### Document Title

**Specification of Watchdog Interface**

<table>
<thead>
<tr>
<th>Document Owner</th>
<th>AUTOSAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Responsibility</td>
<td>AUTOSAR</td>
</tr>
<tr>
<td>Document Identification No</td>
<td>041</td>
</tr>
<tr>
<td>Document Status</td>
<td>Final</td>
</tr>
<tr>
<td>Part of AUTOSAR Standard</td>
<td>Classic Platform</td>
</tr>
<tr>
<td>Part of Standard Release</td>
<td>4.3.1</td>
</tr>
</tbody>
</table>

### Document Change History

<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Changed by</th>
<th>Change Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-12-08</td>
<td>4.3.1</td>
<td>AUTOSAR Release Management</td>
<td>• Renaming – default errors changed to development errors.</td>
</tr>
<tr>
<td>2016-11-30</td>
<td>4.3.0</td>
<td>AUTOSAR Release Management</td>
<td>• Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation</td>
</tr>
<tr>
<td>2015-07-31</td>
<td>4.2.2</td>
<td>AUTOSAR Release Management</td>
<td>• Minor fixes</td>
</tr>
<tr>
<td>2014-10-31</td>
<td>4.2.1</td>
<td>AUTOSAR Release Management</td>
<td>• Editorial changes</td>
</tr>
</tbody>
</table>
| 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                                            |
<p>| 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 |</p>
<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Changed by</th>
<th>Change Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010-02-02</td>
<td>3.1.4</td>
<td>AUTOSAR Administration</td>
<td>• Modifications for windowed watchdog concept</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Further maintenance for R4.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Legal disclaimer revised</td>
</tr>
<tr>
<td>2008-08-13</td>
<td>3.1.1</td>
<td>AUTOSAR Administration</td>
<td>• Legal disclaimer revised</td>
</tr>
<tr>
<td>2007-12-21</td>
<td>3.0.1</td>
<td>AUTOSAR Administration</td>
<td>• The main bullets summarizing the changes are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Tables of chapter 8 has been replaced with</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Contents generated from AUTOSAR BSW model</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Document meta information extended</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Small layout adaptations made</td>
</tr>
<tr>
<td>2007-01-24</td>
<td>2.1.15</td>
<td>AUTOSAR Administration</td>
<td>• In chapter 5.1.2 the file include structure has been changed to comply with the SPAL general include structure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Legal disclaimer revised</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Release Notes added</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• “Advice for users” revised</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• “Revision Information” added</td>
</tr>
<tr>
<td>2006-05-16</td>
<td>2.0</td>
<td>AUTOSAR Administration</td>
<td>• Document structure adapted to common Release 2.0 SWS Template</td>
</tr>
<tr>
<td>2005-05-31</td>
<td>1.0</td>
<td>AUTOSAR Administration</td>
<td>• Initial release</td>
</tr>
</tbody>
</table>
Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.
# Table of Contents

1. Introduction and functional overview ............................................................... 6
2. Acronyms and abbreviations .............................................................................. 7
3. Related documentation ....................................................................................... 8
    3.1 Input documents .......................................................................................... 8
    3.2 Related standards and norms ..................................................................... 8
    3.3 Related specification .................................................................................. 8
4. Constraints and assumptions ............................................................................ 9
    4.1 Limitations .................................................................................................. 9
    4.2 Applicability to car domains ....................................................................... 9
5. Dependencies to other modules ....................................................................... 10
    5.1 File structure ............................................................................................. 10
        5.1.1 Code file structure ............................................................................ 10
        5.1.2 Header file structure ......................................................................... 10
        5.1.3 Version check .................................................................................... 12
6. Requirements traceability ................................................................................ 13
7. Functional specification ................................................................................... 19
    7.1 General behavior ......................................................................................... 19
    7.2 Error classification ..................................................................................... 19
        7.2.1 Development Errors .......................................................................... 19
        7.2.2 Runtime Errors .................................................................................. 19
        7.2.3 Transient Faults ............................................................................... 19
        7.2.4 Production Errors .............................................................................. 20
        7.2.5 Extended Production Errors ................................................................. 20
8. API specification ............................................................................................... 21
    8.1 Imported types .............................................................................................. 21
    8.2 Type definitions ........................................................................................... 21
        8.2.1 WdgIf_ModeType .............................................................................. 21
    8.3 Function definitions ..................................................................................... 21
        8.3.1 WdgIf_SetMode ............................................................................... 22
        8.3.2 WdgIf_SetTriggerCondition ............................................................... 23
        8.3.3 WdgIf_GetVersionInfo ...................................................................... 23
7.4 Call-back notifications .................................................................................. 23
8.5 Scheduled functions ....................................................................................... 24
8.6 Expected interfaces ....................................................................................... 24
    8.6.1 Mandatory interfaces ............................................................................ 24
    8.6.2 Optional interfaces ................................................................................ 24
    8.6.3 Configurable interfaces .......................................................................... 24
9. Sequence diagrams ......................................................................................... 25
10. Configuration specification ............................................................................ 26
    10.1 How to read this chapter .......................................................................... 26
    10.2 Containers and configuration parameters ............................................... 27
10.2.1 WdgIIf ................................................................................................................... 27
10.2.2 WdgIfGeneral ......................................................................................................... 27
10.2.3 WdgIfDevice ........................................................................................................... 28
10.3 Published parameters ............................................................................................... 29
11 Not applicable requirements ....................................................................................... 30
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

   AUTOSAR_SRS_BSWGeneral.pdf

   AUTOSAR_SRS_SPALGeneral.pdf

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

   AUTOSAR_SWS_WatchdogDriver.pdf

   AUTOSAR_SWS_DefaultErrorTracer.pdf

[7] Basic Software Module Description Template
   AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[8] AUTOSAR Requirements on Watchdog Driver
   AUTOSAR_SRS_WatchdogDriver.pdf

   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_Wdglf_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_Wdglf_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_Wdglf_00050] The Watchdog Interface shall comprise a configuration header file “WdgIf_Cfg.h” providing its pre-compile configuration definitions. (SRS_BSW_00381)

[SWS_Wdglf_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 development 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

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
<th>Satisfied by</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRS_BSW_00005</td>
<td>Modules of the μC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00007</td>
<td>All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00009</td>
<td>All Basic SW Modules shall be documented according to a common standard.</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00010</td>
<td>The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00101</td>
<td>The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00159</td>
<td>All modules of the AUTOSAR Basic Software shall support a tool based configuration</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00161</td>
<td>The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00162</td>
<td>The AUTOSAR Basic Software shall provide a hardware abstraction layer</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00164</td>
<td>The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00168</td>
<td>SW components shall be tested by a function defined in a common API in the Basis-SW</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00170</td>
<td>The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00300</td>
<td>All AUTOSAR Basic Software Modules shall be identified by an unambiguous name</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00301</td>
<td>All AUTOSAR Basic Software Modules shall only import the necessary information</td>
<td>SWS_WdgIf_00041</td>
</tr>
<tr>
<td>SRS_BSW_00304</td>
<td>All AUTOSAR Basic Software Modules shall use the following data types instead of native C data types</td>
<td>SWS_WdgIf_00013,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SWS_WdgIf_00030,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00306</td>
<td>AUTOSAR Basic Software Modules shall be compiler and platform independent</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00307</td>
<td>Global variables naming convention</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00308</td>
<td>AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>SRS_BSW_00309</td>
<td>All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00312</td>
<td>Shared code shall be reentrant</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00314</td>
<td>All internal driver modules shall separate the interrupt frame definition from the service routine</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00321</td>
<td>The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00323</td>
<td>All AUTOSAR Basic Software Modules shall check passed API parameters for validity</td>
<td>SWS_WdgIf_00028</td>
</tr>
<tr>
<td>SRS_BSW_00325</td>
<td>The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00327</td>
<td>Error values naming convention</td>
<td>SWS_WdgIf_00006</td>
</tr>
<tr>
<td>SRS_BSW_00328</td>
<td>All AUTOSAR Basic Software Modules shall avoid the duplication of code</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00330</td>
<td>It shall be allowed to use macros instead of functions where source code is used and runtime is critical</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00331</td>
<td>All Basic Software Modules shall strictly separate error and status information</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00333</td>
<td>For each callback function it shall be specified if it is called from interrupt context or not</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00334</td>
<td>All Basic Software Modules shall provide an XML file that contains the meta data</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00335</td>
<td>Status values naming convention</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00336</td>
<td>Basic SW module shall be able to shutdown</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00337</td>
<td>Classification of development errors</td>
<td>SWS_WdgIf_00006</td>
</tr>
<tr>
<td>SRS_BSW_00339</td>
<td>Reporting of production relevant error status</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00342</td>
<td>It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00343</td>
<td>The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00344</td>
<td>BSW Modules shall support link-time configuration</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00347</td>
<td>A Naming separation of different</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>Specification</td>
<td>Instances of BSW drivers shall be in place</td>
<td>SRS_BSW_00348</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>All AUTOSAR standard types and constants shall be placed and organized in a standard type header file</td>
<td>SWS_WdgIf_00001, SWS_WdgIf_00002, SWS_WdgIf_00010</td>
</tr>
<tr>
<td></td>
<td>All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header</td>
<td>SWS_WdgIf_00002</td>
</tr>
<tr>
<td></td>
<td>For success/failure of an API call a standard return type shall be defined</td>
<td>SWS_WdgIf_00046</td>
</tr>
<tr>
<td></td>
<td>The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td></td>
<td>All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td></td>
<td>AUTOASR Basic Software Modules callback functions are allowed to have parameters</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td></td>
<td>All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header</td>
<td>SWS_WdgIf_00002</td>
</tr>
<tr>
<td></td>
<td>All AUTOSAR Basic Software Modules shall not return specific development error codes via the API</td>
<td>SWS_WdgIf_00058</td>
</tr>
<tr>
<td></td>
<td>The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td></td>
<td>The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td></td>
<td>Basic Software Modules shall report wake-up reasons</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td></td>
<td>A Basic Software Module can return a module specific types</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td></td>
<td>AUTOSAR shall provide a boolean type</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td></td>
<td>Configuration parameters being stored in memory shall be placed into separate c-files</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td></td>
<td>The pre-compile time parameters shall be placed into a separate configuration header file</td>
<td>SWS_WdgIf_00050</td>
</tr>
<tr>
<td></td>
<td>The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td></td>
<td>The Basic Software Module specifications shall specify at least in the description which other modules they require</td>
<td>SWS_WdgIf_00047, SWS_WdgIf_00048</td>
</tr>
<tr>
<td>Specification ID</td>
<td>Description</td>
<td>Related SWS_WdgIf IDs</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>SRS_BSW_00385</td>
<td>List possible error notifications</td>
<td>SWS_WdgIf_00006</td>
</tr>
<tr>
<td>SRS_BSW_00386</td>
<td>The BSW shall specify the configuration for detecting an error</td>
<td>SWS_WdgIf_00006</td>
</tr>
<tr>
<td>SRS_BSW_00398</td>
<td>The link-time configuration is achieved on object code basis in the stage after compiling and before linking</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00399</td>
<td>Parameter-sets shall be located in a separate segment and shall be loaded after the code</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00400</td>
<td>Parameter shall be selected from multiple sets of parameters after code has been loaded and started</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00401</td>
<td>Documentation of multiple instances of configuration parameters shall be available</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00404</td>
<td>BSW Modules shall support post-build configuration</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00405</td>
<td>BSW Modules shall support multiple configuration sets</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00406</td>
<td>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</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00412</td>
<td>References to configuration parameters shall be placed into a separate h-file</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00413</td>
<td>An index-based accessing of the instances of BSW modules shall be done</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00414</td>
<td>Init functions shall have a pointer to a configuration structure as single parameter</td>
<td>SWS_WdgIf_00006, SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00415</td>
<td>Interfaces which are provided exclusively for one module shall be separated into a dedicated header file</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00416</td>
<td>The sequence of modules to be initialized shall be configurable</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00417</td>
<td>Software which is not part of the SW-C shall report error events only after the DEM is fully operational.</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00419</td>
<td>If a pre-compile time configuration parameter is implemented as &quot;const&quot; it should be placed into a separate c-file</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00422</td>
<td>Pre-de-bouncing of error status information is done within the DEM</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00423</td>
<td>BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00424</td>
<td>BSW module main processing functions shall not be allowed to enter a wait state</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>SRS_BSW_00425</td>
<td>The BSW module description template shall provide means to model the defined</td>
<td>SWS_WdgIf_00999</td>
</tr>
<tr>
<td>Specification</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00426</td>
<td>BSW Modules shall ensure data consistency of data which is shared between BSW modules</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00427</td>
<td>ISR functions shall be defined and documented in the BSW module description template</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00428</td>
<td>A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00429</td>
<td>Access to OS is restricted</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00432</td>
<td>Modules should have separate main processing functions for read/receive and write/transmit data path</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00433</td>
<td>Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00437</td>
<td>Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00438</td>
<td>Configuration data shall be defined in a structure</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00439</td>
<td>Enable BSW modules to handle interrupts</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00440</td>
<td>The callback function invocation by the BSW module shall follow the signature provided by RTE to invoke servers via Rte_Call API</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00441</td>
<td>Naming convention for type, macro and function</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00447</td>
<td>Standardizing Include file structure of BSW Modules Implementing Autosar Service</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00449</td>
<td>BSW Service APIs used by Autosar Application Software shall return a Std_ReturnType</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00450</td>
<td>A Main function of a un-initialized module shall return immediately</td>
<td></td>
</tr>
<tr>
<td>SRS_MemHwAb_14019</td>
<td>The Memory Abstraction Interface shall provide uniform access to the API services of the underlying memory abstraction modules</td>
<td></td>
</tr>
<tr>
<td>SRS_MemHwAb_14020</td>
<td>The Memory Abstraction Interface shall allow the selection of an underlying memory abstraction module by using a device index</td>
<td></td>
</tr>
<tr>
<td>SRS_MemHwAb_14021</td>
<td>The Memory Abstraction Interface shall allow the pre-compile time configuration of the number of underlying memory abstraction modules</td>
<td></td>
</tr>
</tbody>
</table>
| SRS_MemHwAb_14022 | The Memory Abstraction Interface shall }
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
<th>Test Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRS_MemHwAb_14023</td>
<td>The Memory Abstraction Interface shall only check those parameters that are used within the interface itself</td>
<td>SWS_WdgIf_00028</td>
</tr>
<tr>
<td>SRS_MemHwAb_14024</td>
<td>The Memory Abstraction Interface shall preserve the timing behavior of the underlying memory abstraction modules and their APIs</td>
<td>SWS_WdgIf_00003</td>
</tr>
<tr>
<td>SRS_SPAL_12448</td>
<td>All driver modules shall have a specific behavior after a development error detection</td>
<td>SWS_WdgIf_00028</td>
</tr>
<tr>
<td>SRS_Wdg_12018</td>
<td>The watchdog driver shall provide a service for selecting the watchdog mode</td>
<td>SWS_WdgIf_00016, SWS_WdgIf_00042, SWS_WdgIf_00057, SWS_WdgIf_00061</td>
</tr>
<tr>
<td>SRS_Wdg_12165</td>
<td>For an external watchdog driver the same requirements shall apply like for an internal watchdog driver</td>
<td>SWS_WdgIf_00017, SWS_WdgIf_00026</td>
</tr>
<tr>
<td>SRS_Wdg_12167</td>
<td>The external watchdog driver shall have a semantically identical API as an internal watchdog driver</td>
<td>SWS_WdgIf_00017, SWS_WdgIf_00026</td>
</tr>
<tr>
<td>SRS_Wdg_13500</td>
<td>The watchdog driver shall provide a service to set the watchdog trigger condition</td>
<td>SWS_WdgIf_00044</td>
</tr>
<tr>
<td>SWS_BSW_00212</td>
<td>NULL pointer checking</td>
<td>SWS_WdgIf_00006</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Type or error</th>
<th>Related error code</th>
<th>Value [hex]</th>
</tr>
</thead>
<tbody>
<tr>
<td>API service called with wrong device index parameter</td>
<td>WDGIF_E_PARAM_DEVICE</td>
<td>0x01</td>
</tr>
<tr>
<td>Invalid pointer in parameter list</td>
<td>WDGIF_E_INV_POINTER</td>
<td>0x02</td>
</tr>
<tr>
<td>NULL pointer checking</td>
<td>WDGIF_E_PARAM_POINTER</td>
<td>0x03</td>
</tr>
</tbody>
</table>

(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 development 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]

<table>
<thead>
<tr>
<th>Module</th>
<th>Imported Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Std_Types</td>
<td>Std_ReturnType</td>
</tr>
<tr>
<td></td>
<td>Std_VersionInfoType</td>
</tr>
</tbody>
</table>

(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]

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

(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_Wdglf_00013] The data type for the watchdog device index shall be uint8. DeviceIndex shall provide a zero-based consecutive index.
[SRS_BSW_00304]

[SWS_Wdglf_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 Wdglf_SetMode(DeviceIndex, WdgMode) \n  Wdg_SetMode(WdgMode)

[SWS_Wdglf_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 Wdglf_SetMode(DeviceIndex, WdgMode) \n  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 Wdglf_SetMode

[SWS_Wdglf_00042] |
| Service name: | Wdglf_SetMode |
| Syntax: | Std_ReturnType Wdglf_SetMode( |
| |  uint8 DeviceIndex, |
| |  Wdglf_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 Wdglf_SetMode to the service Wdg_SetMode of the corresponding Watchdog Driver. |
[SRS_Wdg_12018]

[SWS_Wdglf_00057] Wdglf_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]

<table>
<thead>
<tr>
<th>Service name:</th>
<th>WdgIf_SetTriggerCondition</th>
</tr>
</thead>
</table>
| 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]

<table>
<thead>
<tr>
<th>Service name:</th>
<th>WdgIf_GetVersionInfo</th>
</tr>
</thead>
</table>
| 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 development 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]

<table>
<thead>
<tr>
<th>API function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wdg_SetMode</td>
<td>Switches the watchdog into the mode Mode.</td>
</tr>
<tr>
<td>Wdg_SetTriggerCondition</td>
<td>Sets the timeout value for the trigger counter.</td>
</tr>
</tbody>
</table>

(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]

<table>
<thead>
<tr>
<th>API function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Det_ReportError</td>
<td>Service to report development errors.</td>
</tr>
</tbody>
</table>

(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

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_WdgIf_00033 :</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Name</td>
<td>WdgIf</td>
</tr>
<tr>
<td>Module Description</td>
<td>Configuration of the WdgIf (Watchdog Interface) module.</td>
</tr>
<tr>
<td>Post-Build Variant Support</td>
<td>false</td>
</tr>
<tr>
<td>Supported Config Variants</td>
<td>VARIANT-PRE-Compile</td>
</tr>
</tbody>
</table>

### Included Containers

<table>
<thead>
<tr>
<th>Container Name</th>
<th>Multiplicity</th>
<th>Scope / Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>WdgIfDevice</td>
<td>1..*</td>
<td>It contains the information for the selection of a particular Watchdog device in case multiple Watchdog drivers are connected.</td>
</tr>
<tr>
<td>WdgIfGeneral</td>
<td>1</td>
<td>This container collects all generic watchdog interface parameters.</td>
</tr>
</tbody>
</table>

10.2.2 WdgIfGeneral

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_WdgIf_00001 :</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container Name</td>
<td>WdgIfGeneral</td>
</tr>
<tr>
<td>Description</td>
<td>This container collects all generic watchdog interface parameters.</td>
</tr>
</tbody>
</table>

#### Configuration Parameters

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_WdgIf_00005 :</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>WdgIfDevErrorDetect</td>
</tr>
<tr>
<td>Parent Container</td>
<td>WdgIfGeneral</td>
</tr>
<tr>
<td>Description</td>
<td>Switches the development error detection and notification on or off.</td>
</tr>
<tr>
<td></td>
<td>• true: detection and notification is enabled.</td>
</tr>
<tr>
<td></td>
<td>• false: detection and notification is disabled.</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>1</td>
</tr>
<tr>
<td>Type</td>
<td>EcucBooleanParamDef</td>
</tr>
<tr>
<td>Default value</td>
<td>false</td>
</tr>
<tr>
<td>Post-Build Variant Value</td>
<td>false</td>
</tr>
<tr>
<td>Value Configuration Class</td>
<td>Pre-compile time X All Variants</td>
</tr>
<tr>
<td></td>
<td>Link time</td>
</tr>
<tr>
<td></td>
<td>Post-build time</td>
</tr>
<tr>
<td>Scope / Dependency</td>
<td>scope: local</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_WdgIf_00003 :</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>WdgIfVersionInfoApi</td>
</tr>
<tr>
<td>Parent Container</td>
<td>WdgIfGeneral</td>
</tr>
<tr>
<td>Description</td>
<td>Pre-processor switch to enable / disable the service returning the version information.</td>
</tr>
<tr>
<td></td>
<td>• true: Version information service enabled</td>
</tr>
<tr>
<td></td>
<td>• false: Version information service disabled</td>
</tr>
</tbody>
</table>
Multiplicty
Type
Default value
Post-Build Variant Value
Value Configuration Class
Scope / Dependency

No Included Containers

10.2.3 WdgIfDevice

SWS Item
Name
Parent Container
Description
Multiplicity
Type
Default value
Post-Build Variant Value
Value Configuration Class
Scope / Dependency

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_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_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_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)