror_00025] [SWS_Mirror_00026] [SWS_Mirror_00027] [SWS_Mirror_00028] [SWS_Mirror_00029] [SWS_Mirror_00030] [SWS_Mirror_00031] [SWS_Mirror_00032] [SWS_Mirror_00033] [SWS_Mirror_00034]

[SWS_Mirror_00035] [SWS_Mirror_00036] [SWS_Mirror_00037] [SWS_Mirror_00038] [SWS_Mirror_00039] [SWS_Mirror_00040] [SWS_Mirror_00041] [SWS_Mirror_00042] [SWS_Mirror_00043] [SWS_Mirror_00044] [SWS_Mirror_00045] [SWS_Mirror_00046] [SWS_Mirror_00047] [SWS_Mirror_00048] [SWS_Mirror_00049] [SWS_Mirror_00050] [SWS_Mirror_00051] [SWS_Mirror_00052] [SWS_Mirror_00053] [SWS_Mirror_00054] [SWS_Mirror_00055] [SWS_Mirror_00056] [SWS_Mirror_00057] [SWS_Mirror_00058] [SWS_Mirror_00059] [SWS_Mirror_00060] [SWS_Mirror_00061] [SWS_Mirror_00062] [SWS_Mirror_00063] [SWS_Mirror_00064] [SWS_Mirror_00065] [SWS_Mirror_00066] [SWS_Mirror_00067] [SWS_Mirror_00068] [SWS_Mirror_00069] [SWS_Mirror_00070] [SWS_Mirror_00071] [SWS_Mirror_00072] [SWS_Mirror_00073] [SWS_Mirror_00074] [SWS_Mirror_00075] [SWS_Mirror_00076] [SWS_Mirror_00077] [SWS_Mirror_00078] [SWS_Mirror_00079] [SWS_Mirror_00080] [SWS_Mirror_00081] [SWS_Mirror_00082] [SWS_Mirror_00083] [SWS_Mirror_00084] [SWS_Mirror_00085] [SWS_Mirror_00086] [SWS_Mirror_00087] [SWS_Mirror_00088] [SWS_Mirror_00089] [SWS_Mirror_00090] [SWS_Mirror_00091] [SWS_Mirror_00092] [SWS_Mirror_00093] [SWS_Mirror_00094] [SWS_Mirror_00095] [SWS_Mirror_00096] [SWS_Mirror_00097] [SWS_Mirror_00098] [SWS_Mirror_00099] [SWS_Mirror_00100] [SWS_Mirror_00101] [SWS_Mirror_00102] [SWS_Mirror_00103] [SWS_Mirror_00104] [SWS_Mirror_00105] [SWS_Mirror_00106] [SWS_Mirror_00107] [SWS_Mirror_00108] [SWS_Mirror_00109] [SWS_Mirror_00110] [SWS_Mirror_00111] [SWS_Mirror_00112] [SWS_Mirror_00113] [SWS_Mirror_00114] [SWS_Mirror_00115] [SWS_Mirror_00116] [SWS_Mirror_00117] [SWS_Mirror_00118] [SWS_Mirror_00119] [SWS_Mirror_00120] [SWS_Mirror_00121] [SWS_Mirror_00122] [SWS_Mirror_00123] [SWS_Mirror_00124] [SWS_Mirror_00125] [SWS_Mirror_00126] [SWS_Mirror_00127] [SWS_Mirror_00128] [SWS_Mirror_00129] [SWS_Mirror_00131] [SWS_Mirror_00132] [SWS_Mirror_00133] [SWS_Mirror_00134] [SWS_Mirror_00135] [SWS_Mirror_00136] [SWS_Mirror_00137] [SWS_Mirror_00138] [SWS_Mirror_00142] [SWS_Mirror_00143] [SWS_Mirror_00144] [SWS_Mirror_00146] [SWS_Mirror_00147] [SWS_Mirror_00149] [SWS_Mirror_00150] [SWS_Mirror_00151] [SWS_Mirror_00152] [SWS_Mirror_00153] [SWS_Mirror_00154] [SWS_Mirror_00155] [SWS_Mirror_00156] [SWS_Mirror_00157] [SWS_Mirror_00158] [SWS_Mirror_00159] [SWS_Mirror_00160] [SWS_Mirror_00161] [SWS_Mirror_00165] [SWS_Mirror_01000] [SWS_Mirror_01002] [SWS_Mirror_01003] [SWS_Mirror_01004] [SWS_Mirror_01005] [SWS_Mirror_01006] [SWS_Mirror_01007] [SWS_Mirror_01008] [SWS_Mirror_01009] [SWS_Mirror_01010] [SWS_Mirror_01011] [SWS_Mirror_01012] [SWS_Mirror_01013] [SWS_Mirror_01014] [SWS_Mirror_01015] [SWS_Mirror_01016] [SWS_Mirror_01017] [SWS_Mirror_01018] [SWS_Mirror_01019] [SWS_Mirror_01020] [SWS_Mirror_01021] [SWS_Mirror_01022] [SWS_Mirror_01023] [SWS_Mirror_01024] [SWS_Mirror_01025] [SWS_Mirror_01026] [SWS_Mirror_01027] [SWS_Mirror_01028] [SWS_Mirror_01029] [SWS_Mirror_01030] [SWS_Mirror_01031] [SWS_Mirror_01033] [SWS_Mirror_01100] [SWS_Mirror_01101] [SWS_Mirror_01102] [SWS_Mirror_CONSTR_00001] [SWS_Mirror_CONSTR_00002] [SWS_Mirror_CONSTR_00003] [SWS_Mirror_CONSTR_00004]

B.7.2 Changed Specification Items in 4.4.0

none

B.7.3 Deleted Specification Items in 4.4.0

none