Document Title	General Requirements specific to Adaptive Platform
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	714

Document Status	Final
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	17-10

Document Change History			
Date	**Release**	**Changed by**	**Description**
2017-10-27	17-10	AUTOSAR Release Management	• Minor fixes
2017-03-31	17-03	AUTOSAR Release Management	• Initial release

Table of Contents

1 Scope of this document

The goal of this document is to define a common set of basic requirements that apply to all SWS documents of the Adaptive Platform.

1.1 Document Conventions

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability ([1]).

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for Traceability ([1]).

2 Acronyms and Abbreviations

There are no acronyms and abbreviations relevant within this document that are not included in the [2, AUTOSAR glossary].

3 Requirements Tracing

The following table references the requirements specified in [3] and links to the fulfillments of these.

Requirement	Description	Satisfied by
[RS_Main_00030]	AUTOSAR shall support development processes for safety related systems	[RS_AP_00113]
[RS_Main_00150]	AUTOSAR shall support the deployment and reallocation of AUTOSAR Application Software	[RS_AP_00111] [RS_AP_00115] [RS_AP_00116] [RS_AP_00117] [RS_AP_00118] [RS_AP_00119] [RS_AP_00120] [RS_AP_00121] [RS_AP_00122] [RS_AP_00124] [RS_AP_00125]
[RS_Main_00500]	AUTOSAR shall provide naming conventions	[RS_AP_00113] [RS_AP_00115] [RS_AP_00116] [RS_AP_00117] [RS_AP_00120] [RS_AP_00121] [RS_AP_00122] [RS_AP_00124] [RS_AP_00125]
[RS_Main_00513]	AUTOSAR shall support language bindings for different programming languages	[RS_AP_00114]

4 Requirements specification

4.1 Architecture Requirements

4.2 Non-functional Requirements

[RS_AP_00111] The AUTOSAR Adaptive platform shall support source code portability for AUTOSAR Adaptive applications. ⌈

Type:	draft
Description:	The AUTOSAR Adaptive platform shall support source code portability
Rationale:	Ensure reuse of existing IPs.
Dependencies:	–
Use Case:	Integration of applications on a platform, one of them delivered as source code.
Supporting Material:	–

⌋*(RS_Main_00150)*

4.3 Design Requirements

[RS_AP_00113] API specification shall comply with selected coding guidelines. ⌈

Type:	draft
Description:	API specification shall comply with coding guideline "Guidelines for the use of the C++14 language in critical and safety-related systems" in order to adopt best practices for the safe and secure implementations.
Rationale:	Enhance readability and maintainability of specification.
Dependencies:	–
Use Case:	–
Supporting Material:	"Guidelines for the use of the C++14 language in critical and safety-related systems"

⌋*(RS_Main_00500, RS_Main_00030)*

[RS_AP_00114] C++ binding shall be compatible with C++11. ⌈

Type:	draft
Description:	The AUTOSAR Adaptive platform shall be compatible with C++11.
Rationale:	The AUTOSAR Adaptive platform standard is designed to be compatible with C++11 due to high availability of C++11 compiler for embedded devices. Nevertheless projects are free to use newer C++ version like C++14. Adaptive platform vendors can restrict their package to a newer C++ version (e.g. to support newer build systems).
Dependencies:	RS_Main_00513

Use Case:	To master the complexity of the application development, the AUTOSAR Adaptive platform shall support object-oriented programming. C++ is the programming language, which supports object-oriented programming and is best suited for memory-constrained and real-time applications.
Supporting Material:	Currently (as on 2016-02-09), there are four versions of the C++ standard (ISO/IEC 14882) available: C++98, C++03, C++11, and C++14. The standardized, idiomatic, exception-safe C++ threading is only available with C++11 (and not with C++03). There is a high chance that only few compilers provide C++14 with Release 1.0, and no C++14 feature was strongly requested. Hence, the decision is to require C++11 language support for Release 1.0.

⌋*(RS_Main_00513)*

[RS_AP_00115] Standardized scope/namespace definition ⌈

Type:	draft
Description:	The namespace of adaptive platform in global scope shall be "ara". Within "ara" namespace each functional cluster shall have an own namespace with its shortname. All names shall be all lower-case. Underscores can be used.
Rationale:	Harmonized look and feel.
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋*(RS_Main_00500, RS_Main_00150)*

[RS_AP_00116] Header file name ⌈

Type:	draft
Description:	All ara libraries shall provide a self-contained header file for each public class (except scoped enum and exceptions). The header file name shall be derived from the class name as recommended by Google C++ Style Guide. All header file names shall have the extensions .h.
Rationale:	Harmonized look and feel.
Dependencies:	–
Use Case:	–
Supporting Material:	Google C++ Style Guide [https://google.github.io/styleguide/cppguide.html]

⌋*(RS_Main_00500, RS_Main_00150)*

[RS_AP_00117] Class and structure names ⌈

Type:	draft
Description:	For all ara libraries the name of their public classes and structures shall be standardized in PascalCase. Further underscores shall not be used. Capitalized acronyms shall be used as single words.
Rationale:	Harmonized look and feel.
Dependencies:	–
Use Case:	–
Supporting Material:	https://en.wikipedia.org/wiki/PascalCase

⌋(*RS_Main_00500*, *RS_Main_00150*)

[RS_AP_00122] Type names ⌈

Type:	draft
Description:	For all ara libraries the name of their types shall be standardized in PascalCase. Further underscores shall not be used. Capitalized acronyms shall be used as single words.
Rationale:	Harmonized look and feel.
Dependencies:	–
Use Case:	–
Supporting Material:	https://en.wikipedia.org/wiki/PascalCase

⌋(*RS_Main_00500*, *RS_Main_00150*)

[RS_AP_00120] Method and Function names ⌈

Type:	draft
Description:	For all ara libraries the name of their public methods and functions shall be standardized in PascalCase. Further underscores shall not be used. Capitalized acronyms shall be used as single words.
Rationale:	Harmonized look and feel.
Dependencies:	–
Use Case:	–
Supporting Material:	https://en.wikipedia.org/wiki/PascalCase

⌋(*RS_Main_00500*, *RS_Main_00150*)

[RS_AP_00121] Parameter names ⌈

Type:	draft
Description:	For all ara libraries the name of parameters in methods shall be standardized in lower camel case. Further underscores shall not be used. Capitalized acronyms shall be used as single words.
Rationale:	Harmonized look and feel.
Dependencies:	–
Use Case:	–
Supporting Material:	https://en.wikipedia.org/wiki/CamelCase

⌋(*RS_Main_00500*, *RS_Main_00150*)

[RS_AP_00124] Variable names ⌈

Type:	draft
Description:	For all ara libraries the name of their variables (like Common Variable names, Class Data Members and Struct Data Members) shall be standardized in lower camel case. Further underscores shall not be used. Capitalized acronyms shall be used as single words.
Rationale:	Harmonized look and feel.
Dependencies:	–
Use Case:	–

Supporting Material:	https://en.wikipedia.org/wiki/CamelCase

⌋*(RS_Main_00500, RS_Main_00150)*

[RS_AP_00125] Enumerator and constant names ⌈

Type:	draft
Description:	For all ara libraries the name of enumerations shall be standardized in PascalCase. The individual enumerators and constants shall be written with a leading "k" followed by PascalCase. Further underscores shall not be used. Capitalized acronyms shall be used as single words.
Rationale:	Harmonized look and feel.
Dependencies:	–
Use Case:	–
Supporting Material:	https://en.wikipedia.org/wiki/PascalCase

⌋*(RS_Main_00500, RS_Main_00150)*

[RS_AP_00118] Exceptions ⌈

Type:	draft
Description:	For all ara libraries the 'checked exception' shall be standardized and documented: when they are thrown and what is the rationale. The type names for exceptions, that are fully specified in AUTOSAR SWS specifications (i.e. not derived from the configuration), shall include the suffix "Exception". Adaptive Platform Vendors can add 'unchecked exceptions' on their own if necessary.
Rationale:	Harmonized look and feel.
Dependencies:	For adaptive services it is done via 'extended application error'.
Use Case:	–
Supporting Material:	

⌋*(RS_Main_00150)*

[RS_AP_00119] Return values / application errors ⌈

Type:	draft
Description:	For return values and application errors the condition(s) when they are returned shall be specified. Furthermore for each return value the list of valid output parameter(s) shall be specified.
Rationale:	Harmonized look and feel.
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋*(RS_Main_00150)*

5 References

[1] Standardization Template
AUTOSAR_TPS_StandardizationTemplate

[2] Glossary
AUTOSAR_TR_Glossary

[3] Main Requirements
AUTOSAR_RS_Main