Document Title	Specification of Default Error Tracer
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	017
Document Classification	Standard
Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.0

Document Change History

Date	Release	Changed by	Change Description
2016-11-30	4.3.0	AUTOSAR Release Management	• Improved Sequence Diagrams • Added Description of Callouts (8.1.5) • Changed Port Defined Arguments in Service • Improved traceability • Added DetModuleInstance parameter • Made TransientFaults an BSW-Service
2015-07-31	4.2.2	AUTOSAR Release Management	• Harmonized Traceability • Ensured consistent usage of development errors in all modules
2014-10-31	4.2.1	AUTOSAR Release Management	• Extended & renamed DevelopmentErrorTracer to DefaultErrorTracer by adding routines • New Routines Det_ReportRountineError and Det_ReportTransientFault • New configuration paramaters Det_ReportRountineErrorCallout and Det_ReportTransientFaultCallout
2014-03-31	4.1.3	AUTOSAR Release Management	• Improved requirement format of SWS_DET_00050

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2013-10-31	4.1.2	AUTOSAR Release Management	• Structural but non-functional improvements in document structure and creation • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Harmonized requirements according to SWS_General • Formalized service descriptions
2011-12-22	4.0.3	AUTOSAR Administration	• Clarifications related to include structure etc.
2010-09-30	3.1.5	AUTOSAR Administration	• DLT is now an optional interface of DET • Harmonized parameter error handling • Removed known limitation of Revision 4.0.1
2010-02-02	3.1.4	AUTOSAR Administration	• Changed Tracing to requirements now located in SRS_Debugging • Added a Std_ReturnType value to Det_ReportError • Harmonized configuration classes • Adopted to the changed general requirements • Legal disclaimer revised • Chapter 10.3 revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2008-02-01	3.0.2	AUTOSAR Administration	• Added API GetVersionInfo to harmonize SWS with AUTOSAR conventions • Document meta information extended • Small layout adaptations made

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2007-12-21	3.0.1	AUTOSAR Administration	• Added SRS_BSW_00436 to traceability matrix • Added Memmap.h • Added Chapter 11 • Legal disclaimer revised • “Advice for users” revised • “Revision Information” added
2006-05-16	2.0	AUTOSAR Administration	• Changed to new SWS template
2005-05-31	1.0	AUTOSAR Administration	• Initial Release

Disclaimer

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and functional overview

This specification describes the API of the Default Error Tracer. All detected development and runtime errors in the Basic Software are reported to this module. The API parameters allow for tracing source and kind of error:

- Module in which error has been detected
- Function in which error has been detected
- Type of error

The functionality behind the API of this module is not in scope of this specification. It is up to the software developer and software integrator to choose the optimal strategy for his specific application and testing environment. Possible functionalities could be:

- Set debugger breakpoint within error reporting API
- Count reported errors
- Handle the runtime errors by using default values
- Log calls and passed parameters in RAM buffer
- Send reported errors via communication interface to external logger

Note: The software requirements of the Default Error Tracer are specified in the SRS Diagnostics document.

2 Acronyms and abbreviations

DET: Default Error Tracer.

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules,
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture,
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules,
AUTOSAR_SRS_BSWGeneral.pdf

[4] Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[5] Specification of ECU Configuration,
AUTOSAR_TPS_ECUConfiguration.pdf

[6] Requirements on Diagnostic
AUTOSAR_SRS_Diagnostic.pdf

[7] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related standards and norms

Not applicable.

3.3 Related specification

The functional requirements of the DET are contained in the SRS Diagnostics [6].

AUTOSAR provides a General Specification on Basic Software modules [7] (SWS BSW General), which is also valid for Default Error Tracer.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Specification Error Tracer.

4 Constraints and assumptions

4.1 Limitations

This specification does not define the functionality behind the error reporting API.

Memory protection mechanisms of the operating system are not taken into account. It is assumed that the whole Basic Software runs in supervisor mode or the switch to supervisor mode is done by a system call within the error reporting function of the DET module.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

5.1 File structure

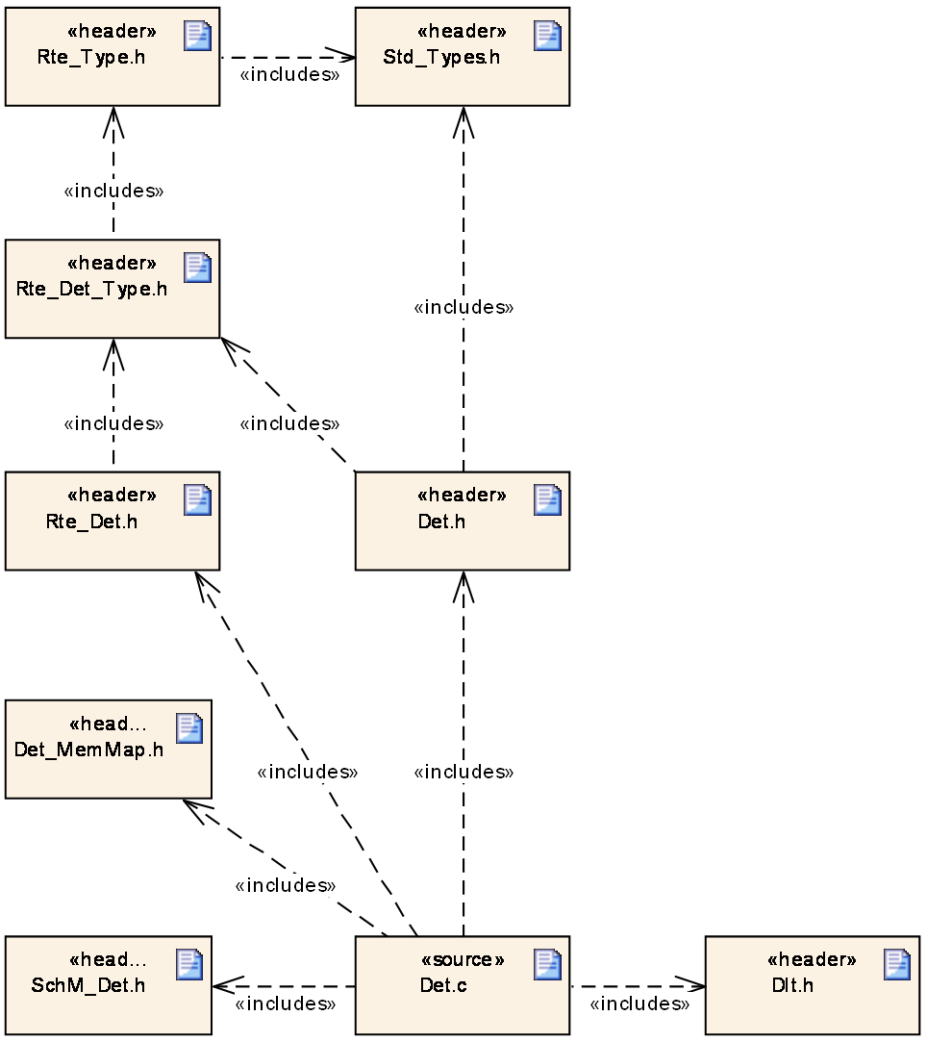

Figure 1: DET Header Structure

[SWS_Det_00004] [The Default Error Tracer module's source code shall offer a header file Det.h, see Figure 1.**]** (SRS_BSW_00300, SRS_BSW_00302, SRS_BSW_00321)

[SWS_Det_00037] [Det.h includes all user relevant information for the tracing of errors reported via its services.**]** (SRS_BSW_00346)

6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00004	All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files	SWS_Det_00999
SRS_BSW_00005	Modules of the µC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_Det_00999
SRS_BSW_00006	The source code of software modules above the µC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_Det_00999
SRS_BSW_00007	All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.	SWS_Det_00999
SRS_BSW_00009	All Basic SW Modules shall be documented according to a common standard.	SWS_Det_00999
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_Det_00999
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_Det_00019, SWS_Det_00020
SRS_BSW_00158	All modules of the AUTOSAR Basic Software shall strictly separate configuration from implementation	SWS_Det_00999
SRS_BSW_00159	All modules of the AUTOSAR Basic Software shall support a tool based configuration	SWS_Det_00018
SRS_BSW_00160	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	SWS_Det_00999
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_Det_00999
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_Det_00999
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_Det_00999
SRS_BSW_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_Det_00035
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_Det_00999
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_Det_00999
SRS_BSW_00171	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	SWS_Det_00015

SRS_BSW_00172	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	SWS_Det_00999
SRS_BSW_00300	All AUTOSAR Basic Software Modules shall be identified by an unambiguous name	SWS_Det_00004
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only import the necessary information	SWS_Det_00999
SRS_BSW_00302	All AUTOSAR Basic Software Modules shall only export information needed by other modules	SWS_Det_00004
SRS_BSW_00304	All AUTOSAR Basic Software Modules shall use the following data types instead of native C data types	SWS_Det_00999
SRS_BSW_00305	Data types naming convention	SWS_Det_00999
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_Det_00999
SRS_BSW_00307	Global variables naming convention	SWS_Det_00999
SRS_BSW_00308	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	SWS_Det_00999
SRS_BSW_00309	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	SWS_Det_00999
SRS_BSW_00310	API naming convention	SWS_Det_00008, SWS_Det_00009, SWS_Det_00010, SWS_Det_00011, SWS_Det_01001, SWS_Det_01003
SRS_BSW_00312	Shared code shall be reentrant	SWS_Det_00039
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_Det_00999
SRS_BSW_00318	Each AUTOSAR Basic Software Module file shall provide version numbers in the header file	SWS_Det_00011
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	SWS_Det_00004
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_Det_00999
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_Det_00999
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_Det_00999
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_Det_00999

SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_Det_00999
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_Det_00999
SRS_BSW_00335	Status values naming convention	SWS_Det_00999
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_Det_00999
SRS_BSW_00337	Classification of development errors	SWS_Det_00026, SWS_Det_00301
SRS_BSW_00339	Reporting of production relevant error status	SWS_Det_00999
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_Det_00999
SRS_BSW_00342	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	SWS_Det_00999
SRS_BSW_00343	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	SWS_Det_00999
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_Det_00999
SRS_BSW_00345	BSW Modules shall support pre-compile configuration	SWS_Det_00014
SRS_BSW_00346	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	SWS_Det_00037
SRS_BSW_00347	A Naming seperation of different instances of BSW drivers shall be in place	SWS_Det_00999
SRS_BSW_00348	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	SWS_Det_00999
SRS_BSW_00350	All AUTOSAR Basic Software Modules shall allow the enabling/disabling of detection and reporting of development errors.	SWS_Det_00025, SWS_Det_00999
SRS_BSW_00353	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	SWS_Det_00999
SRS_BSW_00357	For success/failure of an API call a standard return type shall be defined	SWS_Det_00999
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_Det_00008
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_Det_00999
SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_Det_00999
SRS_BSW_00361	All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header	SWS_Det_00999

SRS_BSW_00369	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	SWS_Det_00999
SRS_BSW_00371	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules	SWS_Det_00999
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	SWS_Det_00999
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_Det_00999
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_Det_00999
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_Det_00999
SRS_BSW_00379	All software modules shall provide a module identifier in the header file and in the module XML description file.	SWS_Det_00999
SRS_BSW_00380	Configuration parameters being stored in memory shall be placed into separate c-files	SWS_Det_00999
SRS_BSW_00381	The pre-compile time parameters shall be placed into a separate configuration header file	SWS_Det_00999
SRS_BSW_00383	The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description	SWS_Det_00999
SRS_BSW_00385	List possible error notifications	SWS_Det_00999
SRS_BSW_00386	The BSW shall specify the configuration for detecting an error	SWS_Det_00999
SRS_BSW_00388	Containers shall be used to group configuration parameters that are defined for the same object	SWS_Det_00999
SRS_BSW_00389	Containers shall have names	SWS_Det_00999
SRS_BSW_00390	Parameter content shall be unique within the module	SWS_Det_00999
SRS_BSW_00392	Parameters shall have a type	SWS_Det_00035
SRS_BSW_00393	Parameters shall have a range	SWS_Det_00999
SRS_BSW_00394	The Basic Software Module specifications shall specify the scope of the configuration parameters	SWS_Det_00035, SWS_Det_00180
SRS_BSW_00395	The Basic Software Module specifications shall list all configuration parameter dependencies	SWS_Det_00999
SRS_BSW_00396	The Basic Software Module specifications shall specify the supported configuration classes for changing values and multiplicities for each parameter/container	SWS_Det_00999
SRS_BSW_00397	The configuration parameters in pre-compile time are fixed before compilation starts	SWS_Det_00999

SRS_BSW_00398	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	SWS_Det_00999
SRS_BSW_00399	Parameter-sets shall be located in a separate segment and shall be loaded after the code	SWS_Det_00999
SRS_BSW_00400	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	SWS_Det_00999
SRS_BSW_00401	Documentation of multiple instances of configuration parameters shall be available	SWS_Det_00999
SRS_BSW_00403	The Basic Software Module specifications shall specify for each parameter/container whether it supports different values or multiplicity in different configuration sets	SWS_Det_00018
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_Det_00999
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_Det_00999
SRS_BSW_00406	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	SWS_Det_00024, SWS_Det_00999
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_Det_00999
SRS_BSW_00409	All production code error ID symbols are defined by the Dem module and shall be retrieved by the other BSW modules from Dem configuration	SWS_Det_00999
SRS_BSW_00410	Compiler switches shall have defined values	SWS_Det_00999
SRS_BSW_00412	References to c-configuration parameters shall be placed into a separate h-file	SWS_Det_00999
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_Det_00999
SRS_BSW_00414	Init functions shall have a pointer to a configuration structure as single parameter	SWS_Det_00008, SWS_Det_00210
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_Det_00999
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_Det_00999
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_Det_00999
SRS_BSW_00419	If a pre-compile time configuration parameter is implemented as "const" it should be placed into a separate c-file	SWS_Det_00999
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_Det_00999
SRS_BSW_00423	BSW modules with AUTOSAR interfaces	SWS_Det_00999

	shall be describable with the means of the SW-C Template	
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_Det_00999
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_Det_00999
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_Det_00999
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_Det_00999
SRS_BSW_00428	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	SWS_Det_00999
SRS_BSW_00429	BSW modules shall be only allowed to use OS objects and/or related OS services	SWS_Det_00999
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_Det_00999
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_Det_00999
SRS_BSW_00437	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	SWS_Det_00999
SRS_BSW_00438	Configuration data shall be defined in a structure	SWS_Det_00999
SRS_BSW_00439	Enable BSW modules to handle interrupts	SWS_Det_00999
SRS_BSW_00440	The callback function invocation by the BSW module shall follow the signature provided by RTE to invoke servers via Rte_Call API	SWS_Det_00999
SRS_BSW_00441	Naming convention for type, macro and function	SWS_Det_00999
SRS_BSW_00458	Classification of production errors	SWS_Det_00999
SRS_BSW_00463	Naming convention of callout prototypes	SWS_Det_00180, SWS_Det_00181, SWS_Det_00184, SWS_Det_00187
SRS_BSW_00466	Classification of extended production errors	SWS_Det_00999
SRS_BSW_00480	NullPointer Errors shall follow a naming rule	SWS_Det_00052
SRS_Diag_04085	The Default Error Tracer shall provide an interface to receive error reports	SWS_Det_00009
SRS_Diag_04086	Report errors shall contain a dedicated set of information	SWS_Det_00009, SWS_Det_01001, SWS_Det_01003
SRS_Diag_04087	The Default Error Tracer shall provide a development error report reception service	SWS_Det_00200, SWS_Det_00202,

		SWS_Det_00203, SWS_Det_00204, SWS_Det_00205, SWS_Det_00206
SRS_Diag_04089	Fan-out of received error reports	SWS_Det_00207
SRS_Diag_04101	The DET module shall forward its trace events to the DLT	SWS_Det_00034
SRS_Diag_04143	The Default Error Tracer shall provide an interface to receive runtime error reports	SWS_Det_01001
SRS_Diag_04144	The Default Error Tracer shall provide an interface to receive transient fault reports	SWS_Det_01003

7 Functional specification

The Default Error Tracer provides functionality to support error detection and tracing of errors during the development and runtime of Software Components and other Basic Software Modules. For this purpose the Default Error Tracer receives and evaluates error messages from these components and modules.

Due to the always specific (non generic!) requirements regarding functionality in error cases there is no explicit specification of the DET implementation, except:

- Configurable lists of error hooks will be executed in case of an error report.
- Interfaces will be provided to report errors, allow optional error recovery after reset, to handle optional error recovery information and to retrieve version information.

7.1 Initialization

[SWS_Det_00019] ⌈ The DET shall provide the initialization function Det_Init (see SWS_Det_00008).⌋ (SRS_BSW_00101)

[SWS_Det_00020] ⌈ Each call of the Det_Init function shall be used to set the Default Error Tracer to a defined initial status (e.g. by removing optional error recovery information).⌋ (SRS_BSW_00101)

Note: SWS_Det_00020 is not testable without knowledge about the non specified functionality and the probably used optional error recovery information.

Note: The usage and meaning of error recovery information is optional and not specified.

[SWS_Det_00025] ⌈ The Default Error Tracer shall provide the function Det_Start (see SWS_Det_00010).⌋ (SRS_BSW_00350)

Note: The Default Error Tracer's environment can use the function Det_Start to trigger the Default Error Tracer module for instance (if needed) in case of completed NVRAM initialization for persistent error storage.

Note: In case the Default Error Tracer does not require a startup call the Det_Start function can be empty.

Note: The integrator can decide by configuration of the EcuM, when Det_Init will be called.

Note: The integrator can decide by configuration of the EcuM or ModeM, when and whether Det_Start will be called.

7.2 Error Hooks

[SWS_Det_00207] ⌈ To support debugging and error tracing during development and runtime, the Default Error Tracer provides functionality for notification of received

error reports. Therefore so called error hooks are configurable. The error hooks will be used to forward error notifications. If at least one error hook has been configured, the Default Error Tracer will notify each received error report by calling the configured error hook(s). Configuration of error hooks is done by the AUTOSAR configuration methods described in chapter 10.⌋ (SRS_Diag_04089)

[SWS_Det_00035] ⌈ Each Error_Hook shall be called with the same set of parameters as the corresponding functions Det_ReportError, Det_ReportTransientFault and Det_ReportRuntimeError. The configured callout functions are ECU configurations, see ECUC_DET_00005, ECUC_DET_00010 and ECUC_DET_00011⌋ (SRS_BSW_00167,SRS_BSW_00392, SRS_BSW_00394)

7.3 Error Reporting

[SWS_Det_00024] ⌈ If the Default Error Tracer has not been initialized before Det reporting functions are called, the reporting functions shall return immediately without any other action (no Error_Hook shall be used, no implementer specific function shall be performed and no error shall be reported).⌋ (SRS_BSW_00406)

[SWS_Det_00014] ⌈ The error report functions Det_ReportError, Det_ReportTransientFault and Det_ReportRuntimeError shall call immediately all configured Error_Hooks (see ECUC_Det_00010, ECUC_Det_00011).
⌋ (SRS_BSW_00345)

[SWS_Det_00018] ⌈ The Default Error Tracer shall execute the corresponding list of configured DetErrorHook (refer to ECUC_Det_00005) in the order given by the configuration.⌋ (SRS_BSW_00403,SRS_BSW_00159)

[SWS_Det_00015] ⌈ Optional implementation specific functionality shall only be performed after all configured Error_Hooks (see ECUC_Det_00010 and ECUC_Det_0011) have been called. Furthermore this functionality shall be pre-compile-time configurable⌋ (SRS_BSW_00171)

[SWS_Det_00034] ⌈ Each call of the Det_ReportError, Det_ReportTransientFault and Det_ReportRuntimeError function shall be forwarded to the DLT module, if this is available/configured.⌋ (SRS_Diag_04101)

[SWS_Det_00039] ⌈ The Det_ReportError, Det_ReportTransientFault and Det_ReportRuntimeError functions shall be reentrant.⌋ (SRS_BSW_00312)

[SWS_Det_00026] ⌈ Det_ReportError shall stop execution. Ensure that DET runtime errors and DET transient faults are handled such that DET is not called recursively.⌋ (SRS_BSW_00337)

Note: Such recursive call could happen in case of calling an un-initialized module via an Error_Hook and would lead to a stack overflow.

7.4 Version Information

No deviations from specified handling in BSW_General.

7.5 Error Classification

The Default Error Tracer has the following AUTOSAR errors:

- Development errors, see Section 7.5.1
- Runtime errors, see Section 7.5.2
- Transient faults: see Section 7.5.3
- Production errors: not applicable
- Extended production errors:not applicable

7.5.1 Development Errors

The following development errors can be reported from DET:

[SWS_Det_00301] Development Error Types
[

Type of error	*Related error code*	*Value [hex]*
Det_GetVersionInfo called with null parameter pointer	DET_E_PARAM_POINTER	0x01

] (SRS_BSW_00337)

7.5.2 Runtime Errors

[SWS_Det_00302] Runtime Error Types

Type of error	*Related error code*	*Value [hex]*
Det unable to report errors, e.g. to lack of ressources	DET_E_CANNOT_REPORT	0x01
Wrong Module parameter	DET_E_WRONG_MODULE	0x02
Wrong Instance ID	DET_E_WRONG_INSTANCE	0x03
Wrong API parameter	DET_E_WRONG_API	0x04
Wrong Error	DET_E_WRONG_ERROR	0x05

] (SRS_BSW_00452)

7.5.3 Transient Faults

[SWS_Det_00303] Transient Faults Types

Type of error	*Related error code*	*Value [hex]*
DET currently unavailable	DET_E_UNAVAILABLE	0x01

] (SRS_BSW_00473)

7.5.4 Production Errors

There are no production errors in DET.

7.5.5 Extended Production Errors

There are no extended production errors in DET.

7.6 Error detection

The call of default error functions will cause calls to all configured callout functions see parameter DetErrorHook, DetReportTransientFault and DetReportRuntimeError.

[SWS_Det_00501] [The calls of Det_ReportError shall invoke all callback functions configured in DetErrorHook (see parameter DetErrorHook, ECUC_Det_00005).(SRS_BSW_00345)

[SWS_Det_00502] [The calls of Det_ReportTransientFault shall invoke all callback functions configured in DetReportTransientFaultCallout (ECUC_Det_00011). (SRS_BSW_00345)

[SWS_Det_00503] [The calls of Det_ReportRuntimeError shall invoke all callback functions configured in DetReportRuntimeErrorCallout (ECUC_Det_00010). (SRS_BSW_00345)

Note: In case no Error_Hooks are configured no additional functions are called. However the forwarding to DLT (see DET0340 and SWS_Det_00006_Conf) is still active if configured.

7.7 Error notification

[SWS_Det_00052] [The DET shall notify the error DET_E_PARAM_POINTER to all functions configured in callouts in case a null pointer error occurs.] (SRS_BSW_00480)

8 API specification

The specification of the default error tracer API is provided here.

8.1 API

8.1.1 Imported types

This section lists all imported types used by the API. Even if the DET does not require new types, some RTE or Component types can be used within the configuration of the hook functions. Therefore the DET also has the standardized include structure (see SRS_BSW_00447) for modules with service interfaces.

Module	Imported Type
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.1.2 Type definitions

8.1.2.1 Det_ConfigType

[SWS_Det_00210] [

Name:	`Det_ConfigType`	
Type:	`Structure`	
Range:	`implementation specific`	--
Description:	Configuration data structure of the Det module.	

] (SRS_BSW_00414)

8.1.3 Function definitions

8.1.3.1 Det_Init

[SWS_Det_00008] [

Service name:	Det_Init	
Syntax:	`void Det_Init( const Det_ConfigType* ConfigPtr )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfigPtr	Pointer to the selected configuration set.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Service to initialize the Default Error Tracer.	

] (SRS_BSW_00310, SRS_BSW_00358, SRS_BSW_00414)

8.1.3.2 Det_ReportError

[SWS_Det_00009] ⌈

Service name:	Det_ReportError	
Syntax:	`Std_ReturnType Det_ReportError( uint16 ModuleId, uint8 InstanceId, uint8 ApiId, uint8 ErrorId )`	
Service ID[hex]:	0x01	
Sync/Async:	Depending on implemented functionality: 1. Breakpoint set: no return 2. Internal error counting/logging in RAM: synchronous 3. External error logging via communication interface: asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	ModuleId	Module ID of calling module.
	InstanceId	The identifier of the index based instance of a module, starting from 0, If the module is a single instance module it shall pass 0 as the InstanceId.
	ApiId	ID of API service in which error is detected (defined in SWS of calling module)
	ErrorId	ID of detected development error (defined in SWS of calling module).
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	returns always E_OK (is required for services)
Description:	Service to report development errors.	

⌋ (SRS_BSW_00310, SRS_Diag_04086, SRS_Diag_04085)

Note: Det_ReportError may be callable in interrupt context. Since the DET can be called in normal mode or in interrupt context (from stack or integration) this has to be considered during implementation of the hook functions: Det_ReportError can be called in interrupt context; this should be considered when halting the system.

8.1.3.3 Det_Start

[SWS_Det_00010] ⌈

Service name:	Det_Start
Syntax:	`void Det_Start( void )`
Service ID[hex]:	0x02
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Service to start the Default Error Tracer.

⌋ (SRS_BSW_00310)

8.1.3.4 Det_ReportRuntimeError

[SWS_Det_01001] ⌈

Service name:	Det_ReportRuntimeError	
Syntax:	`Std_ReturnType Det_ReportRuntimeError( uint16 ModuleId, uint8 InstanceId, uint8 ApiId, uint8 ErrorId )`	
Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	ModuleId	Module ID of calling module.
	InstanceId	The identifier of the index based instance of a module, starting from 0, If the module is a single instance module it shall pass 0 as the InstanceId.
	ApiId	ID of API service in which error is detected (defined in SWS of calling module)
	ErrorId	ID of detected runtime error (defined in SWS of calling module).
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	returns always E_OK (is required for services)
Description:	Service to report runtime errors. If a callout has been configured then this callout shall be called.	

] (SRS_BSW_00310, SRS_Diag_04086, SRS_Diag_04143)

Note: Det_ReportRuntimeError may be callable in interrupt context. Since the DET can be called in normal mode or in interrupt context (from stack or integration) this has to be considered during implementation of the hook functions: Det_ReportRuntimeError can be called in interrupt context; this hook should be reentrant and sufficiently performant.

8.1.3.5 Det_ReportTransientFault

[SWS_Det_01003] [

Service name:	Det_ReportTransientFault	
Syntax:	`Std_ReturnType Det_ReportTransientFault( uint16 ModuleId, uint8 InstanceId, uint8 ApiId, uint8 FaultId )`	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	ModuleId	Module ID of calling module.
	InstanceId	The identifier of the index based instance of a module, starting from 0, If the module is a single instance module it shall pass 0 as the InstanceId.
	ApiId	ID of API service in which transient fault is detected (defined in SWS of calling module)
	FaultId	ID of detected transient fault (defined in SWS of calling module).
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	Propagates return value of assigned callout if exists, otherwise E_OK.
Description:	Service to report transient faults. If a callout has been configured than this callout shall be called and the returned value of the callout shall be returned. Otherwise it	

	returns immediately with E_OK.

] (SRS_BSW_00310, SRS_Diag_04086, SRS_Diag_04144)

Note: Det_ReportTransientFault may be callable in interrupt context. Since the DET can be called in normal mode or in interrupt context (from stack or integration) this has to be considered during implementation of the hook functions: Det_ReportTransientFault can be called in interrupt context; this hook should be reentrant and sufficiently performant.

8.1.3.6 Det_GetVersionInfo

[SWS_Det_00011] [

Service name:	Det_GetVersionInfo

Syntax:	`void Det_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.

] (SRS_BSW_00310, SRS_BSW_00318)

8.1.4 Expected Interfaces

This chapter specifies all required interfaces of other modules.

8.1.4.1 Mandatory Interfaces

There is no mandatory expected interface, but all <User_ErrorHooks> APIs that are used and are configured as callouts have to be included.

API function	Description

Note: The name of the user API will not be specified, <User_ErrorHook> is a synonym only.

Note: A list of User_ErrorHook can be defined.

8.1.4.2 Optional Interfaces

This chapter defines the interfaces that are required to fulfill an optional functionality of the Default Error Tracer.

API function	Description
Dlt_DetForwardErrorTrace	Service to forward error reports from Det to Dlt.

8.1.5 Callout Functions / Configurable Interfaces

[SWS_Det_00180] ⌈ if callout functions are configured, they should have the same signatures as the corresponding functions. If several callouts are defined for the same service they should have the same ID.⌋ (SRS_BSW_00463,SRS_BSW_00394)

[SWS_Det_00181] ⌈ If Det_ReportError function is called, all configured callout functions shall be called. User_ErrorHooks functions should have the ID 0x10. **[]** ⌈

Service name:	<User_ErrorHooks>

Syntax:	`void <User_ErrorHooks>(`
	`  void`
	`)`

Service ID[hex]:	0x10
Sync/Async:	Asynchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	--

⌋ () ⌋ (SRS_BSW_00463)

[SWS_Det_00184] ⌈ If Det_ReportRuntimeError function is called, all configured callout functions shall be called. DetReportRuntimeErrorCallout functions should have the ID 0x11.
[] ⌈

Service name:	<DetReportRuntimeErrorCallout>

Syntax:	`void <DetReportRuntimeErrorCallout>(`
	`  void`
	`)`

Service ID[hex]:	0x11
Sync/Async:	Asynchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	--

⌋ () ⌋ (SRS_BSW_00463)

[SWS_Det_00187] ⌈ If Det_ReportTransientFault function is called, all configured callout functions shall be called. DetReportTransientFaultCallout functions should have the ID 0x12. **[]** ⌈

Service name:	<DetReportTransientFaultCallout>

Syntax:	`void <DetReportTransientFaultCallout>(`
	`  void`
	`)`

Service ID[hex]:	0x12
Sync/Async:	Asynchronous

Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	--

] ()] (SRS_BSW_00463)

8.2 Service Interfaces

8.2.1 Scope of this Chapter

This chapter defines the AUTOSAR Interfaces of the Default Error Tracer Service (DET).

The definitions in this section are interpreted to be in `ARPackage AUTOSAR/Services/DET`.

8.2.2 Overview

The Default Error Tracer BSW module was originally developed for supporting the other BSW modules during development phase. It is however possible to also use the Default Error Tracer to perform error tracing during development and runtime of SW-C's.

8.2.2.1 Use Case

On each ECU, there is one instance of the Default Error Tracer Service and several Atomic Software Component instances named “clients” which interact with the Default Error Tracer Service.

Each component instance may report a development error to the Default Error Tracer Service, and these errors can then be analyzed with debugging tools. The behavior of the Default Error Tracer is not standardized (except the optional forwarding of error reports to the DLT as described in chapter 7.3). Thus, what happens when using DET Service might differ on different ECUs or different Default Error Tracer implementations. The Default Error Tracer might for example do one of the following:

- Set debugger breakpoint within error reporting API
- Count reported errors
- Handle runtime errors by providing default values
- Log calls and passed parameters in RAM buffer
- Send reported errors via communication interface to external logger

Note: For debugging and tracing purpose e.g. in an Error Hook, the use of additional functionalities offered by “Debugging Module” and “Log and Trace Module” should be considered.

8.2.3 Specification of the Ports and Port Interfaces

This chapter specifies the ports and port interfaces which are needed in order to operate the Default Error Tracer functionality over the VFB.

Each AUTOSAR SW-C which uses the service must contain “service ports” in its own SW-C description which will be typed by the same interfaces and which has to be connected to the ports of the Default Error Tracer, so that the RTE, the appropriate IDs and the required symbols can be generated.

8.2.3.1 General Approach

The client-server paradigm is used since more than one parameter has to be transferred.

In order to reuse the C API already defined in the Default Error Tracer BSW module, the Default Error Tracer services uses the same argument names as in the C API, even though the names can not directly be mapped into the SW-C world. "Module ID" can preferably be interpreted as either a component or runnable entity but this is the decision of the implementer of the SW-C.

The Default Error Tracer services need a "Module ID" as first argument for the C-function.

In order to keep the client code independent from the configuration of number of clients, the "Module IDs" are not passed from the clients to Default Error Tracer but are modeled as "port defined argument values" of the Provide ports on the Default Error Tracer side. As a consequence, the "Module IDs" will not show up as arguments in the operation of the client-server interface. As a further consequence for this approach, there will be separate ports for each "Module ID" both on the client side as well as on the server side.

The Module ID type is of range 0…65535. Values in the range of 0...255 are reserved for Basic Software Modules, all others can be used for application software components.

8.2.3.2 Data Types

[SWS_Det_00200] ⌈ For the port interface of the Default Error Tracer service uint8 and uint16 are required and refer to the AUTOSAR data types. ⌋ (SRS_Diag_04087)

8.2.3.3 Port Interface

[SWS_Det_00202] ⌈

Name	DETService
Comment	Service of Default Error Tracer
IsService	true
Variation	--

Possible Errors	0	E_OK

Operations

ReportError			
Comments	calls Det_ReportError with the Module ID of the port		
Variation	--		
Parameters	ApiId	Comment	ID of API service in which error is detected (defined in SWS of calling module).
		Type	uint8
		Variation	--
		Direction	IN
	ErrorId	Comment	ID of detected development error (defined in SWS of calling module).
		Type	uint8
		Variation	--
		Direction	IN
Possible Errors	E_OK	Successfully Reported Error	

ReportRuntimeError			
Comments	calls ReportRuntimeError with the Module ID of the port		
Variation	--		
Parameters	ApiId	Comment	ID of API service in which error is detected (defined in SWS of calling module).
		Type	uint8
		Variation	--
		Direction	IN
	ErrorId	Comment	ID of detected runtime error (defined in SWS of calling module).
		Type	uint8
		Variation	--
		Direction	IN
Possible Errors	E_OK	Successfully Reported Error	

⌋ (SRS_Diag_04087)

[SWS_Det_00203] ⌈ The arguments of the C-Api ModuleId and InstanceId are used to identify the component and component instance by using "port defined argument values". The arguments ApiId and ErrorId are not standardized by AUTOSAR for software components. It is up to the implementer of a SW-C to decide about the semantics of the arguments. However, the ApiId typically corresponds to the operations that can report an error, and ErrorId corresponds to the type of error that is reported. Both ApiId and ErrorId are numbered 0x00..0xFF without specific order. Note that the returned values is always true (E_OK), since a Std_ReturnType is required for all services⌋ (SRS_Diag_04087)

8.2.4 Definition of the Service

[SWS_Det_00204] ⌈ The Provide Ports have a certain relation to the internal behavior of the DET: With each call, the "Module ID" is passed as an additional argument by the RTE to the C-function which implements the associated runnable entity (feature “port defined argument value”).⌋ (SRS_Diag_04087)

The DET shall provide the following Port for each configured SWC module with the given name.

[SWS_Det_00205] ⌈

Name	Det_{Name}		
Kind	ProvidedPort	Interface	DETService
Description	--		
Port Defined Argument Value(s)	Type	uint16	
	Value	{ecuc(Det/DetConfigSet/DetModule/DetModuleId.value)}	
	Type	uint8	
	Value	{ecuc(Det/DetConfigSet/DetModule/DetModuleInstance/DetInstanceId.value)}	
Variation	Name = {ecuc(Det/DetConfigSet/DetModule.SHORT-NAME)}_{ecuc(Det/DetConfigSet/DetModule/DetModuleInstance.SHORT-NAME)}		

⌋ (SRS_Diag_04087)

8.2.5 Configuration of the DET

[SWS_Det_00206] ⌈ The "Module IDs" of the DET service are modeled as “port defined argument values”. Thus the configuration of those values is part of the RTE configuration. Pre-compile configuration can be done by changing the XML specification for the ports on the client (SW-C) or service (i.e. DET) side.⌋ (SRS_Diag_04087)

9 Sequence diagrams

9.1 Initialization

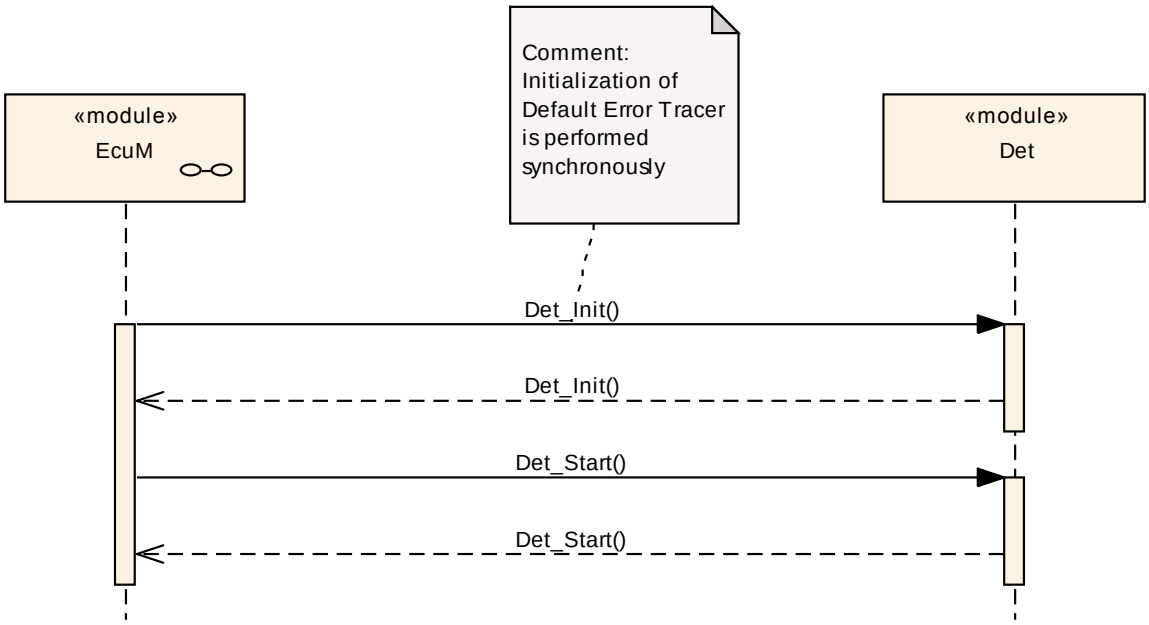

Figure 2: Initialization and start of DET

9.2 Error Reporting

There are different scenarios: one for each error class (DevelopmentError, RuntimeError and TransientFault) and one for each configuration: no hooks configured, at least one hook configured.

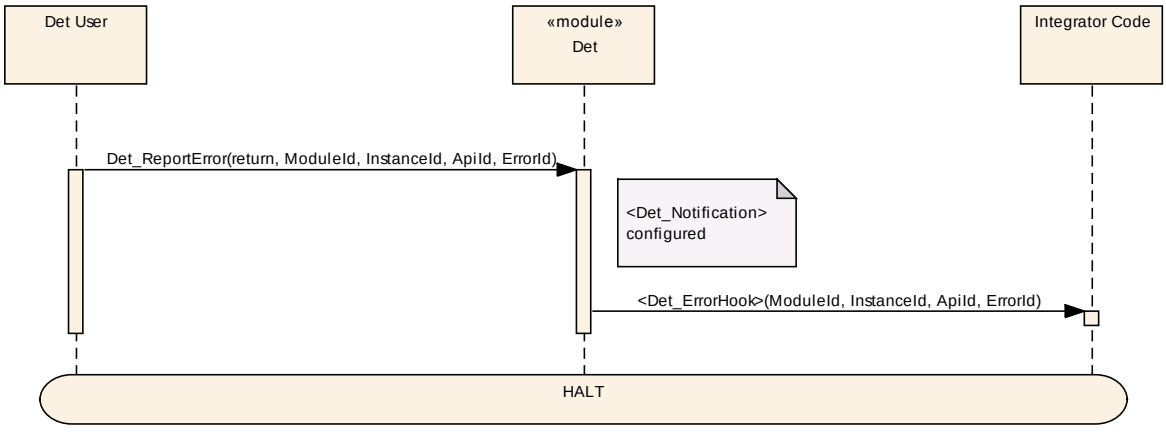

Figure 3 – Det_ReportError with configured hook

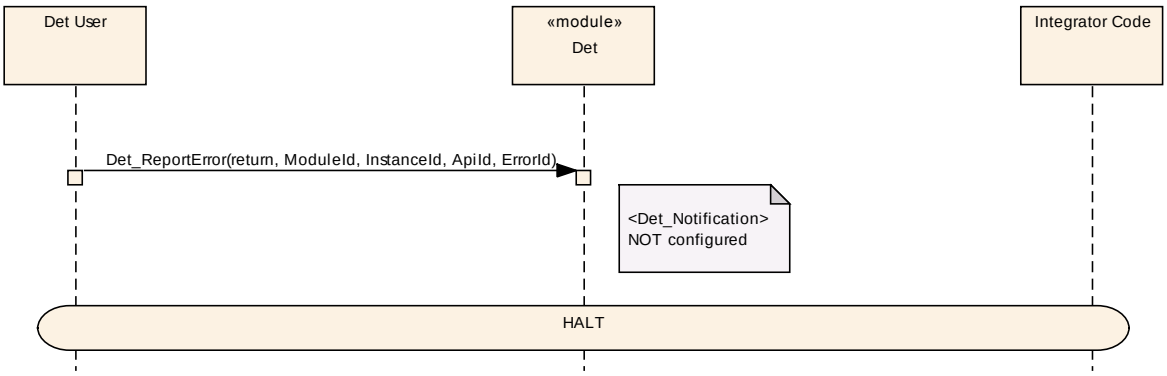

Figure 4 – Det_ReportError without configured hook

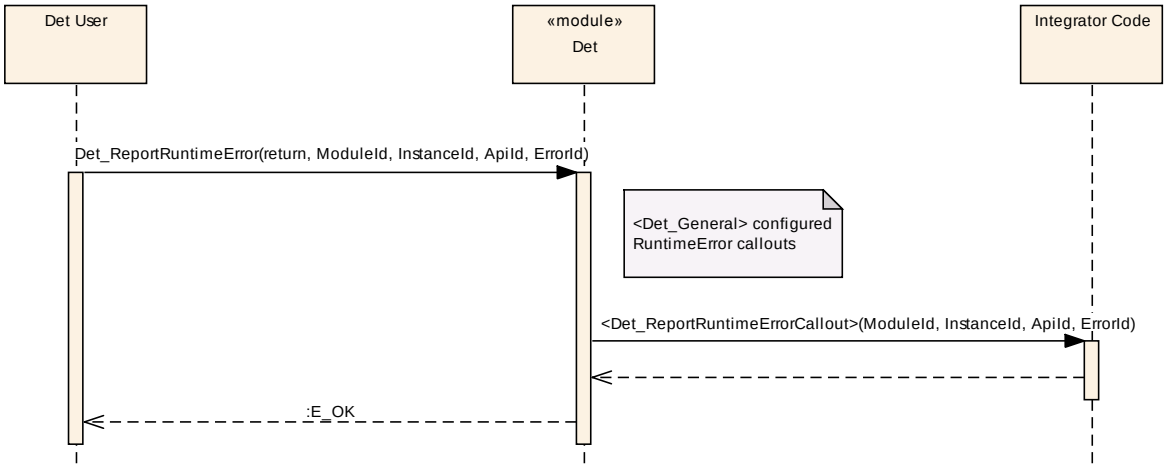

Figure 5 – Det_ ReportRuntimeError with configured hook

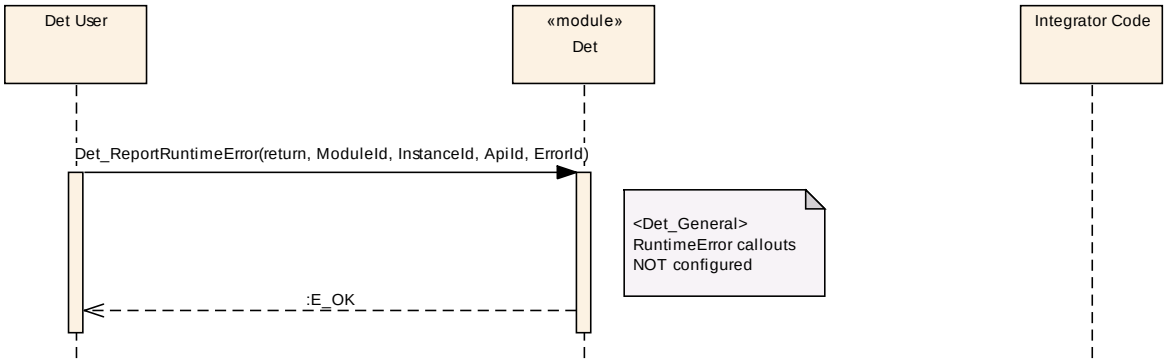

Figure 6 – Det_ ReportRuntimeError without configured hook

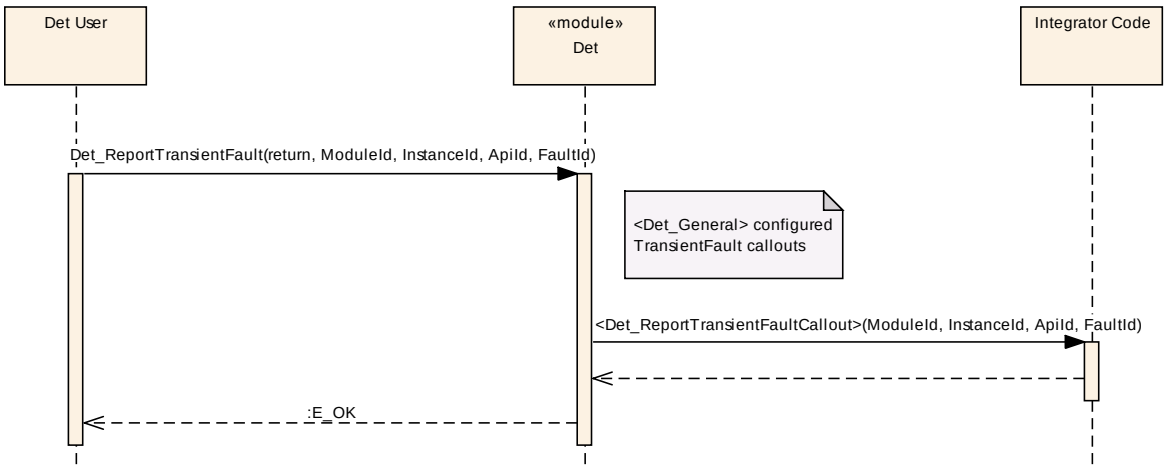

Figure 7 – Det_ ReportTransientFault with configured hook

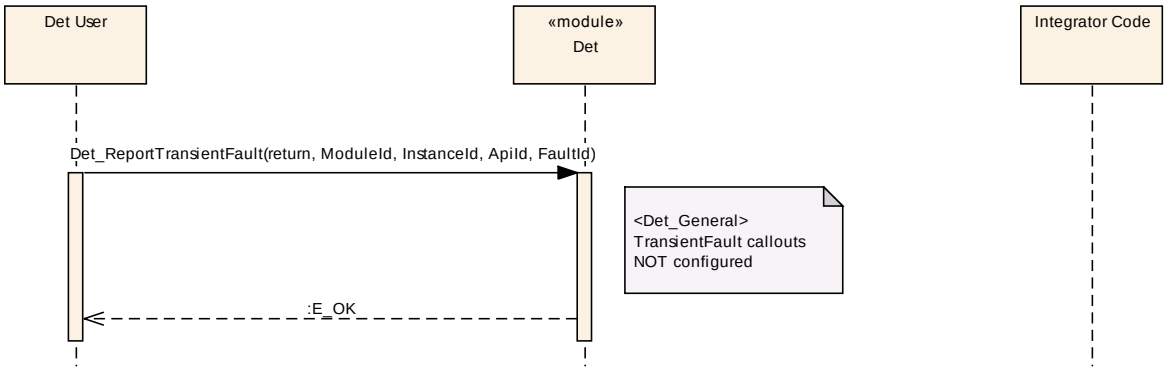

Figure 8 – Det_ ReportTransientFault without configured hook

10 Configuration specification

10.1 Containers and configuration parameters

The Parameters of DET are described in the following sub-sections.

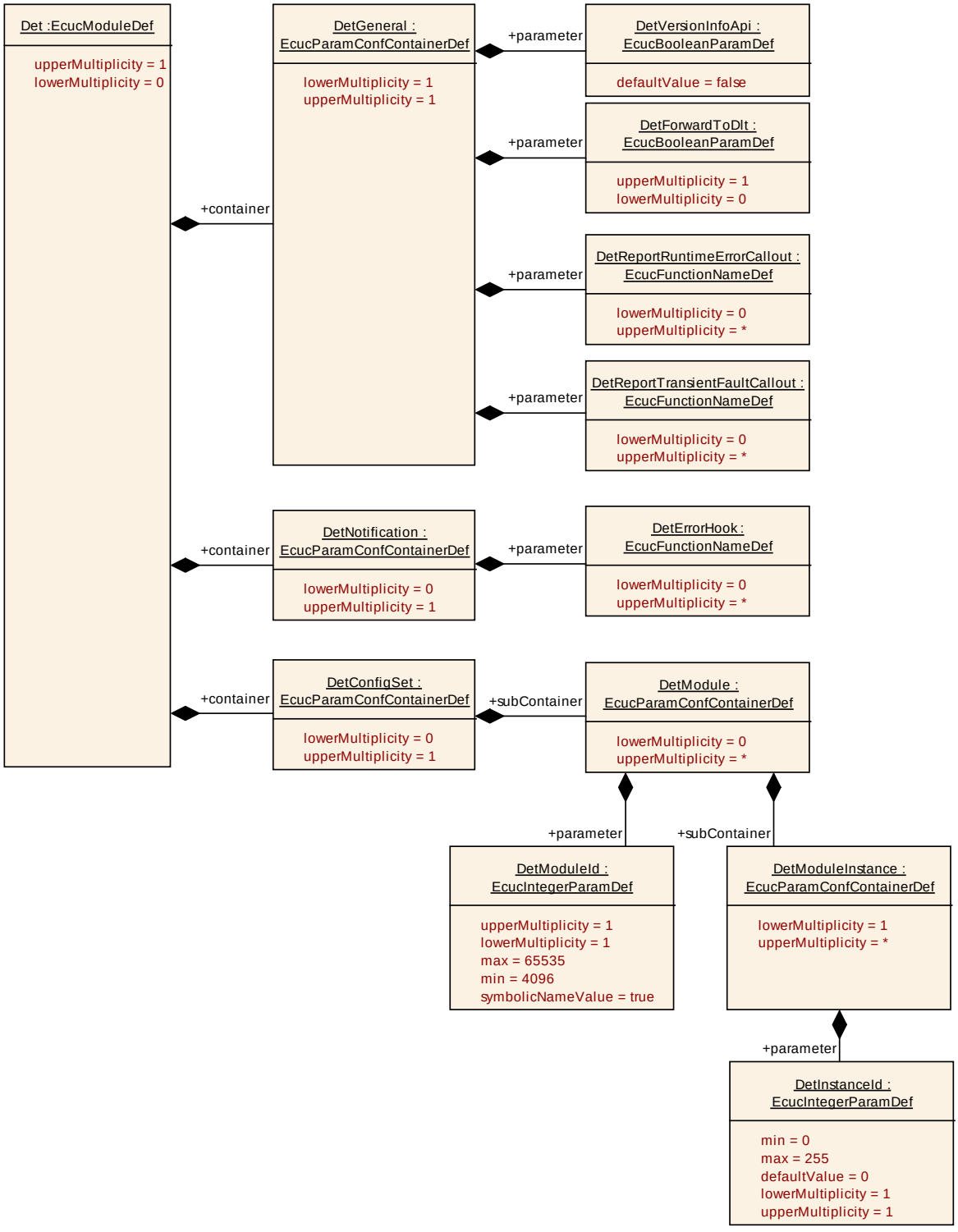

Figure 9 gives an overview over them.

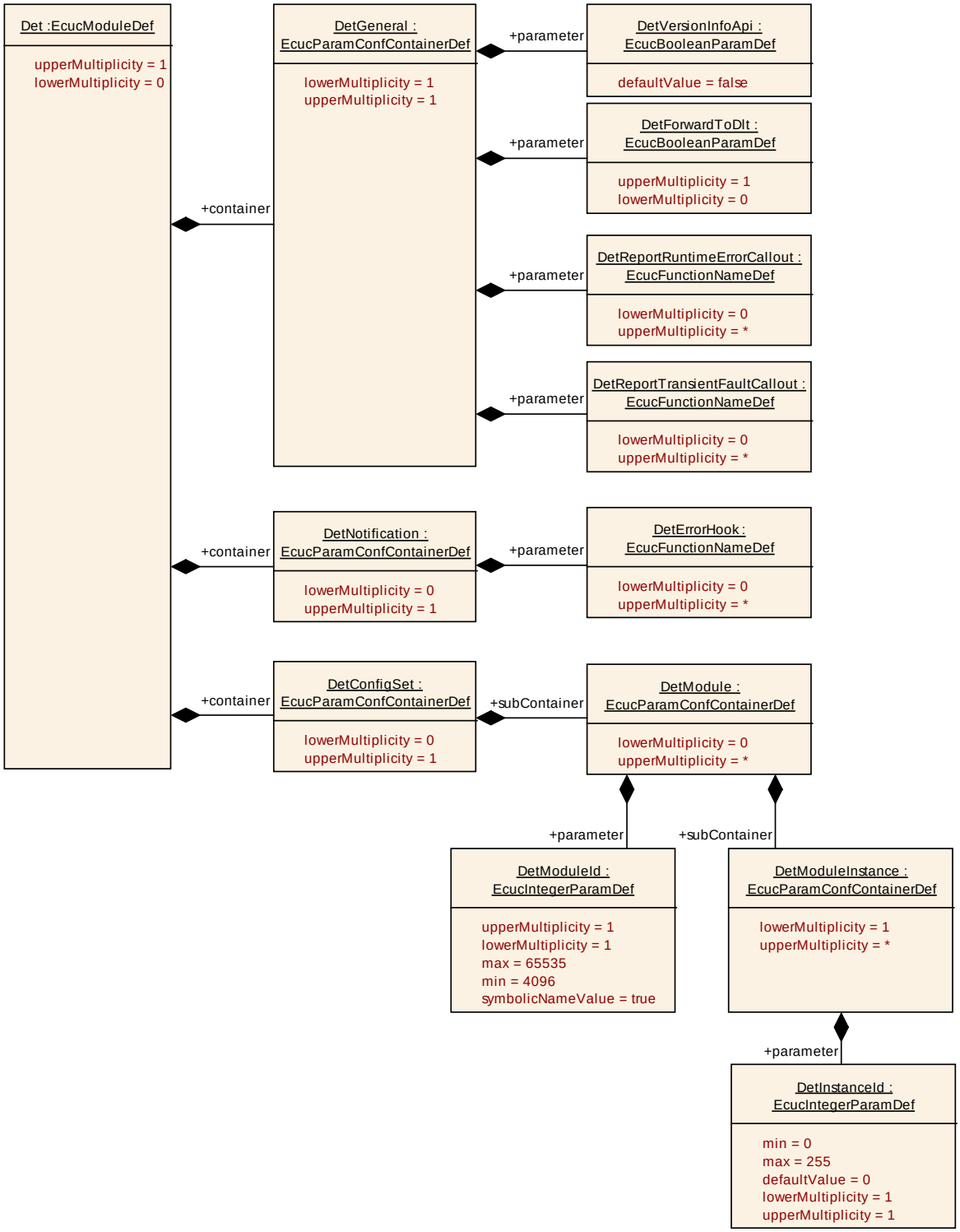

Figure 9: Parameters of DET

10.1.1 Det

SWS Item	**ECUC_Det_00001 :**
Module Name	**Det**
Module Description	Det configuration includes the functions to be called at notification. On one

	side the application functions are specified and in general it can be decided whether Dlt shall be called at each call of Det.
Post-Build Variant Support	false
Supported Config Variants	VARIANT-PRE-COMPILE

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DetConfigSet	0..1	Configuration set container for Det.
DetGeneral	1	Generic configuration parameters of the Det module.
DetNotification	0..1	Configuration of the notification functions.

10.1.2 DetGeneral

SWS Item	**ECUC_Det_00002 :**
Container Name	DetGeneral
Description	Generic configuration parameters of the Det module.
Configuration Parameters	

SWS Item	**ECUC_Det_00006 :**		
Name	DetForwardToDlt		
Description	Only if the parameter is present and set to true, the Det requires the Dlt interface and forwards it's call to the function Dlt_DetForwardErrorTrace. In this case the optional interface to Dlt_Det is required.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	--		
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		

SWS Item	**ECUC_Det_00010 :**		
Name	DetReportRuntimeErrorCallout		
Description	This parameter defines the existence and the names of callout functions for the corresponding runtime error handler. The type of these functions shall be identical the type of Det_ReportRuntimeError itself: Std_ReturnType (*f)(uint16, uint8, uint8, uint8)		
Multiplicity	0..*		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	

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

SWS Item	**ECUC_Det_00011 :**		
Name	DetReportTransientFaultCallout		
Description	This parameter defines the existence and the names of callout functions for the corresponding transient fault handler. The type of these functions shall be identical the type of Det_ReportTransientFault itself: Std_ReturnType (*f)(uint16, uint8, uint8, uint8)		
Multiplicity	0..*		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Det_00003 :**		
Name	DetVersionInfoApi		
Description	Pre-processor switch to enable / disable the API to read out the modules version information. true: Version info API enabled. false: Version info API 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		

No Included Containers

10.1.3 DetNotification

SWS Item	**ECUC_Det_00004 :**
Container Name	DetNotification
Description	Configuration of the notification functions.
Configuration Parameters	

SWS Item	**ECUC_Det_00005 :**
Name	DetErrorHook
Description	Optional list of functions to be called by the Default Error Tracer in context of each call of Det_ReportError. The type of these functions shall be identical the type of Det_ReportError

	itself: Std_ReturnType (*f)(uint16, uint8, uint8, uint8).		
Multiplicity	0..*		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
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		

No Included Containers

10.1.4 DetConfigSet

SWS Item	**ECUC_Det_00007 :**
Container Name	DetConfigSet
Description	Configuration set container for Det.
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DetModule	0..*	This container describes a non BSW module that is using the Det via Service Interface.

10.1.5 DetModule

SWS Item	**ECUC_Det_00008 :**
Container Name	DetModule
Description	This container describes a non BSW module that is using the Det via Service Interface.
Configuration Parameters	

SWS Item	**ECUC_Det_00009 :**	
Name	DetModuleId	
Description	Unique identifier of the error reporting component. When reporting errors to the DET, a symbolic name derived from the moduleID has to be used to identify the reporter.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	4096 .. 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: local		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DetModuleInstance	1..*	Describes the Instance used for the according Service Port. It shall be used to differentiate software component instances when multiple instantiation is used.

10.1.6 DetModuleInstance

SWS Item	**ECUC_Det_00013 :**		
Container Name	DetModuleInstance		
Description	Describes the Instance used for the according Service Port. It shall be used to differentiate software component instances when multiple instantiation is used.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Configuration Parameters			

SWS Item	**ECUC_Det_00012 :**		
Name	DetInstanceId		
Description	Describes the InstanceId used for the according Service Port. It shall be used to differentiate software component instances when multiple instantiation is used. Else it shall be set to 0.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	0		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2 Published Information

Additional module-specific published parameters are listed below if applicable.

11 Not applicable requirements

[SWS_Det_00999] [These requirements are not applicable to this specification.] (SRS_BSW_00301, SRS_BSW_00304, SRS_BSW_00305, SRS_BSW_00306, SRS_BSW_00307, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00439, SRS_BSW_00314, SRS_BSW_00325, SRS_BSW_00328, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00334, SRS_BSW_00335, SRS_BSW_00341, SRS_BSW_00342, SRS_BSW_00343, SRS_BSW_00347, SRS_BSW_00441, SRS_BSW_00353, SRS_BSW_00350, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00440, SRS_BSW_00361, SRS_BSW_00371, SRS_BSW_00373, SRS_BSW_00377, SRS_BSW_00378, SRS_BSW_00379, SRS_BSW_00401, SRS_BSW_00410, SRS_BSW_00413, SRS_BSW_00415, SRS_BSW_00005, SRS_BSW_00006, SRS_BSW_00007, SRS_BSW_00009, SRS_BSW_00010, SRS_BSW_00158, SRS_BSW_00160, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00164, SRS_BSW_00172, SRS_BSW_00344, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00170, SRS_BSW_00380, SRS_BSW_00419, SRS_BSW_00381, SRS_BSW_00412, SRS_BSW_00383, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00393, SRS_BSW_00395, SRS_BSW_00396, SRS_BSW_00397, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00438, SRS_BSW_00375, SRS_BSW_00416, SRS_BSW_00406, SRS_BSW_00437, SRS_BSW_00168, SRS_BSW_00407, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00336, SRS_BSW_00369, SRS_BSW_00339, SRS_BSW_00348, SRS_BSW_00357, SRS_BSW_00422, SRS_BSW_00417, SRS_BSW_00323, SRS_BSW_00004, SRS_BSW_00409, SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00458, SRS_BSW_00466)