Document Title	Specification of Watchdog Interface
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	041
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	• Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2015-07-31	4.2.2	AUTOSAR Release Management	• Minor fixes
2014-10-31	4.2.1	AUTOSAR Release Management	• Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Artifact path fixed
2013-03-15	4.1.1	AUTOSAR Administration	• Reworked according to the new SWS_BSWGeneral • New indexing scheme for requirements
2011-12-22	4.0.3	AUTOSAR Administration	• Modification in DeviceIndex • New template with requirements traceability
2010-09-30	3.1.5	AUTOSAR Administration	• Update of module version check, addition of invalid pointer as error code and checking for null pointer
2010-02-02	3.1.4	AUTOSAR Administration	• Modifications for windowed watchdog concept • Further maintenance for R4.0 • Legal disclaimer revised

Document Change History

Date	Release	Changed by	Change Description
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	• The main bullets summarizing the changes are • Tables of chapter 8 has been replaced with • Contents generated from AUTOSAR BSW model • Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Administration	• In chapter 5.1.2 the file include structure has been changed to comply with the SPAL general include structure. • Legal disclaimer revised • Release Notes added • “Advice for users” revised • “Revision Information” added
2006-05-16	2.0	AUTOSAR Administration	• Document structure adapted to common Release 2.0 SWS Template
2005-05-31	1.0	AUTOSAR Administration	• Initial release

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 functionality, API and the configuration of the AUTOSAR Basic Software module Watchdog Interface.

In case of more than one watchdog device and watchdog driver (e.g. both an internal software watchdog and an external hardware watchdog) being used on an ECU, this module allows the watchdog manager (or any other client of the watchdog) to select the correct watchdog driver - and thus the watchdog device - while retaining the API and functionality of the underlying driver.

The Watchdog Interface is part of the Onboard Device Abstraction Layer (see [1]).

[SWS_WdgIf_00026]⌈ The Watchdog Interface provides uniform access to services of the underlying watchdog drivers like mode switching and setting trigger conditions⌋ (SRS_Wdg_12165, SRS_Wdg_12167, SRS_MemHwAb_14019)

2 Acronyms and abbreviations

Note: For this module there are no local acronyms and abbreviations. All used acronyms and abbreviations should be contained in the AUTOSAR glossary.

3 Related documentation

3.1 Input documents

[1] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[2] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[3] General Requirements on SPAL
AUTOSAR_SRS_SPALGeneral.pdf

[4] Requirements on Memory Hardware Abstraction Layer
AUTOSAR_SRS_MemoryHWAbstractionLayer.pdf

[5] Specification of Watchdog Driver
AUTOSAR_SWS_WatchdogDriver.pdf

[6] Specification of Default Error Tracer
AUTOSAR_SWS_DefaultErrorTracer.pdf

[7] Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[8] AUTOSAR Requirements on Watchdog Driver
AUTOSAR_SRS_WatchdogDriver.pdf

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

3.2 Related standards and norms

None

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [9] (SWS BSW General), which is also valid for Watchdog Interface.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Watchdog Interface.

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

The Watchdog Interface is part of the ECU Abstraction Layer. It allows the upper layer, especially the watchdog manager, to uniformly access one or more watchdog drivers. The implementation of the Watchdog Interface therefore depends on the number of watchdog drivers below.

5.1 File structure

5.1.1 Code file structure

For details refer to the chapter 5.1.6 “Code file structure” in *SWS_BSWGeneral.*

5.1.2 Header file structure

[SWS_WdgIf_00010]⌈ The Watchdog Interface’s implementer shall place the type definitions of the Watchdog Interface in the file WdgIf_Types.h.⌋ (SRS_BSW_00348)

[SWS_WdgIf_00001]⌈ The Watchdog Interface shall comprise a header file “`WdgIf.h`” declaring the API of the Watchdog Interface. If an API is implemented as a macro, it shall be also contained here.⌋ (SRS_BSW_00348)

Note: This is the only header file to be imported by the “user” of the Watchdog Interface.

[SWS_WdgIf_00050]⌈ The Watchdog Interface shall comprise a configuration header file “`WdgIf_Cfg.h`” providing its pre-compile configuration definitions.⌋ (SRS_BSW_00381)

[SWS_WdgIf_00002]⌈ The file include structure shall be as follows:

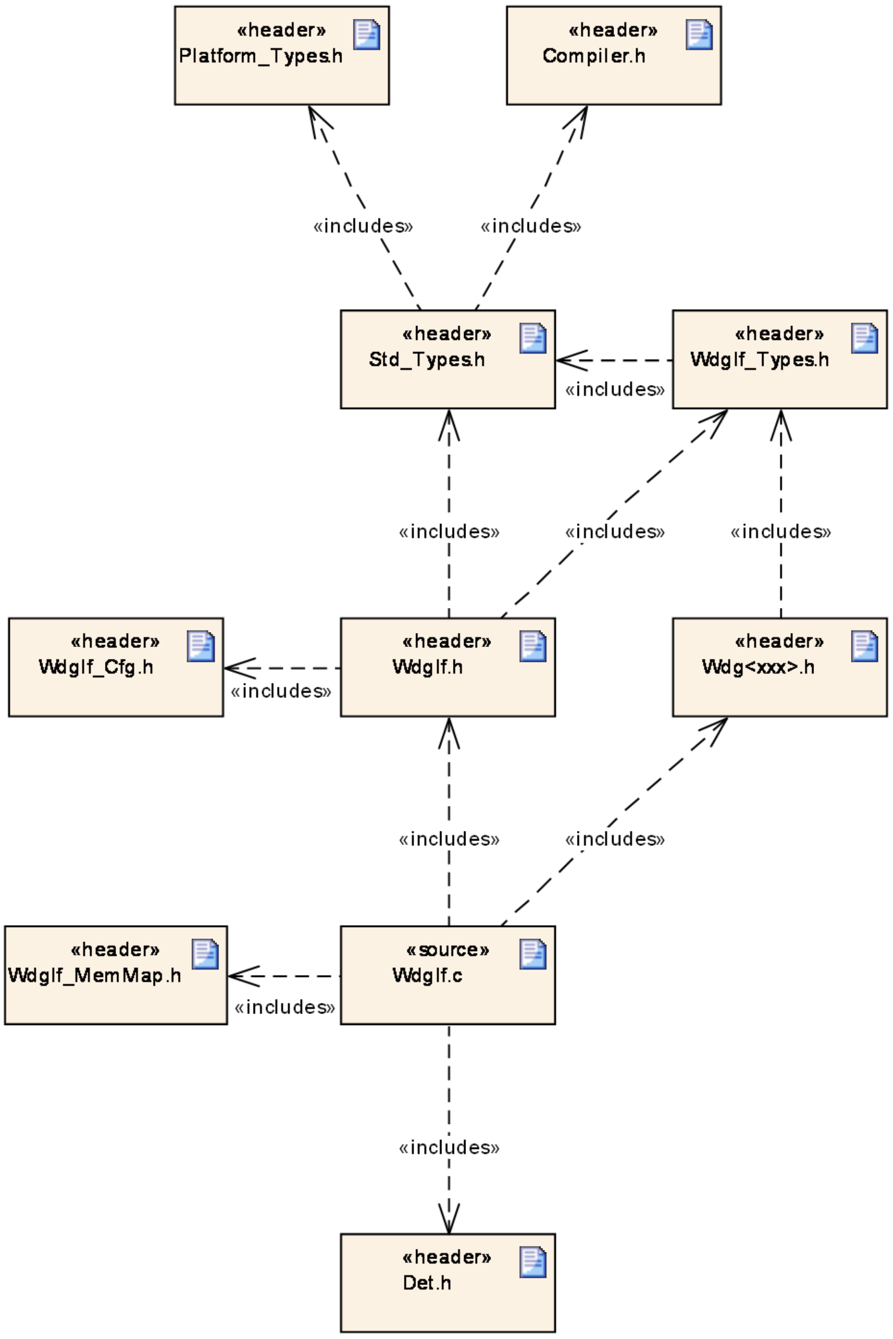

Figure 1: File include structure of the Watchdog Interface

⌋ (SRS_BSW_00348, SRS_BSW_00353, SRS_BSW_00361)

Notes to the figure:

- WdgIf may be a pure macro implementation even in the case of configured default error tracing, which means WdgIf.c may not exist. In this case, Det.h and Wdg<xxx>.h must be included in WdgIf.h instead.
- Wdg<xxx>.h has to be included for the API declaration of the watchdog drivers which, in case of multiple existence, have driver specific “infixes” <xxx> according to SRS_BSW_00374. The figure shows two driver instances as an example.

5.1.3 Version check

For details refer to the chapter 5.1.8 “Version Check” in *SWS_BSWGeneral.*

6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00005	Modules of the µC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_WdgIf_00999
SRS_BSW_00007	All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.	SWS_WdgIf_00999
SRS_BSW_00009	All Basic SW Modules shall be documented according to a common standard.	SWS_WdgIf_00999
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_WdgIf_00999
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_WdgIf_00999
SRS_BSW_00159	All modules of the AUTOSAR Basic Software shall support a tool based configuration	SWS_WdgIf_00999
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_WdgIf_00999
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_WdgIf_00999
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_WdgIf_00999
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_WdgIf_00999
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_WdgIf_00999
SRS_BSW_00300	All AUTOSAR Basic Software Modules shall be identified by an unambiguous name	SWS_WdgIf_00999
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only import the necessary information	SWS_WdgIf_00041
SRS_BSW_00304	All AUTOSAR Basic Software Modules shall use the following data types instead of native C data types	SWS_WdgIf_00013, SWS_WdgIf_00030, SWS_WdgIf_00999
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_WdgIf_00999
SRS_BSW_00307	Global variables naming convention	SWS_WdgIf_00999

SRS_BSW_00308	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	SWS_WdgIf_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_WdgIf_00999
SRS_BSW_00312	Shared code shall be reentrant	SWS_WdgIf_00999
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_WdgIf_00999
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	SWS_WdgIf_00999
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_WdgIf_00028
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_WdgIf_00999
SRS_BSW_00327	Error values naming convention	SWS_WdgIf_00006
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_WdgIf_00999
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_WdgIf_00999
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_WdgIf_00999
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_WdgIf_00999
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_WdgIf_00999
SRS_BSW_00335	Status values naming convention	SWS_WdgIf_00999
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_WdgIf_00999
SRS_BSW_00337	Classification of development errors	SWS_WdgIf_00006
SRS_BSW_00339	Reporting of production relevant error status	SWS_WdgIf_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_WdgIf_00999
SRS_BSW_00343	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	SWS_WdgIf_00999
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_WdgIf_00999
SRS_BSW_00347	A Naming seperation of different	SWS_WdgIf_00999

	instances of BSW drivers shall be in place	
SRS_BSW_00348	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	SWS_WdgIf_00001, SWS_WdgIf_00002, SWS_WdgIf_00010
SRS_BSW_00353	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	SWS_WdgIf_00002
SRS_BSW_00357	For success/failure of an API call a standard return type shall be defined	SWS_WdgIf_00046
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_WdgIf_00999
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_WdgIf_00999
SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_WdgIf_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_WdgIf_00002
SRS_BSW_00369	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	SWS_WdgIf_00058
SRS_BSW_00371	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules	SWS_WdgIf_00999
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	SWS_WdgIf_00999
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_WdgIf_00999
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_WdgIf_00999
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_WdgIf_00999
SRS_BSW_00380	Configuration parameters being stored in memory shall be placed into separate c-files	SWS_WdgIf_00999
SRS_BSW_00381	The pre-compile time parameters shall be placed into a separate configuration header file	SWS_WdgIf_00050
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_WdgIf_00999
SRS_BSW_00384	The Basic Software Module specifications shall specify at least in the description which other modules they require	SWS_WdgIf_00047, SWS_WdgIf_00048

SRS_BSW_00385	List possible error notifications	SWS_WdgIf_00006
SRS_BSW_00386	The BSW shall specify the configuration for detecting an error	SWS_WdgIf_00006
SRS_BSW_00398	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	SWS_WdgIf_00999
SRS_BSW_00399	Parameter-sets shall be located in a separate segment and shall be loaded after the code	SWS_WdgIf_00999
SRS_BSW_00400	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	SWS_WdgIf_00999
SRS_BSW_00401	Documentation of multiple instances of configuration parameters shall be available	SWS_WdgIf_00999
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_WdgIf_00999
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_WdgIf_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_WdgIf_00999
SRS_BSW_00412	References to c-configuration parameters shall be placed into a separate h-file	SWS_WdgIf_00999
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_WdgIf_00999
SRS_BSW_00414	Init functions shall have a pointer to a configuration structure as single parameter	SWS_WdgIf_00006, SWS_WdgIf_00999
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_WdgIf_00999
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_WdgIf_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_WdgIf_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_WdgIf_00999
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_WdgIf_00999
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_WdgIf_00999
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_WdgIf_00999
SRS_BSW_00425	The BSW module description template shall provide means to model the defined	SWS_WdgIf_00999

	trigger conditions of schedulable objects	
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_WdgIf_00999
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_WdgIf_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_WdgIf_00999
SRS_BSW_00429	BSW modules shall be only allowed to use OS objects and/or related OS services	SWS_WdgIf_00999
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_WdgIf_00999
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_WdgIf_00999
SRS_BSW_00437	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	SWS_WdgIf_00999
SRS_BSW_00438	Configuration data shall be defined in a structure	SWS_WdgIf_00999
SRS_BSW_00439	Enable BSW modules to handle interrupts	SWS_WdgIf_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_WdgIf_00999
SRS_BSW_00441	Naming convention for type, macro and function	SWS_WdgIf_00999
SRS_BSW_00447	Standardizing Include file structure of BSW Modules Implementing Autosar Service	SWS_WdgIf_00999
SRS_BSW_00449	BSW Service APIs used by Autosar Application Software shall return a Std_ReturnType	SWS_WdgIf_00999
SRS_BSW_00450	A Main function of a un-initialized module shall return immediately	SWS_WdgIf_00999
SRS_MemHwAb_14019	The Memory Abstraction Interface shall provide uniform access to the API services of the underlying memory abstraction modules	SWS_WdgIf_00017, SWS_WdgIf_00026
SRS_MemHwAb_14020	The Memory Abstraction Interface shall allow the selection of an underlying memory abstraction module by using a device index	SWS_WdgIf_00018
SRS_MemHwAb_14021	The Memory Abstraction Interface shall allow the pre-compile time configuration of the number of underlying memory abstraction modules	SWS_WdgIf_00019, SWS_WdgIf_00020

SRS_MemHwAb_14022	The Memory Abstraction Interface shall preserve the functionality of the underlying memory abstraction module	SWS_WdgIf_00003
SRS_MemHwAb_14023	The Memory Abstraction Interface shall only check those parameters that are used within the interface itself	SWS_WdgIf_00028
SRS_MemHwAb_14024	The Memory Abstraction Interface shall preserve the timing behavior of the underlying memory abstraction modules and their APIs	SWS_WdgIf_00003
SRS_SPAL_12448	All driver modules shall have a specific behavior after a development error detection	SWS_WdgIf_00028
SRS_Wdg_12018	The watchdog driver shall provide a service for selecting the watchdog mode	SWS_WdgIf_00016, SWS_WdgIf_00042, SWS_WdgIf_00057, SWS_WdgIf_00061
SRS_Wdg_12165	For an external watchdog driver the same requirements shall apply like for an internal watchdog driver	SWS_WdgIf_00017, SWS_WdgIf_00026
SRS_Wdg_12167	The external watchdog driver shall have a semantically identical API as an internal watchdog driver	SWS_WdgIf_00017, SWS_WdgIf_00026
SRS_Wdg_13500	The watchdog driver shall provide a service to set the watchdog trigger condition	SWS_WdgIf_00044
SWS_BSW_00212	NULL pointer checking	SWS_WdgIf_00006

7 Functional specification

7.1 General behavior

[SWS_WdgIf_00003]⌈ The Watchdog Interface shall not add functionality to the watchdog drivers. Also the Watchdog Interface does not abstract from watchdog properties like toggle or window mode, timeout periods etc. that is it does not hide any features of the underlying watchdog driver and watchdog hardware.⌋ (SRS_MemHwAb_14022, SRS_MemHwAb_14024)

7.2 Error classification

7.2.1 Development Errors

[SWS_WdgIf_00006]⌈ Development Error Types

Type or error	***Related error code***	***Value [hex]***
API service called with wrong device index parameter	`WDGIF_E_PARAM_DEVICE`	`0x01`
Invalid pointer in parameter list	`WDGIF_E_INV_POINTER`	`0x02`
NULL pointer checking	`WDGIF_E_PARAM_POINTER`	`0x03`

.⌋ (SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00327, SRS_BSW_00414, SWS_BSW_00212)

[SWS_WdgIf_00030]⌈ Development error values are of type
uint8.⌋ (SRS_BSW_00304)

[SWS_WdgIf_00028]⌈ If more than one watchdog driver is configured and the default error detection is enabled for this module, the parameter `DeviceIndex` shall be checked for being an existing device within the module's services**.** Detected errors shall be reported to the Default Error Tracer (DET) with the error code `WDGIF_E_PARAM_DEVICE` and the called service shall not be executed. If the called function has a return value this value shall be set `E_NOT_OK`.⌋ (SRS_BSW_00323, SRS_SPAL_12448, SRS_MemHwAb_14023)

7.2.2 Runtime Errors

There are no runtime errors.

7.2.3 Transient Faults

There are no transient faults.

7.2.4 Production Errors

There are no production errors.

7.2.5 Extended Production Errors

There are no extended production errors.

8 API specification

8.1 Imported types

In this chapter all types included from the following files are listed:

[SWS_WdgIf_00041] ⌈

Module	***Imported Type***
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋ (SRS_BSW_00301)

8.2 Type definitions

Note: The implementer of the Watchdog Interface shall not change or extend the type definitions of the Watchdog Interface for a specific watchdog device or platform.

8.2.1 WdgIf_ModeType

[SWS_WdgIf_00061] ⌈

Name:	WdgIf_ModeType		
Type:	Enumeration		
Range:	WDGIF_OFF_MODE	--	In this mode, the watchdog driver is disabled (switched off).
	WDGIF_SLOW_MODE	--	In this mode, the watchdog driver is set up for a long timeout period (slow triggering).
	WDGIF_FAST_MODE	--	In this mode, the watchdog driver is set up for a short timeout period (fast triggering).
Description:	Mode type of the WdgIf module		

⌋ (SRS_Wdg_12018)

[SWS_WdgIf_00016]⌈ The `WdgIf_ModeType` values shall be passed as parameters to the watchdog drivers mode switching function (`Wdg_SetMode`).⌋ (SRS_Wdg_12018)

Note: The hardware specific settings behind these modes are given in the watchdog drivers configuration set.

8.3 Function definitions

[SWS_WdgIf_00017]⌈ The Watchdog Interface shall map the APIs specified in this chapter to the API of the underlying drivers. For functional behavior refer to the specification of the watchdog driver⌋ (SRS_Wdg_12165, SRS_Wdg_12167, SRS_MemHwAb_14019)

[SWS_WdgIf_00018]⌈ The Watchdog Interface shall use the parameter `DeviceIndex` for selection of watchdog drivers. If only one watchdog driver is configured, the parameter `DeviceIndex` shall be ignored.⌋ (SRS_MemHwAb_14020)

[SWS_WdgIf_00013]⌈ The data type for the watchdog device index shall be `uint8`.DeviceIndex shall provide a zero-based consecutive index.⌋ (SRS_BSW_00304)

[SWS_WdgIf_00019]⌈ If only one watchdog driver is configured, the Watchdog Interface shall cause no runtime overhead when mapping the Watchdog Interface API to the API of the corresponding Watchdog Driver.⌋ (SRS_MemHwAb_14021)

Implementation hint: This could be done by using macros as for example

```
#define WdgIf_SetMode(DeviceIndex, WdgMode) \
      Wdg_SetMode(WdgMode)
```

[SWS_WdgIf_00020]⌈ If more than one watchdog driver is configured, the Watchdog Interface shall use efficient mechanisms to map the API calls to the appropriate watchdog driver.⌋ (SRS_MemHwAb_14021)

Implementation hint: One solution is to use tables of pointers to functions where the parameter `DeviceIndex` is used as array index, for example

```
#define WdgIf_SetMode(DeviceIndex, WdgMode) \
  SetModeFctPtr[DeviceIndex](WdgMode)
```

Note: The service IDs are related to the service IDs of the watchdog driver specification (see [5]). For that reason, they may not start with 0.

8.3.1 WdgIf_SetMode

[SWS_WdgIf_00042] ⌈

Service name:	WdgIf_SetMode	

Syntax:	`Std_ReturnType WdgIf_SetMode(`
	`    uint8 DeviceIndex,`
	`    WdgIf_ModeType WdgMode`
	`)`	

Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DeviceIndex	Identifies the Watchdog Driver instance.
	WdgMode	The watchdog driver mode (see Watchdog Driver).
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	--
Description:	Map the service WdgIf_SetMode to the service Wdg_SetMode of the corresponding Watchdog Driver.	

⌋ (SRS_Wdg_12018)

[SWS_WdgIf_00057]⌈ `WdgIf_SetMode` shall return the value which it gets from the service `Wdg_SetMode` of the corresponding Watchdog Driver.⌋ (SRS_Wdg_12018)

Possible content of the return value is specified by the Watchdog Driver, see [5].

8.3.2 WdgIf_SetTriggerCondition

[SWS_WdgIf_00044] ⌈

Service name:	WdgIf_SetTriggerCondition	
Syntax:	`void WdgIf_SetTriggerCondition( uint8 DeviceIndex, uint16 Timeout )`	
Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DeviceIndex	Identifies the Watchdog Driver instance.
	Timeout	Timeout value (milliseconds) for setting the trigger counter.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Map the service WdgIf_SetTriggerCondition to the service Wdg_SetTriggerCondition of the corresponding Watchdog Driver.	

⌋ (SRS_Wdg_13500)

8.3.3 WdgIf_GetVersionInfo

[SWS_WdgIf_00046] ⌈

Service name:	WdgIf_GetVersionInfo	
Syntax:	`void WdgIf_GetVersionInfo( Std_VersionInfoType* VersionInfoPtr )`	
Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	VersionInfoPtr	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Returns the version information.	

⌋ (SRS_BSW_00357)

[SWS_WdgIf_00058]⌈ If default error detection for the Watchdog Interface module is enabled, then the function WdgIf_GetVersionInfo shall check whether the parameter VersioninfoPtr is a NULL pointer (NULL_PTR). If VersioninfoPtr is a NULL pointer, then the function WdgIf_GetVersionInfo shall raise the development error WDGIF_E_INV_POINTER (i.e. invalid pointer) and return.⌋ (SRS_BSW_00369)

8.4 Call-back notifications

This module does not provide any callback functions.

8.5 Scheduled functions

This module does not need any scheduled functions.

8.6 Expected interfaces

In this chapter all interfaces required from other modules are listed.

8.6.1 Mandatory interfaces

This chapter defines all interfaces which are required to fulfill the core functionality of the module.

[SWS_WdgIf_00047] ⌈

API function	***Description***
Wdg_SetMode	Switches the watchdog into the mode Mode.
Wdg_SetTriggerCondition	Sets the timeout value for the trigger counter.

⌋ (SRS_BSW_00384)

8.6.2 Optional interfaces

This chapter defines all interfaces which are required to fulfill an optional functionality of the module.

[SWS_WdgIf_00048] ⌈

API function	***Description***
Det_ReportError	Service to report development errors.

⌋ (SRS_BSW_00384)

8.6.3 Configurable interfaces

There are no configurable interfaces for this module.

9 Sequence diagrams

Refer to specification of watchdog driver [5].

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module WdgIf.

Chapter 10.3 specifies published information of the module WdgIf.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in *SWS_BSWGeneral.*

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters are described in chapters 7 and 8.

10.2.1 WdgIf

SWS Item	**ECUC_WdgIf_00033 :**
Module Name	*WdgIf*
Module Description	Configuration of the WdgIf (Watchdog Interface) module.
Post-Build Variant Support	false
Supported Config Variants	VARIANT-PRE-COMPILE

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
WdgIfDevice	1..*	It contains the information for the selection of a particular Watchdog device in case multiple Watchdog drivers are connected.
WdgIfGeneral	1	This container collects all generic watchdog interface parameters.

10.2.2 WdgIfGeneral

SWS Item	**ECUC_WdgIf_00001 :**
Container Name	WdgIfGeneral
Description	This container collects all generic watchdog interface parameters.
Configuration Parameters	

SWS Item	**ECUC_WdgIf_00005 :**		
Name	WdgIfDevErrorDetect		
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		

SWS Item	**ECUC_WdgIf_00003 :**
Name	WdgIfVersionInfoApi
Description	Pre-processor switch to enable / disable the service returning the version information. true: Version information service enabled false: Version information service 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

10.2.3 WdgIfDevice

SWS Item	**ECUC_WdgIf_00002 :**
Container Name	WdgIfDevice
Description	It contains the information for the selection of a particular Watchdog device in case multiple Watchdog drivers are connected.
Configuration Parameters	

SWS Item	**ECUC_WdgIf_00006 :**		
Name	WdgIfDeviceIndex		
Description	Represents the watchdog interface ID so that it can be referenced by the watchdog manager.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
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		

SWS Item	**ECUC_WdgIf_00007 :**		
Name	WdgIfDriverRef		
Description	Reference to the watchdog drivers that are controlled by the watchdog interface.		
Multiplicity	1		
Type	Symbolic name reference to [WdgGeneral]		
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

10.3 Published parameters

For details refer to the chapter 10.3 “Published Information” in *SWS_BSWGeneral.*

11 Not applicable requirements

[SWS_WdgIf_00999]⌈ These requirements are not applicable to this specification.⌋ (SRS_BSW_00344, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00159, SRS_BSW_00170, SRS_BSW_00380, SRS_BSW_00419, SRS_BSW_00412, SRS_BSW_00383, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00438, SRS_BSW_00375, SRS_BSW_00101, SRS_BSW_00416, SRS_BSW_00406, SRS_BSW_00437, SRS_BSW_00168, 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_00450, SRS_BSW_00336, SRS_BSW_00339, SRS_BSW_00422, SRS_BSW_00417, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00415, SRS_BSW_00164, SRS_BSW_00325, SRS_BSW_00342, SRS_BSW_00343, SRS_BSW_00007, SRS_BSW_00300, SRS_BSW_00413, SRS_BSW_00347, SRS_BSW_00441, SRS_BSW_00307, SRS_BSW_00373, SRS_BSW_00335, SRS_BSW_00314, SRS_BSW_00447, SRS_BSW_00328, SRS_BSW_00312, SRS_BSW_00439, SRS_BSW_00449, SRS_BSW_00377, SRS_BSW_00304, SRS_BSW_00378, SRS_BSW_00306, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00371, SRS_BSW_00358, SRS_BSW_00414, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00440, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00009, SRS_BSW_00401, SRS_BSW_00010, SRS_BSW_00333, SRS_BSW_00321, SRS_BSW_00334)