Document Title	Specification of RESTful communication
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	876

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	• Initial release

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

This document contains the requirements on the functionality, API and the configuration of the AUTOSAR Adaptive RESTful Communication as part of the Adaptive AUTOSAR platform foundation.

The Communication Management in general realizes functionality to establish communication paths between Adaptive AUTOSAR Applications. This document describes the RESTful Communication part (ara::rest API) whereas [1] covers Service Oriented Communication (ara::com API).

The API design of ara::rest is based on the REST paradigm introduced by [2]. The API is geared towards low and predictable resource usage which is often important in the automotive domain. This specification is focused on the low-level components to provide all features required to design RESTful APIs and deploy services on top.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the Communication Management that are not included in the AUTOSAR glossary [3].

Abbreviation / Acronym:	Description:
REST	Representational State Transfer
HTTP	Hypertext Transfer Protocol

3 Related documentation

3.1 Input documents

[1] Specification of Communication Management
AUTOSAR_SWS_CommunicationManagement

[2] REST: Architectural Styles and the Design of Network-based Software Architectures

[3] Glossary
AUTOSAR_TR_Glossary

[4] Requirements on Communication Management
AUTOSAR_RS_CommunicationManagement

[5] SOME/IP Protocol Specification
AUTOSAR_PRS_SOMEIPProtocol

3.2 Related standards and norms

See chapter 3.1.

3.3 Related specification

See chapter 3.1.

4 Constraints and assumptions

4.1 Limitations

The interfaces are only specified to the point to make semantics clear. To be precise this document does not yet fully specify the qualification C++ functions noexcept, overloading of functions to provide move semantics for optimization purposes nor does it claim to be const-correct. Move semantics in particular are specified where required for semantic correctness only. Also aspects of AUTOSAR meta-modeling and automatic generation of RESTful APIs are not within scope yet.

4.2 Applicability to car domains

No restrictions to applicability.

5 Dependencies to other functional clusters

There are currently no dependencies to other functional clusters.

6 Requirements Tracing

The following tables reference the requirements specified in the Requirements on Communication Management document [4] and links to the fulfillment of these.

Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[RS_CM_00300]	The Communication Management shall provide a framework to support RESTful services.	[SWS_REST_01101] [SWS_REST_01102] [SWS_REST_01103] [SWS_REST_01104] [SWS_REST_01105] [SWS_REST_01106] [SWS_REST_01107] [SWS_REST_01108] [SWS_REST_01109] [SWS_REST_01110] [SWS_REST_01111] [SWS_REST_01112] [SWS_REST_01201] [SWS_REST_01203] [SWS_REST_01301] [SWS_REST_01302] [SWS_REST_01303] [SWS_REST_01304] [SWS_REST_01305] [SWS_REST_01306] [SWS_REST_01307] [SWS_REST_01308] [SWS_REST_01311] [SWS_REST_01312]

Requirement	Description	Satisfied by
		[SWS_REST_01313] [SWS_REST_01401] [SWS_REST_01402] [SWS_REST_01403] [SWS_REST_01404] [SWS_REST_01405] [SWS_REST_01406] [SWS_REST_01407] [SWS_REST_01408] [SWS_REST_01409] [SWS_REST_01410] [SWS_REST_01411] [SWS_REST_01412] [SWS_REST_01413] [SWS_REST_01414] [SWS_REST_01415] [SWS_REST_01416] [SWS_REST_01417] [SWS_REST_01418] [SWS_REST_01419] [SWS_REST_01420] [SWS_REST_01421] [SWS_REST_01422] [SWS_REST_01501] [SWS_REST_01502] [SWS_REST_01503] [SWS_REST_01504] [SWS_REST_01505] [SWS_REST_01506] [SWS_REST_01507] [SWS_REST_01508] [SWS_REST_01509] [SWS_REST_01510] [SWS_REST_01511] [SWS_REST_01512] [SWS_REST_01513] [SWS_REST_01514] [SWS_REST_01515] [SWS_REST_01516] [SWS_REST_01517] [SWS_REST_01518] [SWS_REST_01519] [SWS_REST_01522] [SWS_REST_01523] [SWS_REST_01524] [SWS_REST_01525] [SWS_REST_01526] [SWS_REST_01527]

Requirement	Description	Satisfied by
		[SWS_REST_01528] [SWS_REST_01529] [SWS_REST_01530] [SWS_REST_01531] [SWS_REST_01532] [SWS_REST_01533] [SWS_REST_01534] [SWS_REST_01535] [SWS_REST_01536] [SWS_REST_01537] [SWS_REST_01538] [SWS_REST_01601] [SWS_REST_01602] [SWS_REST_01603] [SWS_REST_01604] [SWS_REST_01605] [SWS_REST_01606] [SWS_REST_01607] [SWS_REST_01608] [SWS_REST_01609] [SWS_REST_01610] [SWS_REST_01611] [SWS_REST_01612] [SWS_REST_01613] [SWS_REST_01614] [SWS_REST_01615] [SWS_REST_01616] [SWS_REST_01617] [SWS_REST_01618] [SWS_REST_01619] [SWS_REST_01620] [SWS_REST_01621] [SWS_REST_01622] [SWS_REST_01623] [SWS_REST_01624] [SWS_REST_01625] [SWS_REST_01626] [SWS_REST_01627] [SWS_REST_01628] [SWS_REST_01629] [SWS_REST_01701] [SWS_REST_01702] [SWS_REST_01703] [SWS_REST_01704] [SWS_REST_01705] [SWS_REST_01706] [SWS_REST_01707] [SWS_REST_01708]

Requirement	Description	Satisfied by
		[SWS_REST_01709] [SWS_REST_01710] [SWS_REST_01711] [SWS_REST_01712] [SWS_REST_01713] [SWS_REST_01714] [SWS_REST_01715] [SWS_REST_02000] [SWS_REST_02001] [SWS_REST_02002] [SWS_REST_02003] [SWS_REST_02004] [SWS_REST_02005] [SWS_REST_02006] [SWS_REST_02007] [SWS_REST_02008] [SWS_REST_02009] [SWS_REST_02010] [SWS_REST_02011] [SWS_REST_02012] [SWS_REST_02013] [SWS_REST_02014] [SWS_REST_02015] [SWS_REST_02016] [SWS_REST_02017] [SWS_REST_02018] [SWS_REST_02019] [SWS_REST_02020] [SWS_REST_02021] [SWS_REST_02022] [SWS_REST_02023] [SWS_REST_02024] [SWS_REST_02025] [SWS_REST_02026] [SWS_REST_02027] [SWS_REST_02028] [SWS_REST_02029] [SWS_REST_02030] [SWS_REST_02031] [SWS_REST_02032] [SWS_REST_02033] [SWS_REST_02034] [SWS_REST_02035] [SWS_REST_02036] [SWS_REST_02037] [SWS_REST_02038] [SWS_REST_02039] [SWS_REST_02040]

Requirement	Description	Satisfied by
		[SWS_REST_02041] [SWS_REST_02042] [SWS_REST_02043] [SWS_REST_02044] [SWS_REST_02045] [SWS_REST_02046] [SWS_REST_02047] [SWS_REST_02048] [SWS_REST_02049] [SWS_REST_02050] [SWS_REST_02051] [SWS_REST_02052] [SWS_REST_02053] [SWS_REST_02054] [SWS_REST_02055] [SWS_REST_02056] [SWS_REST_02057] [SWS_REST_02058] [SWS_REST_02059] [SWS_REST_02060] [SWS_REST_02061] [SWS_REST_02062] [SWS_REST_02063] [SWS_REST_02064] [SWS_REST_02065] [SWS_REST_02066] [SWS_REST_02067] [SWS_REST_02068] [SWS_REST_02069] [SWS_REST_02070] [SWS_REST_02071] [SWS_REST_02072] [SWS_REST_02073] [SWS_REST_02074] [SWS_REST_02075] [SWS_REST_02076] [SWS_REST_02077] [SWS_REST_02078] [SWS_REST_02079] [SWS_REST_02080] [SWS_REST_02081] [SWS_REST_02082] [SWS_REST_02083] [SWS_REST_02084] [SWS_REST_02085] [SWS_REST_02086] [SWS_REST_02087] [SWS_REST_02088]

Requirement	Description	Satisfied by
		[SWS_REST_02089] [SWS_REST_02090] [SWS_REST_02091] [SWS_REST_02092] [SWS_REST_02093] [SWS_REST_02094] [SWS_REST_02095] [SWS_REST_02096] [SWS_REST_02097] [SWS_REST_02098] [SWS_REST_02099] [SWS_REST_02100] [SWS_REST_02101] [SWS_REST_02102] [SWS_REST_02103] [SWS_REST_02104] [SWS_REST_02105] [SWS_REST_02106] [SWS_REST_02107] [SWS_REST_02108] [SWS_REST_02109] [SWS_REST_02110] [SWS_REST_02111] [SWS_REST_02112] [SWS_REST_02113] [SWS_REST_02114] [SWS_REST_02115] [SWS_REST_02116] [SWS_REST_02117] [SWS_REST_02118] [SWS_REST_02119] [SWS_REST_02120] [SWS_REST_02121] [SWS_REST_02122] [SWS_REST_02123] [SWS_REST_02124] [SWS_REST_02125] [SWS_REST_02126] [SWS_REST_02127] [SWS_REST_02128] [SWS_REST_02129] [SWS_REST_02130] [SWS_REST_02131] [SWS_REST_02132] [SWS_REST_02133] [SWS_REST_02134] [SWS_REST_02135] [SWS_REST_02136]

Requirement	Description	Satisfied by
		[SWS_REST_02137] [SWS_REST_02138] [SWS_REST_02139] [SWS_REST_02140] [SWS_REST_02141] [SWS_REST_02142] [SWS_REST_02143] [SWS_REST_02144] [SWS_REST_02145] [SWS_REST_02146] [SWS_REST_02147] [SWS_REST_02148] [SWS_REST_02149] [SWS_REST_02150] [SWS_REST_02151] [SWS_REST_02152] [SWS_REST_02153] [SWS_REST_02154] [SWS_REST_02155] [SWS_REST_02156] [SWS_REST_02157] [SWS_REST_02158] [SWS_REST_02159] [SWS_REST_02160] [SWS_REST_02161] [SWS_REST_02162] [SWS_REST_02163] [SWS_REST_02164] [SWS_REST_02165] [SWS_REST_02166] [SWS_REST_02167] [SWS_REST_02168] [SWS_REST_02169] [SWS_REST_02170] [SWS_REST_02171] [SWS_REST_02172] [SWS_REST_02173] [SWS_REST_02174] [SWS_REST_02175] [SWS_REST_02176] [SWS_REST_02177] [SWS_REST_02178] [SWS_REST_02179] [SWS_REST_02180] [SWS_REST_02181] [SWS_REST_02182] [SWS_REST_02183] [SWS_REST_02184]

Requirement	Description	Satisfied by
		[SWS_REST_02185] [SWS_REST_02186] [SWS_REST_02187] [SWS_REST_02188] [SWS_REST_02189] [SWS_REST_02190] [SWS_REST_02191] [SWS_REST_02192] [SWS_REST_02193] [SWS_REST_02194] [SWS_REST_02195] [SWS_REST_02196] [SWS_REST_02197] [SWS_REST_02198] [SWS_REST_02199] [SWS_REST_02200] [SWS_REST_02201] [SWS_REST_02202] [SWS_REST_02203] [SWS_REST_02204] [SWS_REST_02205] [SWS_REST_02206] [SWS_REST_02207] [SWS_REST_02208] [SWS_REST_02209] [SWS_REST_02210] [SWS_REST_02211] [SWS_REST_02212] [SWS_REST_02213] [SWS_REST_02214] [SWS_REST_02215] [SWS_REST_02216] [SWS_REST_02217] [SWS_REST_02218] [SWS_REST_02219] [SWS_REST_02220] [SWS_REST_02221] [SWS_REST_02222] [SWS_REST_02223] [SWS_REST_02224] [SWS_REST_02225] [SWS_REST_02226] [SWS_REST_02227] [SWS_REST_02228] [SWS_REST_02229] [SWS_REST_02230] [SWS_REST_02231] [SWS_REST_02232]

Requirement	Description	Satisfied by
		[SWS_REST_02233] [SWS_REST_02234] [SWS_REST_02235] [SWS_REST_02236] [SWS_REST_02237] [SWS_REST_02238] [SWS_REST_02239] [SWS_REST_02240] [SWS_REST_02241] [SWS_REST_02242] [SWS_REST_02243] [SWS_REST_02244] [SWS_REST_02245] [SWS_REST_02246] [SWS_REST_02247] [SWS_REST_02248] [SWS_REST_02249] [SWS_REST_02250] [SWS_REST_02251] [SWS_REST_02252] [SWS_REST_02253] [SWS_REST_02254] [SWS_REST_02255] [SWS_REST_02256] [SWS_REST_02257] [SWS_REST_02258] [SWS_REST_02259] [SWS_REST_02260] [SWS_REST_02261] [SWS_REST_02262] [SWS_REST_02263] [SWS_REST_02264] [SWS_REST_02265] [SWS_REST_02266] [SWS_REST_02267] [SWS_REST_02268] [SWS_REST_02269] [SWS_REST_02270] [SWS_REST_02271] [SWS_REST_02272] [SWS_REST_02273] [SWS_REST_02274] [SWS_REST_02275] [SWS_REST_02276] [SWS_REST_02277] [SWS_REST_02278] [SWS_REST_02279] [SWS_REST_02280]

Requirement	Description	Satisfied by
		[SWS_REST_02281] [SWS_REST_02282] [SWS_REST_02283] [SWS_REST_02284] [SWS_REST_02285] [SWS_REST_02286] [SWS_REST_02287] [SWS_REST_02288] [SWS_REST_02289] [SWS_REST_02290] [SWS_REST_02291] [SWS_REST_02292] [SWS_REST_02293] [SWS_REST_02294] [SWS_REST_02295] [SWS_REST_02296] [SWS_REST_02297] [SWS_REST_02298] [SWS_REST_02299] [SWS_REST_02300] [SWS_REST_02301] [SWS_REST_02302] [SWS_REST_02303] [SWS_REST_02304] [SWS_REST_02305] [SWS_REST_02306] [SWS_REST_02307] [SWS_REST_02308] [SWS_REST_02309] [SWS_REST_02310] [SWS_REST_02311] [SWS_REST_02312] [SWS_REST_02313] [SWS_REST_02314] [SWS_REST_02315] [SWS_REST_02316] [SWS_REST_02317] [SWS_REST_02318] [SWS_REST_02319] [SWS_REST_02320] [SWS_REST_02321] [SWS_REST_02322] [SWS_REST_02323] [SWS_REST_02324] [SWS_REST_02325] [SWS_REST_02326] [SWS_REST_02327] [SWS_REST_02328]

Requirement	Description	Satisfied by
		[SWS_REST_02329] [SWS_REST_02330] [SWS_REST_02331] [SWS_REST_02332] [SWS_REST_02333] [SWS_REST_02334] [SWS_REST_02335] [SWS_REST_02336] [SWS_REST_02337] [SWS_REST_02338] [SWS_REST_02339] [SWS_REST_02340] [SWS_REST_02341] [SWS_REST_02342] [SWS_REST_02343] [SWS_REST_02344] [SWS_REST_02345] [SWS_REST_02346] [SWS_REST_02347] [SWS_REST_02348] [SWS_REST_02349] [SWS_REST_02350] [SWS_REST_02351] [SWS_REST_02352] [SWS_REST_02353] [SWS_REST_02354] [SWS_REST_02355] [SWS_REST_02356] [SWS_REST_02357] [SWS_REST_02358] [SWS_REST_02359] [SWS_REST_02360] [SWS_REST_02361] [SWS_REST_02362] [SWS_REST_02363] [SWS_REST_02364] [SWS_REST_02365] [SWS_REST_02366] [SWS_REST_02367] [SWS_REST_02368] [SWS_REST_02369] [SWS_REST_02370] [SWS_REST_02371] [SWS_REST_02372] [SWS_REST_02373] [SWS_REST_02374] [SWS_REST_02375] [SWS_REST_02376]

Requirement	Description	Satisfied by
		[SWS_REST_02377] [SWS_REST_02378] [SWS_REST_02379] [SWS_REST_02380] [SWS_REST_02381] [SWS_REST_02382] [SWS_REST_02383] [SWS_REST_02384] [SWS_REST_02385] [SWS_REST_02386] [SWS_REST_02387] [SWS_REST_02388]
[RS_CM_00301]	The Communication Management shall provide an abstraction of network protocols for RESTful services.	[SWS_REST_01301] [SWS_REST_01302] [SWS_REST_01304] [SWS_REST_01305] [SWS_REST_01306] [SWS_REST_01307] [SWS_REST_01308] [SWS_REST_01311] [SWS_REST_01312] [SWS_REST_01313] [SWS_REST_01401] [SWS_REST_01402] [SWS_REST_01403] [SWS_REST_01404] [SWS_REST_01405] [SWS_REST_01406] [SWS_REST_01407] [SWS_REST_01408] [SWS_REST_01409] [SWS_REST_01410] [SWS_REST_01411] [SWS_REST_01412] [SWS_REST_01413] [SWS_REST_01414]

Requirement	Description	Satisfied by
		[SWS_REST_01415] [SWS_REST_01416] [SWS_REST_01417] [SWS_REST_01418] [SWS_REST_01419] [SWS_REST_01420] [SWS_REST_01421] [SWS_REST_01422] [SWS_REST_01501] [SWS_REST_01502] [SWS_REST_01503] [SWS_REST_01504] [SWS_REST_01505] [SWS_REST_01506] [SWS_REST_01507] [SWS_REST_01508] [SWS_REST_01509] [SWS_REST_01510] [SWS_REST_01511] [SWS_REST_01512] [SWS_REST_01513] [SWS_REST_01514] [SWS_REST_01515] [SWS_REST_01516] [SWS_REST_01517] [SWS_REST_01518] [SWS_REST_01519] [SWS_REST_01522] [SWS_REST_01523] [SWS_REST_01524] [SWS_REST_01525] [SWS_REST_01526] [SWS_REST_01527] [SWS_REST_01528] [SWS_REST_01529] [SWS_REST_01530] [SWS_REST_01531] [SWS_REST_01532] [SWS_REST_01533] [SWS_REST_01534] [SWS_REST_01535] [SWS_REST_01536] [SWS_REST_01537] [SWS_REST_01538] [SWS_REST_02006] [SWS_REST_02007] [SWS_REST_02008] [SWS_REST_02009]

Requirement	Description	Satisfied by
		[SWS_REST_02010] [SWS_REST_02011] [SWS_REST_02012] [SWS_REST_02013] [SWS_REST_02014] [SWS_REST_02015] [SWS_REST_02016] [SWS_REST_02238] [SWS_REST_02239] [SWS_REST_02240] [SWS_REST_02241] [SWS_REST_02242] [SWS_REST_02243] [SWS_REST_02244] [SWS_REST_02245] [SWS_REST_02246] [SWS_REST_02247] [SWS_REST_02248] [SWS_REST_02249]
[RS_CM_00304]	The Communication Management shall support URIs according to RFC3986 to identify data.	[SWS_REST_01101] [SWS_REST_01102] [SWS_REST_02022] [SWS_REST_02167] [SWS_REST_02168] [SWS_REST_02178] [SWS_REST_02179] [SWS_REST_02259] [SWS_REST_02260] [SWS_REST_02261] [SWS_REST_02262] [SWS_REST_02263] [SWS_REST_02264] [SWS_REST_02265] [SWS_REST_02266] [SWS_REST_02267] [SWS_REST_02268] [SWS_REST_02269] [SWS_REST_02270] [SWS_REST_02271] [SWS_REST_02272] [SWS_REST_02273] [SWS_REST_02274] [SWS_REST_02275]

Requirement	Description	Satisfied by
		[SWS_REST_02276] [SWS_REST_02277] [SWS_REST_02278] [SWS_REST_02279] [SWS_REST_02280] [SWS_REST_02281] [SWS_REST_02282] [SWS_REST_02283] [SWS_REST_02284] [SWS_REST_02285] [SWS_REST_02286] [SWS_REST_02287] [SWS_REST_02288] [SWS_REST_02289] [SWS_REST_02290] [SWS_REST_02291] [SWS_REST_02292] [SWS_REST_02293] [SWS_REST_02294] [SWS_REST_02295] [SWS_REST_02296] [SWS_REST_02297] [SWS_REST_02298] [SWS_REST_02299] [SWS_REST_02300] [SWS_REST_02301] [SWS_REST_02302] [SWS_REST_02303] [SWS_REST_02304] [SWS_REST_02305] [SWS_REST_02306] [SWS_REST_02307] [SWS_REST_02308] [SWS_REST_02309] [SWS_REST_02310] [SWS_REST_02311] [SWS_REST_02312] [SWS_REST_02313] [SWS_REST_02314] [SWS_REST_02315] [SWS_REST_02316] [SWS_REST_02317] [SWS_REST_02318] [SWS_REST_02319] [SWS_REST_02320] [SWS_REST_02321] [SWS_REST_02322] [SWS_REST_02323]

Requirement	Description	Satisfied by
		[SWS_REST_02324] [SWS_REST_02325] [SWS_REST_02326] [SWS_REST_02327] [SWS_REST_02328] [SWS_REST_02329] [SWS_REST_02330] [SWS_REST_02372] [SWS_REST_02373] [SWS_REST_02374] [SWS_REST_02375] [SWS_REST_02376] [SWS_REST_02377] [SWS_REST_02378] [SWS_REST_02379] [SWS_REST_02380] [SWS_REST_02381]
[RS_CM_00305]	The Communication Management shall represent data as an Object in an Object Graph.	[SWS_REST_01304] [SWS_REST_01702] [SWS_REST_01703] [SWS_REST_01704] [SWS_REST_01705] [SWS_REST_01706] [SWS_REST_01707] [SWS_REST_01708] [SWS_REST_01709] [SWS_REST_01710] [SWS_REST_01711] [SWS_REST_01712] [SWS_REST_01713] [SWS_REST_01714] [SWS_REST_01715] [SWS_REST_02036] [SWS_REST_02037] [SWS_REST_02038] [SWS_REST_02039] [SWS_REST_02040] [SWS_REST_02041] [SWS_REST_02042] [SWS_REST_02043] [SWS_REST_02044]

Requirement	Description	Satisfied by
		[SWS_REST_02045] [SWS_REST_02046] [SWS_REST_02047] [SWS_REST_02048] [SWS_REST_02049] [SWS_REST_02050] [SWS_REST_02051] [SWS_REST_02052] [SWS_REST_02053] [SWS_REST_02054] [SWS_REST_02055] [SWS_REST_02056] [SWS_REST_02057] [SWS_REST_02058] [SWS_REST_02059] [SWS_REST_02060] [SWS_REST_02061] [SWS_REST_02062] [SWS_REST_02063] [SWS_REST_02064] [SWS_REST_02065] [SWS_REST_02066] [SWS_REST_02067] [SWS_REST_02068] [SWS_REST_02069] [SWS_REST_02070] [SWS_REST_02071] [SWS_REST_02072] [SWS_REST_02073] [SWS_REST_02074] [SWS_REST_02075] [SWS_REST_02076] [SWS_REST_02077] [SWS_REST_02078] [SWS_REST_02079] [SWS_REST_02080] [SWS_REST_02081] [SWS_REST_02082] [SWS_REST_02083] [SWS_REST_02084] [SWS_REST_02085] [SWS_REST_02086] [SWS_REST_02087] [SWS_REST_02088] [SWS_REST_02089] [SWS_REST_02090] [SWS_REST_02091] [SWS_REST_02092]

Requirement	Description	Satisfied by
		[SWS_REST_02093] [SWS_REST_02094] [SWS_REST_02095] [SWS_REST_02096] [SWS_REST_02097] [SWS_REST_02098] [SWS_REST_02099] [SWS_REST_02100] [SWS_REST_02101] [SWS_REST_02102] [SWS_REST_02103] [SWS_REST_02104] [SWS_REST_02105] [SWS_REST_02106] [SWS_REST_02107] [SWS_REST_02108] [SWS_REST_02109] [SWS_REST_02110] [SWS_REST_02111] [SWS_REST_02112] [SWS_REST_02113] [SWS_REST_02114] [SWS_REST_02115] [SWS_REST_02116] [SWS_REST_02117] [SWS_REST_02118] [SWS_REST_02119] [SWS_REST_02120] [SWS_REST_02121] [SWS_REST_02122] [SWS_REST_02123] [SWS_REST_02124] [SWS_REST_02125] [SWS_REST_02126] [SWS_REST_02127] [SWS_REST_02128] [SWS_REST_02129] [SWS_REST_02130] [SWS_REST_02131] [SWS_REST_02132] [SWS_REST_02133] [SWS_REST_02134] [SWS_REST_02135] [SWS_REST_02136] [SWS_REST_02137] [SWS_REST_02138] [SWS_REST_02139] [SWS_REST_02140]

Requirement	Description	Satisfied by
		[SWS_REST_02141] [SWS_REST_02142] [SWS_REST_02143] [SWS_REST_02144] [SWS_REST_02145] [SWS_REST_02146] [SWS_REST_02147] [SWS_REST_02148] [SWS_REST_02149] [SWS_REST_02150] [SWS_REST_02151] [SWS_REST_02152] [SWS_REST_02153] [SWS_REST_02154] [SWS_REST_02155] [SWS_REST_02156] [SWS_REST_02157] [SWS_REST_02158] [SWS_REST_02343] [SWS_REST_02344] [SWS_REST_02345] [SWS_REST_02346] [SWS_REST_02347] [SWS_REST_02348]
[RS_CM_00306]	The Communication Management shall provide an Object Graph which is independent of the used serialization format.	[SWS_REST_02036] [SWS_REST_02037] [SWS_REST_02038] [SWS_REST_02039] [SWS_REST_02040] [SWS_REST_02041] [SWS_REST_02042] [SWS_REST_02043] [SWS_REST_02044] [SWS_REST_02045] [SWS_REST_02046] [SWS_REST_02047] [SWS_REST_02048] [SWS_REST_02049] [SWS_REST_02050] [SWS_REST_02051] [SWS_REST_02052] [SWS_REST_02053] [SWS_REST_02054] [SWS_REST_02055] [SWS_REST_02056] [SWS_REST_02057] [SWS_REST_02058] [SWS_REST_02059]

Requirement	Description	Satisfied by
		[SWS_REST_02060] [SWS_REST_02061] [SWS_REST_02062] [SWS_REST_02063] [SWS_REST_02064] [SWS_REST_02065] [SWS_REST_02066] [SWS_REST_02067] [SWS_REST_02068] [SWS_REST_02069] [SWS_REST_02070] [SWS_REST_02071] [SWS_REST_02072] [SWS_REST_02073] [SWS_REST_02074] [SWS_REST_02075] [SWS_REST_02076] [SWS_REST_02077] [SWS_REST_02078] [SWS_REST_02079] [SWS_REST_02080] [SWS_REST_02081] [SWS_REST_02082] [SWS_REST_02083] [SWS_REST_02084] [SWS_REST_02085] [SWS_REST_02086] [SWS_REST_02087] [SWS_REST_02088] [SWS_REST_02089] [SWS_REST_02090] [SWS_REST_02091] [SWS_REST_02092] [SWS_REST_02093] [SWS_REST_02094] [SWS_REST_02095] [SWS_REST_02096] [SWS_REST_02097] [SWS_REST_02098] [SWS_REST_02099] [SWS_REST_02100] [SWS_REST_02101] [SWS_REST_02102] [SWS_REST_02103] [SWS_REST_02104] [SWS_REST_02105] [SWS_REST_02106] [SWS_REST_02107]

Requirement	Description	Satisfied by
		[SWS_REST_02108] [SWS_REST_02109] [SWS_REST_02110] [SWS_REST_02111] [SWS_REST_02112] [SWS_REST_02113] [SWS_REST_02114] [SWS_REST_02115] [SWS_REST_02116] [SWS_REST_02117] [SWS_REST_02118] [SWS_REST_02119] [SWS_REST_02120] [SWS_REST_02121] [SWS_REST_02122] [SWS_REST_02123] [SWS_REST_02124] [SWS_REST_02125] [SWS_REST_02126] [SWS_REST_02127] [SWS_REST_02128] [SWS_REST_02129] [SWS_REST_02130] [SWS_REST_02131] [SWS_REST_02132] [SWS_REST_02133] [SWS_REST_02134] [SWS_REST_02135] [SWS_REST_02136] [SWS_REST_02137] [SWS_REST_02138] [SWS_REST_02139] [SWS_REST_02140] [SWS_REST_02141] [SWS_REST_02142] [SWS_REST_02143] [SWS_REST_02144] [SWS_REST_02145] [SWS_REST_02146] [SWS_REST_02147] [SWS_REST_02148] [SWS_REST_02149] [SWS_REST_02150] [SWS_REST_02151] [SWS_REST_02152] [SWS_REST_02153] [SWS_REST_02154] [SWS_REST_02155]

Requirement	Description	Satisfied by
		[SWS_REST_02156] [SWS_REST_02157] [SWS_REST_02158] [SWS_REST_02343] [SWS_REST_02344] [SWS_REST_02345] [SWS_REST_02346] [SWS_REST_02347] [SWS_REST_02348]
[RS_CM_00307]	The Communication Management shall provide an Object Graph where each Object is strongly typed.	[SWS_REST_02036] [SWS_REST_02037] [SWS_REST_02038] [SWS_REST_02039] [SWS_REST_02040] [SWS_REST_02041] [SWS_REST_02042] [SWS_REST_02043] [SWS_REST_02044] [SWS_REST_02045] [SWS_REST_02046] [SWS_REST_02047] [SWS_REST_02048] [SWS_REST_02049] [SWS_REST_02050] [SWS_REST_02051] [SWS_REST_02052] [SWS_REST_02053] [SWS_REST_02054] [SWS_REST_02055] [SWS_REST_02056] [SWS_REST_02057] [SWS_REST_02058] [SWS_REST_02059]

Requirement	Description	Satisfied by
		[SWS_REST_02060] [SWS_REST_02061] [SWS_REST_02062] [SWS_REST_02063] [SWS_REST_02064] [SWS_REST_02065] [SWS_REST_02066] [SWS_REST_02067] [SWS_REST_02068] [SWS_REST_02069] [SWS_REST_02070] [SWS_REST_02071] [SWS_REST_02072] [SWS_REST_02073] [SWS_REST_02074] [SWS_REST_02075] [SWS_REST_02076] [SWS_REST_02077] [SWS_REST_02078] [SWS_REST_02079] [SWS_REST_02080] [SWS_REST_02081] [SWS_REST_02082] [SWS_REST_02083] [SWS_REST_02084] [SWS_REST_02085] [SWS_REST_02086] [SWS_REST_02087] [SWS_REST_02088] [SWS_REST_02089] [SWS_REST_02090] [SWS_REST_02091] [SWS_REST_02092] [SWS_REST_02093] [SWS_REST_02094] [SWS_REST_02095] [SWS_REST_02096] [SWS_REST_02097] [SWS_REST_02098] [SWS_REST_02099] [SWS_REST_02100] [SWS_REST_02101] [SWS_REST_02102] [SWS_REST_02103] [SWS_REST_02104] [SWS_REST_02105] [SWS_REST_02106] [SWS_REST_02107]

Requirement	Description	Satisfied by
		[SWS_REST_02108] [SWS_REST_02109] [SWS_REST_02110] [SWS_REST_02111] [SWS_REST_02112] [SWS_REST_02113] [SWS_REST_02114] [SWS_REST_02115] [SWS_REST_02116] [SWS_REST_02117] [SWS_REST_02118] [SWS_REST_02119] [SWS_REST_02120] [SWS_REST_02121] [SWS_REST_02122] [SWS_REST_02123] [SWS_REST_02124] [SWS_REST_02125] [SWS_REST_02126] [SWS_REST_02127] [SWS_REST_02128] [SWS_REST_02129] [SWS_REST_02130] [SWS_REST_02131] [SWS_REST_02132] [SWS_REST_02133] [SWS_REST_02134] [SWS_REST_02135] [SWS_REST_02136] [SWS_REST_02137] [SWS_REST_02138] [SWS_REST_02139] [SWS_REST_02140] [SWS_REST_02141] [SWS_REST_02142] [SWS_REST_02143] [SWS_REST_02144] [SWS_REST_02145] [SWS_REST_02146] [SWS_REST_02147] [SWS_REST_02148] [SWS_REST_02149] [SWS_REST_02150] [SWS_REST_02151] [SWS_REST_02152] [SWS_REST_02153] [SWS_REST_02154] [SWS_REST_02155]

Requirement	Description	Satisfied by
		[SWS_REST_02156] [SWS_REST_02157] [SWS_REST_02158] [SWS_REST_02343] [SWS_REST_02344] [SWS_REST_02345] [SWS_REST_02346] [SWS_REST_02347] [SWS_REST_02348]
[RS_CM_00308]	The Communication Management shall provide methods to read and manipulate the Object Graph	[SWS_REST_02039] [SWS_REST_02040] [SWS_REST_02041] [SWS_REST_02042] [SWS_REST_02043] [SWS_REST_02044] [SWS_REST_02045] [SWS_REST_02046] [SWS_REST_02047] [SWS_REST_02048] [SWS_REST_02049] [SWS_REST_02050] [SWS_REST_02051] [SWS_REST_02052] [SWS_REST_02053] [SWS_REST_02054] [SWS_REST_02055] [SWS_REST_02056] [SWS_REST_02057] [SWS_REST_02058] [SWS_REST_02059] [SWS_REST_02060] [SWS_REST_02061] [SWS_REST_02065]

Requirement	Description	Satisfied by
		[SWS_REST_02066] [SWS_REST_02067] [SWS_REST_02068] [SWS_REST_02069] [SWS_REST_02070] [SWS_REST_02071] [SWS_REST_02072] [SWS_REST_02073] [SWS_REST_02074] [SWS_REST_02075] [SWS_REST_02076] [SWS_REST_02081] [SWS_REST_02082] [SWS_REST_02083] [SWS_REST_02084] [SWS_REST_02085] [SWS_REST_02086] [SWS_REST_02087] [SWS_REST_02088] [SWS_REST_02092] [SWS_REST_02093] [SWS_REST_02094] [SWS_REST_02095] [SWS_REST_02096] [SWS_REST_02097] [SWS_REST_02098] [SWS_REST_02099] [SWS_REST_02100] [SWS_REST_02104] [SWS_REST_02105] [SWS_REST_02106] [SWS_REST_02107] [SWS_REST_02108] [SWS_REST_02109] [SWS_REST_02110] [SWS_REST_02111] [SWS_REST_02112] [SWS_REST_02113] [SWS_REST_02114] [SWS_REST_02115] [SWS_REST_02116] [SWS_REST_02117] [SWS_REST_02118] [SWS_REST_02119] [SWS_REST_02120] [SWS_REST_02121] [SWS_REST_02122] [SWS_REST_02123]

Requirement	Description	Satisfied by
		[SWS_REST_02124] [SWS_REST_02125] [SWS_REST_02126] [SWS_REST_02131] [SWS_REST_02132] [SWS_REST_02133] [SWS_REST_02134] [SWS_REST_02135] [SWS_REST_02136] [SWS_REST_02137] [SWS_REST_02138] [SWS_REST_02143] [SWS_REST_02144] [SWS_REST_02145] [SWS_REST_02146] [SWS_REST_02147] [SWS_REST_02148] [SWS_REST_02149] [SWS_REST_02150] [SWS_REST_02151] [SWS_REST_02155] [SWS_REST_02156] [SWS_REST_02157] [SWS_REST_02158] [SWS_REST_02343] [SWS_REST_02344] [SWS_REST_02345] [SWS_REST_02346] [SWS_REST_02347] [SWS_REST_02348]
[RS_CM_00309]	The Communication Management shall provide a way to match requests to corresponding server handlers and vice versa.	[SWS_REST_02027] [SWS_REST_02028] [SWS_REST_02029] [SWS_REST_02030] [SWS_REST_02031] [SWS_REST_02032] [SWS_REST_02033] [SWS_REST_02034] [SWS_REST_02035] [SWS_REST_02159] [SWS_REST_02160] [SWS_REST_02161] [SWS_REST_02162] [SWS_REST_02163]
[RS_CM_00310]	The Communication Management shall provide an interface to install request handlers.	[SWS_REST_01616] [SWS_REST_01617] [SWS_REST_01618] [SWS_REST_01624] [SWS_REST_02244]

Requirement	Description	Satisfied by
[RS_CM_00311]	The Communication Management shall provide type aliases for abstraction of standard C++ components.	[SWS_REST_01001] [SWS_REST_01002] [SWS_REST_01003] [SWS_REST_01004] [SWS_REST_01005] [SWS_REST_01006] [SWS_REST_01007] [SWS_REST_01008] [SWS_REST_01009] [SWS_REST_01010] [SWS_REST_01011] [SWS_REST_01012] [SWS_REST_01013] [SWS_REST_01014] [SWS_REST_01015] [SWS_REST_01016] [SWS_REST_01017] [SWS_REST_01018] [SWS_REST_01019] [SWS_REST_01020] [SWS_REST_02354] [SWS_REST_02355] [SWS_REST_02356] [SWS_REST_02357] [SWS_REST_02358] [SWS_REST_02359] [SWS_REST_02360]

7 Functional specification

7.1 General description

This chapter and chapter 8 specify an API design for a RESTful application framework for Adaptive AUTOSAR. Traditionally RESTful services are applications of the mobile world where resource constraints are not as severe as in the automotive domain. Neither clients, servers, transport protocols, formats and last but not least RESTful applications on top of all this are usually particularly geared towards low and predictable resource usage. Providing RESTful services in an environment with strict quality requirements such as low and deterministic memory and processing time is therefore particularly challenging. ara::rest is specifically designed for this use-case.

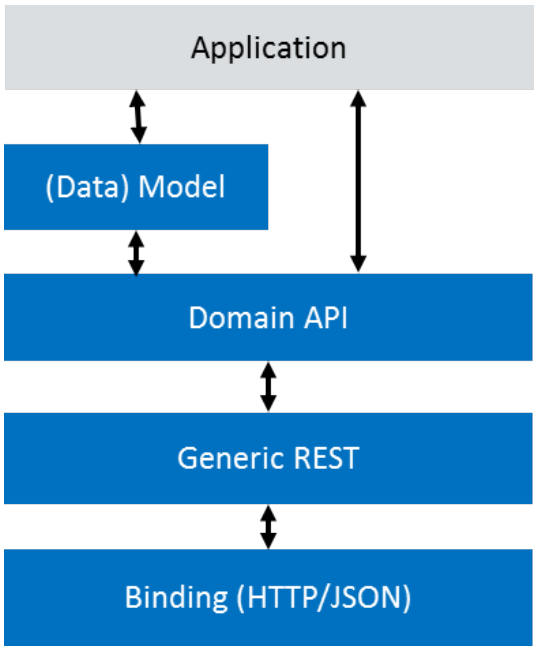

Figure 7.1: Typical RESTful service stack

A typical RESTful API stack is depicted in figure 7.1. In this, ara::rest provides the lower to stack elements and a generic data representation from which elements like a domain-specific API and a domain-specific data model can be constructed.

7.1.1 Architectural concepts

The ara::rest framework is modular in that it enables developers to access different layers involved in RESTful message transactions directly. This is in contrast to ara::com whose focus is to provide to the user a traditional function call interface and to hide all details of the transactions beyond this point. This shifts some technical effort but also a lot of control away from the user into a monolithic black box. This is a feasible design choice since there is a clear and very simple notion of what such an API constitutes: C-style functions.

As opposed to this RESTful APIs are an entirely unclear design target. Specific designs range from entirely data driven access up to simulation of RPC interfaces. In ara::com interfaces are precisely defined and message payloads are known to the byte. In RESTful APIs in general this is not necessarily the case. Control of a service is exercised entirely by means of (often dynamic) data. Specific functionality of a service is triggered by any combination of URI and data payload in the messages themselves and it is up to the service to interpret this data and turn this back into actual side-effects. To make the contrast clear, in ara::com a "meta contract" of the API exists which allows a high degree of optimization since the form of interaction is "function calls with mostly static data".

For RESTful APIs such a basic concept does not exist. Therefore RESTful API design is a two-step process of first designing a specific instance of an API which defines a class of services that agree on the general rules of interaction and second defining specific services by using the tools enabled by those rules. To make this more hands-on, there might exists a vendor-specific implementation of ara::rest, there might exist a OEM-specific RESTful API on top of this and there might exists a domain-specific service built by means of this RESTful API. ara::rest specifically addresses these requirements by incorporating a modular design which supports developers at the level of API as well as service design. The following diagram illustrates its general design. It depicts how a service application is composed in ara::rest:

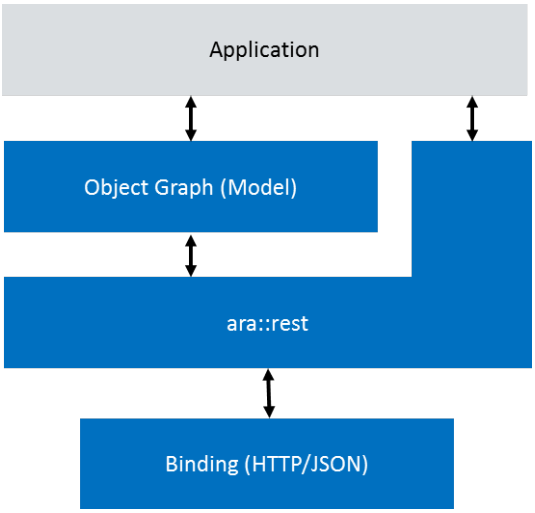

Figure 7.2: ara::rest component stack

All ara::rest communication is performed through protocol-specific bindings. By default ara::rest abstracts from these protocol details. The generic REST layer of ara::rest only provides three fundamental abstractions: a tree-structured message payload (Object Graph), a URI and a request method (like GET or POST known from HTTP). From these basic primitives domain-specific RESTful APIs can be composed (such as W3C ViWi) which defines a concrete high-level protocol for interaction via object graphs, URIs and methods. Its purpose is to define the rules for access into a domain-specific data model and to provide an abstract (C++) API to an application.

The unified data representation, called Object Graph Model (OGM) comprises of a very limited set of data primitives which reflects the simplicity of usual RESTful communication artifacts (such as JSON message payloads).

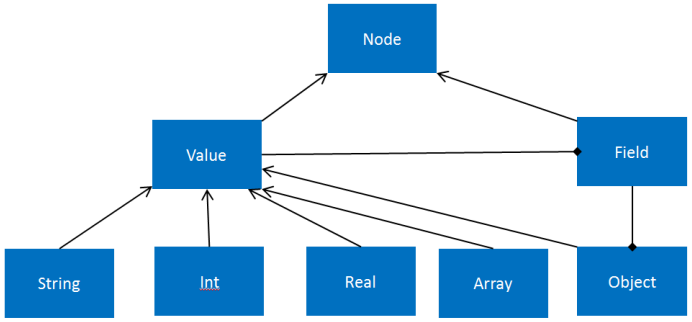

Figure 7.3: Class diagram of the Object Graph Model

7.1.2 Design Scope

This specification covers the topics support-library, client/server endpoints and object graph data structure in detail. It does not yet cover topics related to mapping the AUTOSAR meta model onto ara::rest. Security features such as the built-in support for authentication are not conclusively decided upon; mainly due to the ongoing standardization efforts in Adaptive AUTOSAR.

7.1.3 Design objectives

ara::rest has been carefully designed to enable implementations with deterministic timing, low resource footprint and low latency while being simple enough to reason about aspects of software safety. This sets it apart from existing C++ RESTful service frameworks. In the following a brief overview of its key design objectives are provided.

- Component-based: ara::rest is a construction kit to build RESTful APIs. It does not directly define a specific API upon which services can be realized. Instead it supports its construction by providing basic components that enable efficient implementations with a high degree of resource control to meet software safety requirements.
- Standard compatibility: ara::rest abstracts from standard C++ for some key components. On the one hand some critical features are not yet sufficiently mature, are not yet part of the standard or are incomplete. On the other hand some standard components inhibit safe and efficient implementations when it comes to resource control and asynchronous programming. Therefore ara::rest defines a small set of type aliases which allow for an implementation to provide custom components if the platform does not support them yet, or for enhanced safety and

efficiency. ara::rest is carefully designed to enable the use of standard C++ components for quick deployment (abstraction then boils down to simple type aliases) but also enables an implementation to replace all of them by custom components. It is not a wrapper.

- Asynchronous programming model: ara:rest is designed for asynchronous I/O as its default programming model. Consequently implementations can be highly responsive without the need for multi-threading. ara::rest does not enforce a single-thread execution model though. It is up to a concrete implementation how computing resources are being exploited.
- Task-based execution model: ara::rest is designed for low latency and low resource usage by defining a task-based execution model which complements the asynchronous programming model. ara::rest tasks abstract from traditional threading as they leave unspecified whether a task corresponds to a POSIX thread one-to-one, or whether it corresponds to a lightweight user-level thread abstraction or none of the above.
- User-defined memory management: To complement the stack-based resource model, all dynamic data structures of ara::rest support a custom allocator model to support safety and efficiency at all levels. (Allocators are compatible with C++17 PMR.) They complement the abstraction of certain critical standard C++ components.
- Unified data abstraction: The framework provides a unified API to handle message payloads and optionally the underlying data model of services. ara::rest communicates via tree-structured data called object graphs. It provides all necessary abstractions to build, traverse and dissect object graphs at C++ level. It reflects the capabilities of JSON to some extent for easy serialization. However it is designed with two additional objectives in mind. First, object graphs are designed for ara::rest specifically. Although they can be used independently of other ara::rest components, they fit into the general resource model. Second, object graphs can be specialized such that abstract objects can be derived from the AUTOSAR meta model. This lifts the handling of message payloads from the handling of primitives data types such as int, strings or "records" to the level of handling Doors, Windows and Batteries, for example. Nevertheless, a domain-specific ara::rest server that replies with a "Battery" object in its payload can communicate with a client which has no such notion and vice versa. This is key to enable communication with non-AUTOSAR RESTful clients and services. In addition this abstraction simplifies the mapping into Classic AUTOSAR since abstract object graphs precisely map to static struct representations required by SOME/IP [5].

7.1.4 Basic Components

ara::rest can roughly be subdivided into functional blocks as depcited in figure 7.4.

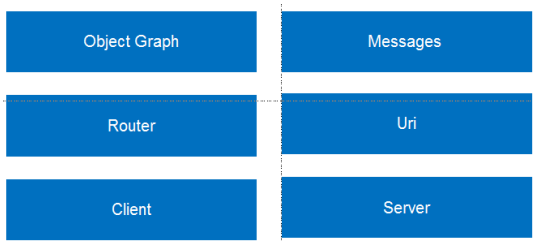

Figure 7.4: Basic components comprising ara::rest

The Object Graph is a protocol-independent tree-like data structure which is the cornerstone of all ara::rest communication. Its purpose is to map to a protocol format such as JSON as well as to C structs. This maximizes compatibility with non-ARA communication peers and Classic AUTOSAR. Object graphs are transmitted in messages which abstract completely from a concrete underlying protocol binding.

Messages encapsulate the entire context of a request/reply communication cycle in the asynchronous programming model of ara::rest.

The routing concept provides a means to map requests (including request method and URI) onto user-defined handler functions. Routing is the cornerstone to lift abstraction from generic REST into a specific kind of RESTful API.

Uri is a generic RFC-compliant, memory-efficient and exception-safe URI representation. ara::rest provides so-called (network) endpoints for server and client communication which both provide a comparable degree of resource control.

The entire framework design is strictly geared towards maximal resource control. All computations and allocations can be strictly controlled and customized to the precise needs of an application (deployment).

7.2 Support Functionality

The ara::rest framework requires abstractions from some standard C++ components that are particularly critical in terms of resource control and safety. This provides much greater control of resources such as allocation of memory and computing time and allows to fix many of the existing problems with standard C++ components. However all of these abstractions indeed have an existing counterpart defined in the C++ standard (C++11 or C++17).

[SWS_REST_01001] Use of support type aliases ⌈ The support type aliases specified in this section may either alias standard C++ library types as available, or alias custom but API-compliant implementations. ⌋*(RS_CM_00311)*

[SWS_REST_01002] Duration type ⌈ A type alias `ara::rest::duration_t` shall exist to express time spans of precision of at least microseconds. ⌋*(RS_CM_00311)*

[SWS_REST_01003] Pointer type ⌈ A type `ara::rest::Pointer` shall exist that models the precise semantics of `std::unique_ptr`. ⌋*(RS_CM_00311)*

[SWS_REST_01004] Basic string type ⌈ A type template `ara::rest::BasicString` shall exist that models the precise semantics of `std::basic_string`. The following type alias template may be used:

```
template<typename CharT, typename TraitsT = std::char_traits<CharT>>
using BasicString = std::basic_string<CharT, TraitsT, ara::rest::
StdAllocator<CharT>>

```

⌋*(RS_CM_00311)*

[SWS_REST_01005] String type ⌈ For convenience, a type alias `ara::rest::String` shall exist that models the precise semantics of `std::string`. The following type alias may be used:

```
using String = BasicString<char>

```

⌋*(RS_CM_00311)*

[SWS_REST_01006] String encoding ⌈ A String is nothing but a dynamic array of bytes. Its interpretation as a specific character encoding like UTF-8 shall not be in the scope of ara::rest. ⌋*(RS_CM_00311)*

[SWS_REST_01007] String View ⌈ A type alias template `ara::rest::BasicStringView` shall exist that models the precise semantics of `std::basic_string_view`. The following type alias template may be used:

```
template<typename CharT, typename TraitsT = std::char_traits<CharT> >
using BasicStringView = std::experimental::basic_string_view<CharT,
TraitsT, ara::rest::StdAllocator<CharT>>;

```

⌋*(RS_CM_00311)*

[SWS_REST_01008] String View aliasing ⌈ If `std::string_view` is not available, it is allowed to alias with `std::string`. ⌋*(RS_CM_00311)*

[SWS_REST_01009] String View exceptions ⌈ `ara::rest::StringView` shall not throw. ⌋*(RS_CM_00311)*

[SWS_REST_01010] String View alias exceptions ⌈ If `ara::rest::StringView` is an alias for `ara::rest::String` `noexcept` predication shall conditionally change to maintain proper exception safety specification. ⌋*(RS_CM_00311)*

[SWS_REST_01011] String View type ⌈ For convenience, a type alias `ara::rest::StringView` shall exist that models the precise semantics of `std::string_view`. The following type alias may be used:

```
using StringView = BasicStringView<char>;

```

⌋*(RS_CM_00311)*

[SWS_REST_01012] String view encoding ⌈ A StringView is nothing but a static array of bytes. Its interpretation as a specific character encoding like UTF-8 shall not be subject to ara::rest. ⌋*(RS_CM_00311)*

[SWS_REST_01013] Function type ⌈ A type alias template `ara::rest::Function` shall exist that models the precise semantics of `std::function`. The following type alias template may be used:

```
template<typename T>
using Function = std::function<T>;
```

⌋*(RS_CM_00311)*

[SWS_REST_01014] Function type ⌈ A type alias template `ara::rest::Task` shall exist that models the precise semantics of C++14 `std::future`. ⌋*(RS_CM_00311)*

[SWS_REST_01015] Function type continuations ⌈ In particular, `ara::rest::Function` shall support continuations by means of `ara::rest::Function::then`. ⌋*(RS_CM_00311)*

[SWS_REST_01016] Iterator ranges ⌈ A type `IteratorRange` shall be defined as a thin wrapper around a pair of iterators according to API `ara::rest::IteratorRange`. ⌋*(RS_CM_00311)*

[SWS_REST_01017] Allocator ⌈ A type `ara::rest::Allocator` shall exist that models the precise semantics of C++17 `std::pmr::memory_resource`. The following type alias may be used, if available:

```
using Allocator = std::experimental::pmr::memory_resource
```

⌋*(RS_CM_00311)*

[SWS_REST_01018] Allocator Adapter ⌈ For compatibility, a standard allocator adapter type shall be provided that models the precise semantics of C++17 `std::pmr::polymorphic_allocator`. The following type alias template may be used, if available:

```
template<typename T>
using StdAllocator = std::experimental::pmr::polymorphic_allocator<T>
```

⌋*(RS_CM_00311)*

[SWS_REST_01019] NewDelete Allocator ⌈ For compatibility, a default allocator type shall be provided that models the precise semantics of C++17 `std::pmr::new_delete_resource`. This type shall allocate and free memory via default new and delete operators internally. The following type alias template may be used, if available:

```
template<typename T>
using NewDeleteAllocator = std::experimental::pmr::new_delete_resource
```

⌋(*RS_CM_00311*)

[SWS_REST_01020] Get and Set Default Allocator ⌈ There shall exist two functions to request () and reset () a global default allocator instance. The following type alias template may be used, if available:

```
1    Allocator* GetDefaultAllocator() noexcept;
2    Allocator* SetDefaultAllocator(Allocator*) noexcept;
3
```

⌋(*RS_CM_00311*)

7.3 URI

ara::rest::Uri is a universal container for RFC 3986 compliant identifiers and is specifically designed with resource control in mind. ara::rest messaging is essentially based on two kinds of payloads: object graph and URI. In simple transactions, URIs potentially yield larger memory footprints than the message payloads themselves. An efficient URI type is therefore required.

[SWS_REST_01101] URI API ⌈ This type shall at least provide the interface along with its functional description as specified in `ara::rest::Uri`. ⌋(*RS_CM_00300, RS_CM_00304*)

[SWS_REST_01102] URI UTF-8 ⌈ `ara::rest::Uri` shall support UTF-8 in percent-encoded form (see RFC 3986). Otherwise no awareness of a particular character encoding is required. ⌋(*RS_CM_00300, RS_CM_00304*)

[SWS_REST_01103] URI String output ⌈ `ara::rest::Uri` shall not provide the user a percent-encoded URI representation by means of its relevant member functions. To convert into percent-encoding `ara::rest::ToString` shall be used. ⌋ (*RS_CM_00300*)

[SWS_REST_01104] URI Mutability ⌈ `ara::rest::Uri` shall be immutable. New versions of Uri objects can only by created via `ara::rest::Uri::Builder`. ⌋ (*RS_CM_00300*)

[SWS_REST_01105] URI Exceptions ⌈ `ara::rest::Uri` shall never throw. ⌋ (*RS_CM_00300*)

[SWS_REST_01106] URI Maximal Length ⌈ `ara::rest::Uri` shall keep a constant length limit of 2048 bytes including potential string terminators and shall be provided via `ara::rest::Uri::LENGTH_MAX`. ⌋(*RS_CM_00300*)

[SWS_REST_01107] Builder API ⌈ `ara::rest::Uri::Builder` shall at least provide the API defined in its API specification. ⌋(*RS_CM_00300*)

[SWS_REST_01108] Builder Exceptions ⌈ `ara::rest::Uri::Builder` may throw. ⌋(*RS_CM_00300*)

[SWS_REST_01109] URI Normalization ⌈ `ara::rest::Normalize` shall normalize `ara::rest::Uri`s according to RFC 3986. ⌋*(RS_CM_00300)*

[SWS_REST_01110] URI Resolve ⌈ `ara::rest::Resolve` shall resolve `ara::rest::Uri`s according to RFC 3986. ⌋*(RS_CM_00300)*

[SWS_REST_01111] URI Relativize ⌈ `ara::rest::Relativize` shall relativize `ara::rest::Uri`s according to RFC 3986. ⌋*(RS_CM_00300)*

[SWS_REST_01112] URI String conversion ⌈ All `ara::rest::Uri` member functions that return string objects shall return a textual representation in non-percent-encoded form. All forms of `ara::rest::ToString` shall return a percent or non-percent encoded form, depending the flag specified as its function argument. ⌋*(RS_CM_00300)*

7.4 UUID

[SWS_REST_01201] UUID type ⌈ To represent UUIDs according to RFC 4122, `ara::rest::Uuid` shall be defined according to its API specification. ⌋*(RS_CM_00300)*

[SWS_REST_01203] UUID exceptions ⌈ `Uuid` shall not throw excecpt during construction. ⌋*(RS_CM_00300)*

7.5 Endpoints

ara::rest provides two (network) endpoints - ara::rest::Client and ara::rest::Server - that manage I/O and resources of RESTful communication. This section specifies the common requirements for both client and server. Specific requirements are specified below.

[SWS_REST_01301] Endpoints ⌈ ara::rest endpoints shall have no notion of the underlying communication protocol. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01302] Endpoint RESTful communication paradigm ⌈ ara::rest endpoints may violate the REST-principle by maintaining state about their respective peers internally, depending on the requirements of the underlying transport protocol used in peer communication. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01303] Endpoint resources ⌈ ara::rest endpoints shall maintain all resources related to interaction with an application and network communication. All reserved resources shall be released upon destruction. Resource access after destruction is undefined. ⌋*(RS_CM_00300)*

[SWS_REST_01304] Endpoint I/O abstraction ⌈ All communication of an application with a remote peer shall performed via `ara::rest::RequestMethod`, `ara::rest::Uri` and the Object Graph Model. ⌋*(RS_CM_00300, RS_CM_00301, RS_CM_00305)*

[SWS_REST_01305] Request Methods ⌈ All accesses into an application data model shall be performed with a small, predefined set of request methods that denote whether an access shall be reading (GET), creating (POST), creating or changing (PUT), deleting (DELETE) or introspection (OPTIONS). Their precise semantics is application-defined. An enumeration `ara::rest::RequestMethod` shall exist which represents these respective access methods as well as ara::rest::operator| for recombination. ⌋ *(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01306] SubscriptionState ⌈ To facilitate events, an implementation shall provide a enumeration `ara::rest::SubscriptionState` whose elements shall have the following semantics: An event subscription "subscribed" if client and server successfully performed a subscription handshake. It is "canceled" if both peers agreed to cancel the subscription. If the state is neither "subscribed" nor "canceled" it shall be "invalid". ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01307] EventPolicy ⌈ Events in ara::rest can be subscribed to in different modes. An implementation shall provide an enumeration `ara::rest::EventPolicy` that shall provide the following semantics for event subscriptions:

- `EventPolicy::kTriggered`: An event is only triggered upon explicit request through `ara::rest::ServerEvent::Notify`. The given time bound limits the number of requests per time frame to at most once.
- `EventPolicy::kPeriodic`: An event is triggered in constant time frames. Explicit triggers of events have not effect.
- `EventPolicy::kTriggered|EventPolicy::kPeriodic`: An event is triggered the end of each given time frame, if and only if a trigger has been issued within this frame at least once. Additional triggers shall have no effect.

⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01308] ShutdownPolicy ⌈ To facilitate a well-defined shutdown of an endpoint, an implementation shall provide an enumeration `ara::rest::ShutdownPolicy` which shall provide the general semantics as described subsequently. Specific semantic differences between client and server are specified below.

- `ShutdownPolicy::Graceful`: Endpoints may shut down "gracefully", which shall allow all ongoing transactions to finish while blocking the caller. Precise semantics of these policies are implementation defined.
- `ShutdownPolicy::Forced`: A forced shutdown shall cancel (terminate) all transations as fast as possible does not block the caller for "unreasonably" (implementation-defined) long period of time. During a forced shutdown, further network I/O is not allowed. A forced shutdown shall not allocate new memory resources. Apart from this semantics of these policies are implementation defined.

⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01311] StartupPolicy ⌈ To facilitate a well-defined start-up of a server endpoint, an implementation shall provide an enumeration `ara::rest::StartupPolicy` which shall provide the following general semantics:

- `StartupPolicy::kAttached`: the endpoint shall block its calling context during startup and regular operation until it is explicitly shut down.
- `StartupPolicy::kDetached`: the endpoint shall not block its calling context during startup and regular operation.

⌋*(RS_CM_00300, RS_CM_00301)* -

[SWS_REST_01312] Reply Header ⌈ A `ara::rest::ReplyHeader` shall exist which shall provide mutable access to `ara::rest::RequestMethod` and `ara::rest::Uri` of a `ara::rest::Reply` or `ara::rest::ServerReply`. The precise semantics of `ara::rest::ReplyHeader::GetStatus` and `ara::rest::ReplyHeader::SetStatus` as well as `ara::rest::ReplyHeader::GetUri` and `ara::rest::ReplyHeader::SetUri` is protocol-dependent. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01313] Request Header ⌈ A `ara::rest::RequestHeader` shall exist which shallx provide mutable access to `ara::rest::RequestMethod` and `ara::rest::Uri` of a `ara::rest::Request` or `ara::rest::ServerRequest`. ⌋*(RS_CM_00300, RS_CM_00301)*

7.6 Client

ara::rest requires the existence of a type ara::rest::Client which represents the client network endpoint for RESTful communication.

An ara::rest client represents a single "session" with potentially multiple servers. A client is not bound to a particular remote endpoint.

[SWS_REST_01401] Client Resources ⌈ `ara::rest::Client` maintains all resources related to communication with a peer. Upon destruction, all such resources shall be released. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01402] Client Interface ⌈ An `ara::rest::Client` shall at least provide the interface as specified in its API specification and according to the detailed semantics specified there. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01403] Client Instances ⌈ `ara::rest::Client` shall not be copyable. It shall be movable. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01404] Client Multiplicity ⌈ `ara::rest::Client` shall not be bound to one particular peer. As such, a client is not bound to a particular server instance. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01405] Client Asynchronicity ⌈ `ara::rest::Client` shall be able to operate according to its semantic requirements without employing multi-threading.

Consequently, a user shall be aware of the fact that invoking blocking functions such as `ara::rest::Task<T>::wait()` may degrade service quality. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01406] Client Concurrency ⌈ It is implementation-defined whether `ara::rest::Client` is operating concurrently. Consequently, a user shall be aware of the fact that invoking blocking functions may degrade service quality. ⌋ *(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01407] Client Construction ⌈ `ara::rest::Client` shall be provided with a unique identifier that selects an implementation-defined configuration record. ⌋ *(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01408] Client Startup ⌈ `ara::rest::Client` shall be ready for transmission to remote hosts if construction succeeded. In particular, a client may throw if construction fails. Otherwise, an error shall be indicated via `ara::rest::Client::GetError`. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01409] Client Shutdown ⌈ `ara::rest::Client` shall be stopped via `ara::rest::Client::Stop` according to the semantics specified in its API specification and SWS_REST_01311. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01410] Client Stop ⌈ `ara::rest::Client::Stop` shall be thread-safe. After it has been invoked, calling `ara::rest::Client::Send`, `ara::rest::Client::Subscribe` or `ara::rest::Client::Stop` shall have no effect. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01411] Client Stop Task ⌈ `ara::rest::Client::Stop` returns a `Task` which shall only complete once shutdown succeeded. It may throw otherwise. ⌋ *(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01412] Client Sending ⌈ `ara::rest::Client::Send` shall transmit a `ara::rest::Client::Request` to a peer specified via the given request URI. The call shall never block even for multiple requests to the same peer. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01413] Client Peer Addressing ⌈ `ara::rest::Client::Send` shall transmit a request to the peer specified by the uri authority part via `ara::rest::Uri::GetHost` and `ara::rest::Uri::GetPort`. If not such information is available a client shall fall back to the implemention-defined configuratiomn record uniquely identified by the identifier passed to `ara::rest::Client::Client`. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01414] Client Events ⌈ An implementation shall provide `ara::rest::Client::Subscribe` to create `ara::rest::Event` instances which represent a single event subscription. `Subscribe` shall never block. ⌋ *(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01415] Client Event Uri ⌈ The URI passed to `ara::rest::Client::Subscribe` denotes an entity to subscribe to. URI query parameters shall be taken into account. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01416] Client Event Policy ⌈ An implementation shall honor the event policy passed to according to SWS_REST_01307 along with the time bound specified. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01417] Client Event Notification ⌈ An instance of `ara::rest::Client::NotificationHandlerType` shall be passed to `ara::rest::Client::Subscribe` for the asynchronous reception of event notification. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01418] Client Event Status ⌈ An instance of `ara::rest::Client::SubscriptionStateHandlerType` may be passed to `ara::rest::Client::Subscribe` for the asynchronous changes to the event status. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01419] Client Exception ⌈ `ara::rest::Client` itself may throw during construction and during regular operation only for errors that leave the client in an undefined state. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01420] Client Error ⌈ All errors not leaving a client in an undefined state shall be indicated via `ara::rest::Client::GetError` and `ara::rest::ObserveError`. In particular this concerns all I/O related errors. The concrete definition of `std::error_code` is implementation defined. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01421] Client Request ⌈ `ara::rest::Request` shall be non-copyable and shall maintain all resources related to issue a data request to a peer. In particular, the lifetime of a request object and the lifetime of the objects passed to it for transmission shall be independent. This implies copying or moving. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01422] Client Reply ⌈ `ara::rest::Reply` shall be non-copyable and shall maintain all resources related to receiption of data from a peer. The lifetime of all resources referenced by a reply object is bound to the lifetime of the reply itself. This implies copying or moving. ⌋*(RS_CM_00300, RS_CM_00301)*

7.7 Server

ara::rest requires the existence of a type ara::rest::Server which represents the server network endpoint for RESTful communication along with its components.

[SWS_REST_01501] Server Resources ⌈ `ara::rest::Server` maintains all resources related to communication with a peer. Upon destruction, all such resources shall be released. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01502] Server Interface ⌈ An `ara::rest::Server` shall at least provide the interface as specified in the API specification and according to the detailed semantics specified there. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01503] Server Instances ⌈ `ara::rest::Server` shall not be copyable. It shall be movable. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01504] Server Multibinding ⌈ `ara::rest::Server` shall be able to handle multiple transport protocol bindings. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01505] Server Multiplicity ⌈ `ara::rest::Server` shall be able to handle multiple peers concurrently over potentially multiple transport protocols. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01506] Server Asynchronicity ⌈ `ara::rest::Server` shall be able to operate according to its semantic requirements without employing multi-threading. Consequently, an application developer shall be aware of the fact that invoking blocking functions such as `ara::rest::Task<T>::wait()` may degrade service quality. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01507] Server Concurrency ⌈ It is implementation-defined whether `ara::rest::Server` is operating concurrently. Consequently, an application developer shall be aware of the fact that invoking blocking functions may degrade service quality. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01508] Server Construction ⌈ `ara::rest::Server` shall be provided with a unique identifier that selects an implementation-defined configuration record. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01509] Server Startup ⌈ `ara::rest::Server` shall not be operational until `ara::rest::Server::Start` is invoked explicitly. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01510] Server Startup Policies ⌈ `ara::rest::Server` can be started in two operational modes according to SWS_REST_01311. Servicing shall not start until the `Task` returned by `Send` signals completion. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01511] Server Startup Task ⌈ `ara::rest::Server::Start` returns a `Task` which shall only complete once startup succeeded. It may throw otherwise. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01512] Server Shutdown ⌈ `ara::rest::Server` shall be stopped via `ara::rest::Server::Stop` according to the semantics specified in its API specification and SWS_REST_01311. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01513] Server Restart ⌈ `ara::rest::Server` shall leave its instances in a well-defined state after shutdown such that a subsequent startup remains feasible. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01514] Server Stop ⌈ `ara::rest::Server::Stop` shall be thread-safe. After it has been invoked, no new requests shall be accepted. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01515] Server Stop Task ⌈ `ara::rest::Server::Stop` returns a `Task` which shall only complete once shutdown succeeded. It may throw otherwise. ⌋ *(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01516] Server User Message Notification ⌈ `ara::rest::Server` shall invoke the user-defined handler function specified in its destructor at least as early as a valid message header has been received. It is implementation-defined whether the handler is activated only the entire message has been received. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01517] Server Event Subscriptions ⌈ `ara::rest::Server` shall accept event subscriptions without user-intervention. It shall inform the application of new subscriptions or subscription state changed via `ara::rest::Server::ObserveSubscriptions`. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01518] Server Event Data Request ⌈ Depending on the subscription parameters, event notifications shall be issued at certain points in time. `ara::rest::Server` shall issue a regular GET request to the application for the URI and the respective query parameters as supplied by the subscriber. To the application such a request shall be indistinguishhable from regular requests. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01519] Server Event Object ⌈ Each subscription shall be represented by a unique `ara::rest::ServerEvent` object which shall maintain all resources related to this subscription. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01522] Server Event Object Destruction ⌈ Upon destruction of a `ara::rest::ServerEvent` object the corresponding event subscription shall be terminated instantly, unilaterally and all managed server-side resources shall be released. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01523] Server Event Object Subscription Cancellation ⌈ `ara::rest::ServerEvent::Unsubscribe` shall cancel the subscription in accordance to the rules of the underlying protocols implementing the event mechanism. The function shall return a task waiting for the cancelation handshake with the respective peer to complete. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01524] Server Event Object Re-Subscription ⌈ Once `ara::rest::ServerEvent::Unsubscribe` is called, resubscription on the same object shall not be allowed. Each subscription shall yield a unique event object. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01525] Server Event Object Notify ⌈ `ara::rest::ServerEvent::Notify` shall notify its corresponding `ara::rest::Server` instance of potential updates to the data referenced by the URI subscribed to. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01526] Server Event Object Notify Semantics ⌈ Following a call to `ara::rest::ServerEvent::Notify`, it is the responsibility of

ara::rest::Server to obain the actual data from the related URI and to decide whether an actual event notification message shall be transmitted to the subscriber. ⌋(*RS_CM_00300*, *RS_CM_00301*)

[SWS_REST_01527] Server Event Observation ⌈ ara::rest::ServerEvent::ObserveSubscriptions shall be used by an application to register user-defind handler functions to be called on new subscriptions and state changed to existing subscriptions. ⌋(*RS_CM_00300*, *RS_CM_00301*)

[SWS_REST_01528] Server Event Creation ⌈ An instance of ara::rest::ServerEvent::SubscriptionHanderType that is registered via ara::rest::ServerEvent::ObserveSubscriptions is called by ara::rest::Server for each newly accepted subscription. It is the reponsibility of the server object to create an instance of ara::rest::ServerEvent and pass it to the user via the handler function. ⌋(*RS_CM_00300*, *RS_CM_00301*)

[SWS_REST_01529] Server Event Status Changes ⌈ An instance of ara::rest::ServerEvent::SubscriptionStateHanderType that is registered via ara::rest::ServerEvent::ObserveSubscriptions is called by ara::rest::Server for each detected state change of a subscription according to ara::rest::SubscriptionState. ⌋(*RS_CM_00300*, *RS_CM_00301*)

[SWS_REST_01530] Server Exception ⌈ ara::rest::Server itself may throw during construction and during regular operation only for errors that leave the server in an undefined state. ⌋(*RS_CM_00300*, *RS_CM_00301*)

[SWS_REST_01531] Server Error ⌈ All errors not leaving a server in an undefined state shall be indicated via ara::rest::Server::GetError and ara::rest::ObserveError. In particular this concerns all I/O related errors. The concrete definition of std::error_code is implementation defined. ⌋ (*RS_CM_00300*, *RS_CM_00301*)

[SWS_REST_01532] Server Requests and Responses ⌈ ara::rest::ServerRequest and ara::rest::ServerResponse shall be instantiated by ara::rest::Server for each received request message. Both objects shall manage all resource related to the current transaction. An application shall neither construct or destroy these objects. ⌋(*RS_CM_00300*, *RS_CM_00301*)

[SWS_REST_01533] Server Request Semantics ⌈ A reference of ara::rest::ServerRequest may be passed to the user-defined handler function as soon as a valid message header (according to implementation-defined transport protocol requirements) has been accepted. Access to message payloads shall be asynchronous via ara::rest::ServerRequest::GetObject or ara::rest::ServerRequest::ReleaseObject respectively. ⌋(*RS_CM_00300*, *RS_CM_00301*)

[SWS_REST_01534] Server Reply Semantics ⌈ A reference of ara::rest::ServerReply may be passed to the user-defined handler function along with a reference to its corresponding ara::rest::ServerRequest.

The instance shall be initialized such that it represents an empy reply message. ⌋ *(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01535] Server Reply Send Semantics ⌈ `ara::rest::ServerReply::Send` shall trigger the transmission of a message to sender of the respective request. Upon destruction a `ara::rest::ServerReply` shall call `Send` implicitly. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01536] Server Reply Multiple Send Semantics ⌈ `ara::rest::ServerReply::Send` shall not be called multiple times. ⌋ *(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01537] Server Reply Redirect Semantics ⌈ `ara::rest::ServerReply::Redirect` shall issue a protocol-dependent client redirection which shall cause a client to repeat the request to the endpoint indicated via `ara::rest::ReplyHeader::SetUri`. ⌋*(RS_CM_00300, RS_CM_00301)*

[SWS_REST_01538] Server Reply Send/Redirect Interaction ⌈ `ara::rest::ServerReply::Send` shall not be called after `ara::rest::ServerReply::Redirect`, and vice versa. ⌋*(RS_CM_00300, RS_CM_00301)*

7.8 Routing

Routing is multiplexing of server requests depending on the valuation of `ara::rest::RequestMethod` and `ara::rest::Uri`. It connects RESTful API accesses with an underlying execution or data model. In the following the components for routing are specified bottom-up.

7.8.1 Patterns

[SWS_REST_01601] Pattern API ⌈ An implementation shall provide a type `ara::rest::Pattern` that satisfies at least the interface and basic semantics as defined in the API specification below. ⌋*(RS_CM_00300)*

[SWS_REST_01602] Pattern Syntax ⌈ `ara::rest::Pattern` represents a pattern string to match against `ara::rest::Uri::Path` instances. A pattern string may be composed of all valid URI path characters as well as characters “*” and “**” (wildcards). ⌋*(RS_CM_00300)*

[SWS_REST_01603] Pattern General Wildcard semantics ⌈ Wildcards shall match URI path segments, not general string characters in a URI. In other words, URI path segment delimiters "/" restrict matching. ⌋*(RS_CM_00300)*

[SWS_REST_01604] Pattern Single Wildcard semantics ⌈ Wildcard character “*” shall match exactly a single URI path segment.

Example: Pattern `/foo/*/bar` shall match URI path `/foo/baz/bar`. It shall not match URI path `/foo/baz/bab/baz` nor `/foo/baz`. ⌋(*RS_CM_00300*)

[SWS_REST_01605] Pattern Double Wildcard semantics ⌈ Wildcard characters "**" shall match any number of path segments (including none).

Example: Pattern `/foo/**` shall match URI path `/foo`, `/foo/baz` and `/foo/baz/bar`. It shall not match URI path `/foo/bar/baz` nor `/foo/baz`. ⌋(*RS_CM_00300*)

[SWS_REST_01606] Pattern Comparability ⌈ `ara::rest::Pattern` shall be equal-to, unequal-to and less-than comparable. ⌋(*RS_CM_00300*)

[SWS_REST_01607] Pattern Order Criterion ⌈ `ara::rest::Pattern` shall be less-than comparable and form a lexicographic order is which URI path segments are considered "characters" in left-to-right order.

Example: It holds that `"car" < "car/window"` which means `"car"` before `"/car/window"`. For wildcards the order `"**" < "*" < "anything else"` holds. ⌋(*RS_CM_00300*)

7.8.2 Match

[SWS_REST_01608] Match API ⌈ An implementation shall provide a type `ara::rest::Match` that satisfies at least the interface and basic semantics as defined in the API specification below. ⌋(*RS_CM_00300*)

[SWS_REST_01609] Match Creation ⌈ When matching URIs against patterns, then for every path segment matched against either single wildcard "*" or double wildcard "**" an instance of `ara::rest::Match` shall be created. ⌋(*RS_CM_00300*)

[SWS_REST_01610] Match Access As String ⌈ `ara::rest::Match::Get` shall return a view of the matched URI segment. ⌋(*RS_CM_00300*)

[SWS_REST_01611] Match Access As Type ⌈ `ara::rest::Match::GetAs` shall convert the match into type T specified by the template parameter. Type T shall be InputStreamable: there shall exist a global function `std::istream& operator»(std::istream&, const T&)` that performs lexical conversion. If conversion fails, GetAs() shall throw `std::invalid_argument`. Function overload `GetAs(T&&)` shall perform the same conversion but instead of throwing it returns the function argument if conversion fails. ⌋(*RS_CM_00300*)

7.8.3 Matches

[SWS_REST_01612] Matches API ⌈ An implementation shall provide a type `ara::rest::Matches` that satisfies at least the interface and basic semantics as defined in the API specification below. ⌋(*RS_CM_00300*)

[SWS_REST_01613] Matches Creation ⌈ When matching URIs against patterns, then a `ara::rest::Matches` instance shall be created that shall contain all `ara::rest::Match` objects created. Matches shall own the Match instances and release them upon destruction. ⌋*(RS_CM_00300)*

[SWS_REST_01614] Matches of General Path Segments ⌈ Matches shall not represent non-wildcard path segments.

Example: "`/foo/*/bar`" shall only create a single Match. ⌋*(RS_CM_00300)*

7.8.4 Route

[SWS_REST_01615] Route API ⌈ An implementation shall provide a type `ara::rest::Route` that satisfies at least the interface and basic semantics as defined in the API specification below. ⌋*(RS_CM_00300)*

[SWS_REST_01616] Route Semantics ⌈ `ara::rest::Route` shall call the user-defined function of type `ara::rest::Route::RouteHandlerType` provided as a constructor argument if `ara::rest::RequestMethod` and `ara::rest::Pattern` provided as constructor arguments match the request method and URI of the `ara::rest::ServerRequest` passed to `ara::rest::Route::operator()`. ⌋*(RS_CM_00300, RS_CM_00310)*

[SWS_REST_01617] Route Match ⌈ If a route matches, the `ara::rest::Route::RouteHandlerType` specified via its constructor shall be called along with the respective request and reply objects, and a set of `ara::rest:Matches` that represent wildcard matches of the request URI. ⌋*(RS_CM_00300, RS_CM_00310)*

[SWS_REST_01618] Route Return Values ⌈ `ara::rest::Route::operator()` returns values of type `ara::rest::Route::Upshot`. The respective return values shall have the following effect on the routing described later:

- `Accept`: `ara::rest::Router` shall not search for further matches.
- `Yield`: `ara::rest::Router` shall select the next matching route (multiple routes may match) or the default handler function in case of no next matching route.
- `Default`: `ara::rest::Router` shall execute its default handler function (specified below).

⌋*(RS_CM_00300, RS_CM_00310)*

[SWS_REST_01619] Route Comparability ⌈ `ara::rest::Route` shall be equal-to, unequal-to and less-than comparable. ⌋*(RS_CM_00300)*

[SWS_REST_01620] Route Order Criterion ⌈ Routes shall compare less-than in lexicographic order such that the given `ara::rest::Uri::Path` compare first, the given `ara::rest::RequestMethod` compare last. ⌋*(RS_CM_00300)*

[SWS_REST_01621] Route Order Criterion for RequestMethod ⌈ While `ara::rest::Uri` is ordered lexicographically, `ara::rest::RequestMethod` shall be less-than comparable for route matching according to the following rule: The order of request methods is lexicographic with each enumerator representing a character of a string concatenated by operator |. Therefore, for single "digits" it holds that $kGET < kPOST < \ldots$ etc according to their underlying numeric values. For multiple "digits" it holds that - for example - $kGET < kGET \mid kPOST < kGET \mid kPUT < \ldots < kAny$. But note that $kGET \mid kPUT == kPUT \mid kGET$. In words, the most precise specifiers (singleton request methods) have precedence over the less precise specificiation of sets of enumerators. This is not the same as simply taking the underlying numeric value of the OR-combined enumerators. ⌋*(RS_CM_00300)*

7.8.5 Router

[SWS_REST_01622] Router API ⌈ An implementation shall provide a type `ara::rest::Router` that satisfies at least the interface and basic semantics as defined in the API specification below. ⌋*(RS_CM_00300)*

[SWS_REST_01623] Router Semantics ⌈ `ara::rest::Router` shall maintain an ordered set of routes and it shall find a matching routes by comparing the request method and URI components of a given `ara::rest::ServerRequest` against each given `ara::rest::Route` in the set. ⌋*(RS_CM_00300)*

[SWS_REST_01624] Router Usage ⌈ A router is a pre-defined request handler type which may be passed to the constructor of `ara::rest::Server` in order to demultiplex messaging. ⌋*(RS_CM_00300, RS_CM_00310)*

[SWS_REST_01625] Router Route Order ⌈ Routes shall be matched according to their order criterion as defined above from “smallest” to “greatest”. Incomparable routes shall be matched in the order of insertion into a router. ⌋*(RS_CM_00300)*

[SWS_REST_01626] Router Route Match ⌈ If a route matches, its `ara::rest::Route::operator()` shall be called. ⌋*(RS_CM_00300)*

[SWS_REST_01627] Router Route Skipping ⌈ If a `ara::rest::Route::operator()` returns an `ara::rest::Route::Upshot` value of “Yield”, then a router shall call the next match. ⌋*(RS_CM_00300)*

[SWS_REST_01628] Router Default Route ⌈ If no route matches, the request handler function specified via `ara::rest::Router::SetDefaultHandler` shall be called, if it has been set. If no such handler is set, the router shall silently ignore the request currently under inspection. ⌋*(RS_CM_00300)*

[SWS_REST_01629] Router Route Defaulting ⌈ If a `ara::rest::Route::operator()` returns an `ara::rest::Route::Upshot` value of “Default”, then the request handler function specified via `ara::rest::Router::SetDefaultHandler` shall be called, if it has been

set. If no such handler is set, the router shall silently ignore the request silently under inspection. ⌋(*RS_CM_00300*)

7.9 Object Graph Model

ara::rest message payloads are always represented as object graph data-structures (object graph models; OGM). OGM serve as a universal exchange format.

[SWS_REST_01701] OGM Representation ⌈ All transport-specific message payloads shall be converted to and from OGM for communication and interaction with a service application. ⌋(*RS_CM_00300*)

[SWS_REST_01702] OGM Syntax ⌈ An implementation shall provide a basic set of data types whose interfaces shall at least satisfy interfaces and basic functionalities according to their API specifications provided by `ara::rest::ogm::Node` `ara::rest::ogm::Field` `ara::rest::ogm::Value` `ara::rest::ogm::String` `ara::rest::ogm::Int` `ara::rest::ogm::Real` `ara::rest::ogm::Array` and `ara::rest::ogm::Object` ⌋(*RS_CM_00300, RS_CM_00305*)

[SWS_REST_01703] OGM Semantics: Internal Ownership ⌈ OGM instances are always trees. Each parent node shall uniquely own its children. ⌋(*RS_CM_00300, RS_CM_00305*)

[SWS_REST_01704] OGM Semantics: External Ownership ⌈ Ownership of an OGM shall never be shared within an application. The lifetime of an OGM shall strictly be bound to the lifetime of the `Pointer` instance owning an OGM after construction. ⌋(*RS_CM_00300, RS_CM_00305*)

[SWS_REST_01705] OGM Construction Semantics ⌈ OGM node constructors are non-public. To construct OGM nodes their respective static `Make()` member functions shall be used which return a `Pointer` instance that own the OGM just created. Only leaf-types in the type hierarchy may be insantiated. ⌋(*RS_CM_00300, RS_CM_00305*)

[SWS_REST_01706] OGM Destruction Semantics ⌈ If an owner (a parent node or a `Pointer` holding a reference to an OGM) is destroyed, all owned objects shall be destroyed too. ⌋(*RS_CM_00300, RS_CM_00305*)

[SWS_REST_01707] OGM Copy Semantics ⌈ An OGM cannot be copied directly or implicitly. To copy an OGM `ara::rest::ogm::Copy` shall be used. ⌋(*RS_CM_00300, RS_CM_00305*)

[SWS_REST_01708] OGM Move Semantics For Owners ⌈ To move an OGM, its owning `Pointer` instance shall be moved. ⌋(*RS_CM_00300, RS_CM_00305*)

[SWS_REST_01709] OGM Move Semantics For Non-owners: Release ⌈ To obtain ownership of subtrees, member functions with prefix "`Release`" shall be called on the owning OGM node. Only node types that represent sets (`ara::rest::ogm::Array`

and `ara::rest::ogm::Object`) may be empty and therefore have a "`Release`" functionality. ⌋*(RS_CM_00300, RS_CM_00305)*

[SWS_REST_01710] OGM Move Semantics For Non-owners: Replace ⌈ For consistency, some node types have "`Replace`" functions instead of "`Release`". To take on ownership of a subtree, it shall be replaced by a suitable replacement object. ⌋ *(RS_CM_00300, RS_CM_00305)*

[SWS_REST_01711] OGM Iterator Semantics ⌈ Some OGM node types are iterable. Iterators shall not expose any internal data management. The respective iterator types shall provide C++ references directly to the referenced set elements ⌋*(RS_CM_00300, RS_CM_00305)*

[SWS_REST_01712] OGM String encoding ⌈ `ara::rest::ogm::String` shall support UTF-8. ⌋*(RS_CM_00300, RS_CM_00305)*

[SWS_REST_01713] OGM Int precision ⌈ `ara::rest::ogm::Int` is signed shall be at least as precise as a C++ `std::int64_t` integer type. ⌋*(RS_CM_00300, RS_CM_00305)*

[SWS_REST_01714] OGM Real precision ⌈ `ara::rest::ogm::Int` shall be at least as precise as a C++ `double` floating point type. ⌋*(RS_CM_00300, RS_CM_00305)*

[SWS_REST_01715] OGM Visit ⌈ `ara::rest::ogm::Visit` implements the visit pattern and shall expose the actual type of an OGM node from a reference to any of its parents in the OGM node type hierarchy. ⌋*(RS_CM_00300, RS_CM_00305)*

8 API specification

This chapter contains the formal API documentation of ara::rest.

8.1 ara::rest::Allocator

[SWS_REST_02000] ⌈`ara::rest::Allocator` class shall be declared in the `ara/rest/allocator.h` header file:

```
1 class ara::rest::Allocator;
```

⌋(*RS_CM_00300*)

8.1.1 Allocator

Service name:	ara::rest::Allocator::Allocator
Type:	`Member function`
Syntax:	`ara::rest::Allocator::Allocator()=default`
Function param:	None
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/allocator.h
Class:	ara::rest::Allocator
Description:	Constructs this object.

Table 8.1: ara::rest::Allocator::Allocator

[SWS_REST_02001] ara::rest::Allocator::Allocator ⌈Table 8.1 describes the interface `ara::rest::Allocator::Allocator`.⌋(*RS_CM_00300*)

8.1.2 ~Allocator

Service name:	ara::rest::Allocator::~Allocator
Type:	`Member function`
Syntax:	`virtual ara::rest::Allocator::~Allocator()`
Function param:	None
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/allocator.h
Class:	ara::rest::Allocator
Description:	Destroys this object.

Table 8.2: ara::rest::Allocator::~Allocator

[SWS_REST_02002] ara::rest::Allocator::~Allocator ⌈Table 8.2 describes the interface `ara::rest::Allocator::~Allocator`.⌋*(RS_CM_00300)*

8.1.3 allocate

Service name:	ara::rest::Allocator::allocate	
Type:	`Member function`	
Syntax:	`void* ara::rest::Allocator::allocate(std::size_t bytes, std::size_t alignment=alignof(std::max_align_t))`	
Function param:	bytes	desired size of the memory area to be allocated
Function param:	alignment	alignment of the memory area
Return value:	a pointer to the allocated memory area	
Exceptions:	Implementation-defined	
Header file:	ara/rest/allocator.h	
Class:	ara::rest::Allocator	
Description:	Allocates a memory area.	

Table 8.3: ara::rest::Allocator::allocate

[SWS_REST_02003] ara::rest::Allocator::allocate ⌈Table 8.3 describes the interface `ara::rest::Allocator::allocate`.⌋*(RS_CM_00300)*

8.1.4 deallocate

Service name:	ara::rest::Allocator::deallocate	
Type:	`Member function`	
Syntax:	`void ara::rest::Allocator::deallocate(void *p, std::size_t bytes, std::size_t alignment=alignof(std::max_align_t))`	
Function param:	p	pointer to the allocated memory area
Function param:	bytes	size of the allocated memory area
Function param:	alignment	alignment of allocated memory area
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/allocator.h	
Class:	ara::rest::Allocator	
Description:	Releases a memory area.	

Table 8.4: ara::rest::Allocator::deallocate

[SWS_REST_02004] ara::rest::Allocator::deallocate ⌈Table 8.4 describes the interface `ara::rest::Allocator::deallocate`.⌋*(RS_CM_00300)*

8.1.5 is_equal

Service name:	ara::rest::Allocator::is_equal	
Type:	`Member function`	
Syntax:	`bool ara::rest::Allocator::is_equal(const Allocator &alloc) const`	
Function param:	alloc	an allocator to compare against
Return value:	true if the two allocators compare equal	
Exceptions:	`noexcept`	
Header file:	ara/rest/allocator.h	
Class:	ara::rest::Allocator	
Description:	Tests whether two allocators are equal. Allocators are equal if memory allocated by one can be deallocated by the other.	

Table 8.5: ara::rest::Allocator::is_equal

[SWS_REST_02005] ara::rest::Allocator::is_equal ⌈Table 8.5 describes the interface `ara::rest::Allocator::is_equal`.⌋*(RS_CM_00300)*

8.2 ara::rest::Client

[SWS_REST_02006] ⌈`ara::rest::Client` class shall be declared in the `ara/rest/client.h` header file:

```
1 class ara::rest::Client;
```

⌋*(RS_CM_00300, RS_CM_00301)*

8.2.1 NotificationHandlerType

Name:	`NotificationHandlerType`
Type:	`Member type alias`
Syntax:	`using ara::rest::Client::NotificationHandlerType = void(const ogm::Object&)`
Header file:	ara/rest/client.h
Class:	ara::rest::Client
Description:	Denotes a callback function for notifications.

Table 8.6: ara::rest::Client::NotificationHandlerType

[SWS_REST_02007] NotificationHandlerType ⌈Table 8.6 describes the type alias `ara::rest::Client::NotificationHandlerType`.⌋*(RS_CM_00300, RS_CM_00301)*

8.2.2 SubscriptionStateHandlerType

Name:	`SubscriptionStateHandlerType`

Type:	`Member type alias`
Syntax:	`using ara::rest::Client::SubscriptionStateHandlerType = void(const Event&, SubscriptionState)`
Header file:	ara/rest/client.h
Class:	ara::rest::Client
Description:	Denotes a callback to call if subscription status changes.

Table 8.7: ara::rest::Client::SubscriptionStateHandlerType

[SWS_REST_02008] SubscriptionStateHandlerType ⌈Table 8.7 describes the type alias `ara::rest::Client::SubscriptionStateHandlerType`.⌋ *(RS_CM_00300, RS_CM_00301)*

8.2.3 Client

Service name:	ara::rest::Client::Client	
Type:	`Member function`	
Syntax:	`ara::rest::Client::Client(const String &conf_id, Allocator *alloc=GetDefaultAllocator())`	
Function param:	conf_id	identifier for a corresponding configuration record
Function param:	alloc	allocator for dynamic memory
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/client.h	
Class:	ara::rest::Client	
Description:	Constructs a client.	

Table 8.8: ara::rest::Client::Client

[SWS_REST_02009] ara::rest::Client::Client ⌈Table 8.8 describes the interface `ara::rest::Client::Client`.⌋ *(RS_CM_00300, RS_CM_00301)*

8.2.4 Client

Service name:	ara::rest::Client::Client
Type:	`Member function`
Syntax:	`ara::rest::Client::Client(const Client &)=delete`
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/client.h
Class:	ara::rest::Client
Description:	non-copy-constructible

Table 8.9: ara::rest::Client::Client

[SWS_REST_02010] ara::rest::Client::Client ⌈Table 8.9 describes the interface `ara::rest::Client::Client`.⌋ *(RS_CM_00300, RS_CM_00301)*

8.2.5 operator=

Service name:	ara::rest::Client::operator=
Type:	`Member function`
Syntax:	`Client& ara::rest::Client::operator=(const Client &)=delete`
Return value:	a value of type `Client &`
Exceptions:	Implementation-defined
Header file:	ara/rest/client.h
Class:	ara::rest::Client
Description:	non-copy-assignable

Table 8.10: ara::rest::Client::operator=

[SWS_REST_02011] ara::rest::Client::operator= ⌈Table 8.10 describes the interface `ara::rest::Client::operator=`.⌋ *(RS_CM_00300, RS_CM_00301)*

8.2.6 Stop

Service name:	ara::rest::Client::Stop	
Type:	`Member function`	
Syntax:	`Task<void> ara::rest::Client::Stop(ShutdownPolicy policy=ShutdownPolicy::kGraceful)`	
Function param:	policy	shutdown policy
Return value:	a task waiting for shutdown to complete	
Exceptions:	Implementation-defined	
Header file:	ara/rest/client.h	
Class:	ara::rest::Client	
Description:	Requests a client shutdown. If shutting down gracefully, the client waits for all transactions to finish. If not, then all connections must be terminated instantly.	

Table 8.11: ara::rest::Client::Stop

[SWS_REST_02012] ara::rest::Client::Stop ⌈Table 8.11 describes the interface `ara::rest::Client::Stop`.⌋ *(RS_CM_00300, RS_CM_00301)*

8.2.7 Send

Service name:	ara::rest::Client::Send	
Type:	`Member function`	
Syntax:	`Task<Pointer<Reply> > ara::rest::Client::Send(const Request &req)`	
Function param:	req	a request message
Return value:	a task waiting for the corresponding reply	
Exceptions:	Implementation-defined	
Header file:	ara/rest/client.h	
Class:	ara::rest::Client	

Description:	Issues a request to a peer. Issues a request to the peer either specified in the client configuration record or the URI of the request. The configuration record is identified by the id specified in the Client constructor. If Uri::Authority is set, it overwrites the configuration record.

Table 8.12: ara::rest::Client::Send

[SWS_REST_02013] ara::rest::Client::Send ⌈Table 8.12 describes the interface `ara::rest::Client::Send`.⌋*(RS_CM_00300, RS_CM_00301)*

8.2.8 Subscribe

Service name:	ara::rest::Client::Subscribe	
Type:	`Member function`	
Syntax:	`Task<Event> ara::rest::Client::Subscribe(const Uri &uri, EventPolicy policy, duration_t time, const Function< NotificationHandlerType > &notify, const Function< SubscriptionStateHandlerType > &state={})`	
Function param:	uri	the event to subscribe to
Function param:	policy	the notification policy
Function param:	time	time bound as a parameter of the notification policy
Function param:	notify	user-defined event notification handler function
Function param:	state	user-define subscription state observer function
Return value:	a task waiting for the Event construction and subscription Reply.	
Exceptions:	Implementation-defined	
Header file:	ara/rest/client.h	
Class:	ara::rest::Client	
Description:	Performs an event subscription. An event is uniquely identified by its Uri. A subscription to an event means that if preconditions are met a notification is issued whose message payload is identical to the result set obtained by issuing a GET request on the Uri.	

Table 8.13: ara::rest::Client::Subscribe

[SWS_REST_02014] ara::rest::Client::Subscribe ⌈Table 8.13 describes the interface `ara::rest::Client::Subscribe`.⌋*(RS_CM_00300, RS_CM_00301)*

8.2.9 GetError

Service name:	ara::rest::Client::GetError
Type:	`Member function`
Syntax:	`std::error_code ara::rest::Client::GetError() const`
Function param:	None
Return value:	status of the client
Exceptions:	`noexcept`
Header file:	ara/rest/client.h
Class:	ara::rest::Client
Description:	Obtain client status.

Table 8.14: ara::rest::Client::GetError

[SWS_REST_02015] ara::rest::Client::GetError ⌈Table 8.14 describes the interface `ara::rest::Client::GetError`.⌋*(RS_CM_00300, RS_CM_00301)*

8.2.10 ObserveError

Service name:	ara::rest::Client::ObserveError	
Type:	`Member function`	
Syntax:	`void ara::rest::Client::ObserveError(const Function< void(std::error_code)> &hnd)`	
Function param:	hnd	user-defined handler function to called on status changes
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/client.h	
Class:	ara::rest::Client	
Description:	Observe status changes.	

Table 8.15: ara::rest::Client::ObserveError

[SWS_REST_02016] ara::rest::Client::ObserveError ⌈Table 8.15 describes the interface `ara::rest::Client::ObserveError`.⌋*(RS_CM_00300, RS_CM_00301)*

8.3 ara::rest::Event

[SWS_REST_02017] ⌈`ara::rest::Event` class shall be declared in the `ara/rest/client.h` header file:

```
1 class ara::rest::Event;
```

⌋*(RS_CM_00300)*

8.3.1 Event

Service name:	ara::rest::Event::Event
Type:	`Member function`
Syntax:	`ara::rest::Event::Event(const Event &)=delete`
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/client.h
Class:	ara::rest::Event
Description:	Non-copyable.

Table 8.16: ara::rest::Event::Event

[SWS_REST_02018] ara::rest::Event::Event ⌈Table 8.16 describes the interface `ara::rest::Event::Event`.⌋*(RS_CM_00300)*

8.3.2 operator=

Service name:	ara::rest::Event::operator=
Type:	`Member function`
Syntax:	`Event& ara::rest::Event::operator=(const Event &)=delete`
Return value:	a value of type `Event &`
Exceptions:	Implementation-defined
Header file:	ara/rest/client.h
Class:	ara::rest::Event
Description:	Non-copy-assignable.

Table 8.17: ara::rest::Event::operator=

[SWS_REST_02019] ara::rest::Event::operator= ⌈Table 8.17 describes the interface `ara::rest::Event::operator=`.⌋*(RS_CM_00300)*

8.3.3 Unsubscribe

Service name:	ara::rest::Event::Unsubscribe
Type:	`Member function`
Syntax:	`Task<bool> ara::rest::Event::Unsubscribe()`
Function param:	None
Return value:	a task waiting for cancellation which returns true on success.
Exceptions:	Implementation-defined
Header file:	ara/rest/client.h
Class:	ara::rest::Event
Description:	Cancels an event subscription by issuing a cancelation request. A subscription can also be terminated (but not canceled) by destroying the correspond Event object.

Table 8.18: ara::rest::Event::Unsubscribe

[SWS_REST_02020] ara::rest::Event::Unsubscribe ⌈Table 8.18 describes the interface `ara::rest::Event::Unsubscribe`.⌋*(RS_CM_00300)*

8.3.4 Resubscribe

Service name:	ara::rest::Event::Resubscribe
Type:	`Member function`
Syntax:	`Task<bool> ara::rest::Event::Resubscribe()`
Function param:	None

Return value:	a task waiting for re-subscription to be finished which returns true on success
Exceptions:	Implementation-defined
Header file:	ara/rest/client.h
Class:	ara::rest::Event
Description:	Re-subscribes to an event. Resubscription to an already subscribed event is valid but has not user-visible effect.

Table 8.19: ara::rest::Event::Resubscribe

[SWS_REST_02021] ara::rest::Event::Resubscribe ⌈Table 8.19 describes the interface `ara::rest::Event::Resubscribe`.⌋*(RS_CM_00300)*

8.3.5 GetUri

Service name:	ara::rest::Event::GetUri
Type:	`Member function`
Syntax:	`const Uri& ara::rest::Event::GetUri() const`
Function param:	None
Return value:	the Uri corresponding to this event subscription
Exceptions:	`noexcept`
Header file:	ara/rest/client.h
Class:	ara::rest::Event
Description:	Returns the event Uri.

Table 8.20: ara::rest::Event::GetUri

[SWS_REST_02022] ara::rest::Event::GetUri ⌈Table 8.20 describes the interface `ara::rest::Event::GetUri`.⌋*(RS_CM_00300, RS_CM_00304)*

8.3.6 GetSubscriptionState

Service name:	ara::rest::Event::GetSubscriptionState
Type:	`Member function`
Syntax:	`SubscriptionState ara::rest::Event::GetSubscriptionState() const`
Function param:	None
Return value:	the current subscription state as perceived by the client
Exceptions:	`noexcept`
Header file:	ara/rest/client.h
Class:	ara::rest::Event
Description:	Returns the current subscription state.

Table 8.21: ara::rest::Event::GetSubscriptionState

[SWS_REST_02023] ara::rest::Event::GetSubscriptionState ⌈Table 8.21 describes the interface `ara::rest::Event::GetSubscriptionState`.⌋*(RS_CM_00300)*

8.3.7 operator==

Service name:	ara::rest::Event::operator==	
Type:	`Non-member function`	
Syntax:	`friend bool operator==(const Event &a, const Event &b)`	
Function param:	a	an event
Function param:	b	an event
Return value:	true if a and b are equal	
Exceptions:	`noexcept`	
Header file:	ara/rest/client.h	
Namespace:	ara::rest::Event	
Description:	Tests events for equality.	

Table 8.22: ara::rest::Event::operator==

[SWS_REST_02024] ara::rest::Event::operator== ⌈Table 8.22 describes the interface `ara::rest::Event::operator==`.⌋ *(RS_CM_00300)*

8.3.8 operator!=

Service name:	ara::rest::Event::operator!=	
Type:	`Non-member function`	
Syntax:	`friend bool operator!=(const Event &a, const Event &b)`	
Function param:	a	an event
Function param:	b	an event
Return value:	true if a and b are unequal	
Exceptions:	`noexcept`	
Header file:	ara/rest/client.h	
Namespace:	ara::rest::Event	
Description:	Tests events for inequality.	

Table 8.23: ara::rest::Event::operator!=

[SWS_REST_02025] ara::rest::Event::operator!= ⌈Table 8.23 describes the interface `ara::rest::Event::operator!=`.⌋ *(RS_CM_00300)*

8.3.9 operator<

Service name:	ara::rest::Event::operator<	
Type:	`Non-member function`	
Syntax:	`friend bool operator<(const Event &a, const Event &b)`	
Function param:	a	an event
Function param:	b	an event
Return value:	true if a less-than b	
Exceptions:	`noexcept`	
Header file:	ara/rest/client.h	

Namespace:	ara::rest::Event
Description:	Tests events for their partial order Order criterion is implementation-defined.

Table 8.24: ara::rest::Event::operator<

[SWS_REST_02026] ara::rest::Event::operator< ⌈Table 8.24 describes the interface `ara::rest::Event::operator<`.⌋*(RS_CM_00300)*

8.4 ara::rest::IteratorRange

[SWS_REST_02382] ⌈`ara::rest::IteratorRange` class shall be declared in the `ara/rest/iterator.h` header file:

```
template <typename IterT >
class ara::rest::IteratorRange;
```

⌋*(RS_CM_00300)*

8.4.1 Iterator

Name:	`Iterator`
Type:	`Member type alias`
Syntax:	`using ara::rest::IteratorRange< IterT >::Iterator = IterT`
Header file:	ara/rest/iterator.h
Class:	ara::rest::IteratorRange
Description:	Type of the underlying pair of iterators.

Table 8.25: ara::rest::IteratorRange::Iterator

[SWS_REST_02383] Iterator ⌈Table 8.25 describes the type alias `ara::rest::IteratorRange::Iterator`.⌋*(RS_CM_00300)*

8.4.2 IteratorRange

Service name:	ara::rest::IteratorRange::IteratorRange	
Type:	`Member function`	
Syntax:	`ara::rest::IteratorRange< IterT >::IteratorRange(Iterator first, Iterator last)`	
Function param:	first	an iterator denoting the start of the sequence
Function param:	last	an iterator denoting the end of the sequence
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/iterator.h	
Class:	ara::rest::IteratorRange	

Description:	Constructs an IteratorRange from a pair of iterators. For convenient construction, see MakeIteratorRange().

Table 8.26: ara::rest::IteratorRange::IteratorRange

[SWS_REST_02384] ara::rest::IteratorRange::IteratorRange ⌈Table 8.26 describes the interface `ara::rest::IteratorRange::IteratorRange`.⌋*(RS_CM_00300)*

8.4.3 begin

Service name:	ara::rest::IteratorRange::begin
Type:	`Member function`
Syntax:	`Iterator ara::rest::IteratorRange< IterT >::begin() const`
Function param:	None
Return value:	an iterator
Exceptions:	Implementation-defined
Header file:	ara/rest/iterator.h
Class:	ara::rest::IteratorRange
Description:	Returns the start of the sequence.

Table 8.27: ara::rest::IteratorRange::begin

[SWS_REST_02385] ara::rest::IteratorRange::begin ⌈Table 8.27 describes the interface `ara::rest::IteratorRange::begin`.⌋*(RS_CM_00300)*

8.4.4 end

Service name:	ara::rest::IteratorRange::end
Type:	`Member function`
Syntax:	`Iterator ara::rest::IteratorRange< IterT >::end() const`
Function param:	None
Return value:	an iterator
Exceptions:	Implementation-defined
Header file:	ara/rest/iterator.h
Class:	ara::rest::IteratorRange
Description:	Returns the end of the sequence.

Table 8.28: ara::rest::IteratorRange::end

[SWS_REST_02386] ara::rest::IteratorRange::end ⌈Table 8.28 describes the interface `ara::rest::IteratorRange::end`.⌋*(RS_CM_00300)*

8.4.5 begin

Service name:	ara::rest::IteratorRange::begin	
Type:	`Non-member function`	
Syntax:	`friend Iterator begin(const IteratorRange &r)`	
Function param:	r	an InteratorRange
Return value:	the start of the sequence	
Exceptions:	Implementation-defined	
Header file:	ara/rest/iterator.h	
Namespace:	ara::rest::IteratorRange	
Description:	Non-member equivalent of IteratorRange::begin()	

Table 8.29: ara::rest::IteratorRange::begin

[SWS_REST_02387] ara::rest::IteratorRange::begin ⌈Table 8.29 describes the interface `ara::rest::IteratorRange::begin`.⌋*(RS_CM_00300)*

8.4.6 end

Service name:	ara::rest::IteratorRange::end	
Type:	`Non-member function`	
Syntax:	`friend Iterator end(const IteratorRange &r)`	
Function param:	r	an InteratorRange
Return value:	the end of the sequence	
Exceptions:	Implementation-defined	
Header file:	ara/rest/iterator.h	
Namespace:	ara::rest::IteratorRange	
Description:	Non-member equivalent of IteratorRange::end()	

Table 8.30: ara::rest::IteratorRange::end

[SWS_REST_02388] ara::rest::IteratorRange::end ⌈Table 8.30 describes the interface `ara::rest::IteratorRange::end`.⌋*(RS_CM_00300)*

8.5 ara::rest::Matches

[SWS_REST_02027] ⌈`ara::rest::Matches` class shall be declared in the `ara/rest/routing.h` header file:

```
1 class ara::rest::Matches;
```

⌋*(RS_CM_00300, RS_CM_00309)*

8.5.1 MatchRange

Name:	`MatchRange`
Type:	`Member type alias`
Syntax:	`using ara::rest::Matches::MatchRange = IteratorRange<unspecified_iterator_type>`
Header file:	ara/rest/routing.h
Class:	ara::rest::Matches
Description:	An IteratorRange of all pattern matches for this Route.

Table 8.31: ara::rest::Matches::MatchRange

[SWS_REST_02028] MatchRange ⌈Table 8.31 describes the type alias `ara::rest::Matches::MatchRange`.⌋ *(RS_CM_00300, RS_CM_00309)*

8.5.2 Count

Service name:	ara::rest::Matches::Count
Type:	`Member function`
Syntax:	`std::size_t ara::rest::Matches::Count() const`
Function param:	None
Return value:	the number of URI matches
Exceptions:	`noexcept`
Header file:	ara/rest/routing.h
Class:	ara::rest::Matches
Description:	Provides the number of URI wildcard matches after applying a pattern to a route.

Table 8.32: ara::rest::Matches::Count

[SWS_REST_02029] ara::rest::Matches::Count ⌈Table 8.32 describes the interface `ara::rest::Matches::Count`.⌋ *(RS_CM_00300, RS_CM_00309)*

8.5.3 Get

Service name:	ara::rest::Matches::Get	
Type:	`Member function`	
Syntax:	`const Match& ara::rest::Matches::Get(std::size_t i) const`	
Function param:	i	index to the i'th URI wildcard match
Return value:	return type	
Exceptions:	`noexcept`	
Header file:	ara/rest/routing.h	
Class:	ara::rest::Matches	
Description:	Provides access to a specific URI match.	

Table 8.33: ara::rest::Matches::Get

[SWS_REST_02030] ara::rest::Matches::Get ⌈Table 8.33 describes the interface `ara::rest::Matches::Get`.⌋*(RS_CM_00300, RS_CM_00309)*

8.5.4 Get

Service name:	ara::rest::Matches::Get
Type:	`Member function`
Syntax:	`MatchRange ara::rest::Matches::Get() const`
Function param:	None
Return value:	a range of URI matches
Exceptions:	`noexcept`
Header file:	ara/rest/routing.h
Class:	ara::rest::Matches
Description:	Provides access to the sequence of URI (wildcard) matches. After this route has been matched against a given request, all wildcard URI matches are accessible with this range.

Table 8.34: ara::rest::Matches::Get

[SWS_REST_02031] ara::rest::Matches::Get ⌈Table 8.34 describes the interface `ara::rest::Matches::Get`.⌋*(RS_CM_00300, RS_CM_00309)*

8.5.5 GetRoute

Service name:	ara::rest::Matches::GetRoute
Type:	`Member function`
Syntax:	`const Route& ara::rest::Matches::GetRoute() const`
Function param:	None
Return value:	a reference to the currently matched route
Exceptions:	`noexcept`
Header file:	ara/rest/routing.h
Class:	ara::rest::Matches
Description:	Returns the route currently matched.

Table 8.35: ara::rest::Matches::GetRoute

[SWS_REST_02032] ara::rest::Matches::GetRoute ⌈Table 8.35 describes the interface `ara::rest::Matches::GetRoute`.⌋*(RS_CM_00300, RS_CM_00309)*

8.6 ara::rest::Match

[SWS_REST_02033] ⌈`ara::rest::Match` class shall be declared in the `ara/rest/routing.h` header file:

```
1 class ara::rest::Match;
```

⌋(*RS_CM_00300*, *RS_CM_00309*)

8.6.1 Get

Service name:	ara::rest::Match::Get
Type:	`Member function`
Syntax:	`StringView ara::rest::Match::Get() const`
Function param:	None
Return value:	a string of the matches path segment
Exceptions:	`noexcept(std::is_nothrow_constructible< StringView >::value)`
Header file:	ara/rest/routing.h
Class:	ara::rest::Match
Description:	Returns a path segment as a string.

Table 8.36: ara::rest::Match::Get

[SWS_REST_02034] ara::rest::Match::Get ⌈Table 8.36 describes the interface `ara::rest::Match::Get`.⌋(*RS_CM_00300*, *RS_CM_00309*)

8.6.2 GetAs

Service name:	ara::rest::Match::GetAs	
Type:	`Member function template`	
Syntax:	`template <typename T > T ara::rest::Match::GetAs(T &&def={})`	
Function param:	def	if conversion fails,
Return value:	The converted value of conversion succeeded, otherwise it returns the function argument.	
Exceptions:	Implementation-defined	
Header file:	ara/rest/routing.h	
Class:	ara::rest::Match	
Description:	Returns a type-converted path segment. Applies a type conversion on the matched path segment. The conversion result is assigned to the function argument which is subsequently returned. If conversion fails the function argument is returned unchanged. So either form is valid: GetAs<string>(), GetAs(string{my_allocator}), GetAs<string>("conversion failed")	

Table 8.37: ara::rest::Match::GetAs

[SWS_REST_02035] ara::rest::Match::GetAs ⌈Table 8.37 describes the interface `ara::rest::Match::GetAs`.⌋(*RS_CM_00300*, *RS_CM_00309*)

8.7 ara::rest::ogm::Array

[SWS_REST_02036] ⌈`ara::rest::ogm::Array` class shall be declared in the `ara/rest/ogm/array.h` header file:

```
1 class ara::rest::ogm::Array : public ara::rest::ogm::Value;
```

⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*)

8.7.1 SelfType

Name:	`SelfType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Array::SelfType = Array`
Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	Its own type.

Table 8.38: ara::rest::ogm::Array::SelfType

[SWS_REST_02037] **SelfType** ⌈Table 8.38 describes the type alias `ara::rest::ogm::Array::SelfType`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*)

8.7.2 ParentType

Name:	`ParentType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Array::ParentType = Value`
Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	Type of its parent in the OGM type hierarchy.

Table 8.39: ara::rest::ogm::Array::ParentType

[SWS_REST_02038] **ParentType** ⌈Table 8.39 describes the type alias `ara::rest::ogm::Array::ParentType`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*)

8.7.3 Iterator

Name:	`Iterator`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Array::Iterator = unspecified_iterator_type`

Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	A forward iterator of the represented set values.

Table 8.40: ara::rest::ogm::Array::Iterator

[SWS_REST_02039] Iterator ⌈Table 8.40 describes the type alias `ara::rest::ogm::Array::Iterator`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.4 ConstIterator

Name:	`ConstIterator`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Array::ConstIterator = unspecified_iterator_type`
Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	Value iterator.

Table 8.41: ara::rest::ogm::Array::ConstIterator

[SWS_REST_02040] ConstIterator ⌈Table 8.41 describes the type alias `ara::rest::ogm::Array::ConstIterator`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.5 ValueRange

Name:	`ValueRange`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Array::ValueRange = IteratorRange<Iterator>`
Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	Iterator range.

Table 8.42: ara::rest::ogm::Array::ValueRange

[SWS_REST_02041] ValueRange ⌈Table 8.42 describes the type alias `ara::rest::ogm::Array::ValueRange`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.6 ConstValueRange

Name:	`ConstValueRange`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Array::ConstValueRange = IteratorRange<ConstIterator>`
Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	Iterator range.

Table 8.43: ara::rest::ogm::Array::ConstValueRange

[SWS_REST_02042] ConstValueRange ⌈Table 8.43 describes the type alias `ara::rest::ogm::Array::ConstValueRange`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.7 GetParent

Service name:	ara::rest::ogm::Array::GetParent
Type:	`Member function`
Syntax:	`Node* ara::rest::ogm::Array::GetParent()`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.44: ara::rest::ogm::Array::GetParent

[SWS_REST_02043] ara::rest::ogm::Array::GetParent ⌈Table 8.44 describes the interface `ara::rest::ogm::Array::GetParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.8 GetParent

Service name:	ara::rest::ogm::Array::GetParent
Type:	`Member function`
Syntax:	`const Node* ara::rest::ogm::Array::GetParent() const`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.45: ara::rest::ogm::Array::GetParent

[SWS_REST_02044] **ara::rest::ogm::Array::GetParent** ⌈Table 8.45 describes the interface `ara::rest::ogm::Array::GetParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.9 HasParent

Service name:	ara::rest::ogm::Array::HasParent
Type:	`Member function`
Syntax:	`bool ara::rest::ogm::Array::HasParent() const`
Function param:	None
Return value:	true if this node has a structural parent
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	Denotes whether this node has a structural parent.

Table 8.46: ara::rest::ogm::Array::HasParent

[SWS_REST_02045] **ara::rest::ogm::Array::HasParent** ⌈Table 8.46 describes the interface `ara::rest::ogm::Array::HasParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.10 GetSize

Service name:	ara::rest::ogm::Array::GetSize
Type:	`Member function`
Syntax:	`std::size_t ara::rest::ogm::Array::GetSize() const`
Function param:	None
Return value:	the number of array elements
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	Returns the number of elements.

Table 8.47: ara::rest::ogm::Array::GetSize

[SWS_REST_02046] **ara::rest::ogm::Array::GetSize** ⌈Table 8.47 describes the interface `ara::rest::ogm::Array::GetSize`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.11 IsEmpty

Service name:	ara::rest::ogm::Array::IsEmpty
Type:	`Member function`
Syntax:	`bool ara::rest::ogm::Array::IsEmpty() const`

Function param:	None
Return value:	true if the array holds no elements
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	Returns whether the array holds no elements.

Table 8.48: ara::rest::ogm::Array::IsEmpty

[SWS_REST_02047] ara::rest::ogm::Array::IsEmpty ⌈Table 8.48 describes the interface `ara::rest::ogm::Array::IsEmpty`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.12 GetValue

Service name:	ara::rest::ogm::Array::GetValue	
Type:	`Member function`	
Syntax:	`Value& ara::rest::ogm::Array::GetValue(std::size_t index)`	
Function param:	index	an integral index into the array
Return value:	a reference to a Value	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/array.h	
Class:	ara::rest::ogm::Array	
Description:	Returns a Value at a specific index. If the index is out-of-bounds, the result in undefined.	

Table 8.49: ara::rest::ogm::Array::GetValue

[SWS_REST_02048] ara::rest::ogm::Array::GetValue ⌈Table 8.49 describes the interface `ara::rest::ogm::Array::GetValue`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.13 GetValue

Service name:	ara::rest::ogm::Array::GetValue	
Type:	`Member function`	
Syntax:	`const Value& ara::rest::ogm::Array::GetValue(std::size_t index) const`	
Function param:	index	an integral index into the array
Return value:	a reference to a Value	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/array.h	
Class:	ara::rest::ogm::Array	
Description:	Returns a Value at a specific index. If the index is out-of-bounds, the result in undefined.	

Table 8.50: ara::rest::ogm::Array::GetValue

[SWS_REST_02049] ara::rest::ogm::Array::GetValue ⌈Table 8.50 describes the interface `ara::rest::ogm::Array::GetValue`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.7.14 GetValues

Service name:	ara::rest::ogm::Array::GetValues
Type:	`Member function`
Syntax:	`ValueRange ara::rest::ogm::Array::GetValues()`
Function param:	None
Return value:	an iterator range of values
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	Returns a range of values.

Table 8.51: ara::rest::ogm::Array::GetValues

[SWS_REST_02050] ara::rest::ogm::Array::GetValues ⌈Table 8.51 describes the interface `ara::rest::ogm::Array::GetValues`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.7.15 GetValues

Service name:	ara::rest::ogm::Array::GetValues
Type:	`Member function`
Syntax:	`ConstValueRange ara::rest::ogm::Array::GetValues() const`
Function param:	None
Return value:	an iterator range of values
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	Returns a range of values.

Table 8.52: ara::rest::ogm::Array::GetValues

[SWS_REST_02051] ara::rest::ogm::Array::GetValues ⌈Table 8.52 describes the interface `ara::rest::ogm::Array::GetValues`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.7.16 Append

Service name:	ara::rest::ogm::Array::Append	
Type:	`Member function`	
Syntax:	`void ara::rest::ogm::Array::Append(Pointer< Value > &&v)`	
Function param:	v	a Pointer to a value
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/array.h	
Class:	ara::rest::ogm::Array	
Description:	Appends a Value object to the array.	

Table 8.53: ara::rest::ogm::Array::Append

[SWS_REST_02052] ara::rest::ogm::Array::Append ⌈Table 8.53 describes the interface `ara::rest::ogm::Array::Append`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.17 Insert

Service name:	ara::rest::ogm::Array::Insert	
Type:	`Member function`	
Syntax:	`void ara::rest::ogm::Array::Insert(Iterator iter, Pointer< Value > &&v)`	
Function param:	iter	an Array iterator
Function param:	v	a value to insert.
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/array.h	
Class:	ara::rest::ogm::Array	
Description:	Inserts a Value at a specific position into the Array. Inserts a value before the element pointed to by the iterator argument. To insert a Value ownership has to be passeed to the Array	

Table 8.54: ara::rest::ogm::Array::Insert

[SWS_REST_02053] ara::rest::ogm::Array::Insert ⌈Table 8.54 describes the interface `ara::rest::ogm::Array::Insert`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.18 Remove

Service name:	ara::rest::ogm::Array::Remove	
Type:	`Member function`	
Syntax:	`Iterator ara::rest::ogm::Array::Remove(Iterator iter)`	
Function param:	iter	an iterator pointing to an array element

Return value:	an iterator pointing to the element following the one just removed.
Exceptions:	Implementation-defined
Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	Removes an element from the array. Removes the element pointed to be the iterator argument

Table 8.55: ara::rest::ogm::Array::Remove

[SWS_REST_02054] ara::rest::ogm::Array::Remove ⌈Table 8.55 describes the interface `ara::rest::ogm::Array::Remove`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.19 Release

Service name:	ara::rest::ogm::Array::Release	
Type:	`Member function`	
Syntax:	`std::pair<Iterator, Pointer<Value> > ara::rest::ogm::Array::Release(Iterator iter)`	
Function param:	iter	an iterator pointing to the element to be removed
Return value:	a pair of the iterator pointing to the element following the one just deleted and a pointer to the element	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/array.h	
Class:	ara::rest::ogm::Array	
Description:	Similar to Remove but does not destroy the removed element. Instead of destroying the removed element, ownership is passed back to the user	

Table 8.56: ara::rest::ogm::Array::Release

[SWS_REST_02055] ara::rest::ogm::Array::Release ⌈Table 8.56 describes the interface `ara::rest::ogm::Array::Release`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.20 Replace

Service name:	ara::rest::ogm::Array::Replace	
Type:	`Member function`	
Syntax:	`Pointer<Value> ara::rest::ogm::Array::Replace(Iterator iter, Pointer< Value > &&v)`	
Function param:	iter	an iterator pointing to the element to be removed
Function param:	v	a pointer to the value to replace the one pointed to by iter
Return value:	a pointer to the old array element	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/array.h	

Class:	ara::rest::ogm::Array
Description:	Replaces an element by a new one without the destroying the old one. Replaces an array element without destroying it. Instead the replaced element is returned. Effectively, ownership is passed back to the user.

Table 8.57: ara::rest::ogm::Array::Replace

[SWS_REST_02056] ara::rest::ogm::Array::Replace ⌈Table 8.57 describes the interface `ara::rest::ogm::Array::Replace`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.21 Clear

Service name:	ara::rest::ogm::Array::Clear
Type:	`Member function`
Syntax:	`void ara::rest::ogm::Array::Clear()`
Function param:	None
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/ogm/array.h
Class:	ara::rest::ogm::Array
Description:	Removes and destroys all elements of the array.

Table 8.58: ara::rest::ogm::Array::Clear

[SWS_REST_02057] ara::rest::ogm::Array::Clear ⌈Table 8.58 describes the interface `ara::rest::ogm::Array::Clear`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.22 Make

Service name:	ara::rest::ogm::Array::Make	
Type:	`Member function`	

Syntax:	`template <typename... Ts>`
	`static Pointer<SelfType>`
	`ara::rest::ogm::Array::Make(Ts &&...ts)`	

Function param:	ts	constructor arguments forwarded to the constructor of this type
Return value:	a pointer to a node of type SelfType	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/array.h	
Class:	ara::rest::ogm::Array	
Description:	Creates a node of type SelfType.	

Table 8.59: ara::rest::ogm::Array::Make

[SWS_REST_02058] **ara::rest::ogm::Array::Make** ⌈Table 8.59 describes the interface `ara::rest::ogm::Array::Make`.⌋ *(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.23 Make

Service name:	ara::rest::ogm::Array::Make	
Type:	`Member function`	

Syntax:	`template <typename... Ts>`
	`static Pointer<SelfType>`
	`ara::rest::ogm::Array::Make(Allocator *alloc, Ts &&...ts)`	

Function param:	alloc	an allocator to use to construct this node
Function param:	ts	constructor arguments forwarded to the constructor of this type
Return value:	a pointer to a node of type SelfType	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/array.h	
Class:	ara::rest::ogm::Array	
Description:	Creates a node of type SelfType.	

Table 8.60: ara::rest::ogm::Array::Make

[SWS_REST_02059] **ara::rest::ogm::Array::Make** ⌈Table 8.60 describes the interface `ara::rest::ogm::Array::Make`.⌋ *(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.24 Array

Service name:	ara::rest::ogm::Array::Array	
Type:	`Member function`	
Syntax:	`template <typename... Ts>` `ara::rest::ogm::Array::Array(Pointer< Ts > &&...ts)`	
Function param:	ts	OGM objects to insert into the array
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/array.h	
Class:	ara::rest::ogm::Array	
Description:	Constructs an Array.	

Table 8.61: ara::rest::ogm::Array::Array

[SWS_REST_02060] **ara::rest::ogm::Array::Array** ⌈Table 8.61 describes the interface `ara::rest::ogm::Array::Array`.⌋ *(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.7.25 Array

Service name:	ara::rest::ogm::Array::Array	
Type:	`Member function`	
Syntax:	`template <typename... Ts>` `ara::rest::ogm::Array::Array(Allocator *alloc, Pointer< Ts > &&...ts)`	
Function param:	alloc	an allocator
Function param:	ts	OGM objects to insert into the array
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/array.h	
Class:	ara::rest::ogm::Array	
Description:	Constructs an Array.	

Table 8.62: ara::rest::ogm::Array::Array

[SWS_REST_02061] ara::rest::ogm::Array::Array ⌈Table 8.62 describes the interface `ara::rest::ogm::Array::Array`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.8 ara::rest::ogm::Field

[SWS_REST_02062] ⌈`ara::rest::ogm::Field` class shall be declared in the `ara/rest/ogm/field.h` header file:

```
1 class ara::rest::ogm::Field : public ara::rest::ogm::Node;
```

⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307)*

8.8.1 SelfType

Name:	`SelfType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Field::SelfType = Field`
Header file:	ara/rest/ogm/field.h
Class:	ara::rest::ogm::Field
Description:	Its own type.

Table 8.63: ara::rest::ogm::Field::SelfType

[SWS_REST_02063] SelfType ⌈Table 8.63 describes the type alias `ara::rest::ogm::Field::SelfType`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307)*

8.8.2 ParentType

Name:	`ParentType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Field::ParentType = Node`
Header file:	ara/rest/ogm/field.h
Class:	ara::rest::ogm::Field
Description:	Type of its parent in the OGM type hierarchy.

Table 8.64: ara::rest::ogm::Field::ParentType

[SWS_REST_02064] ParentType ⌈Table 8.64 describes the type alias `ara::rest::ogm::Field::ParentType`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307)*

8.8.3 GetParent

Service name:	ara::rest::ogm::Field::GetParent
Type:	`Member function`
Syntax:	`Node* ara::rest::ogm::Field::GetParent()`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/field.h
Class:	ara::rest::ogm::Field
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.65: ara::rest::ogm::Field::GetParent

[SWS_REST_02065] ara::rest::ogm::Field::GetParent ⌈Table 8.65 describes the interface `ara::rest::ogm::Field::GetParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.8.4 GetParent

Service name:	ara::rest::ogm::Field::GetParent
Type:	`Member function`
Syntax:	`const Node* ara::rest::ogm::Field::GetParent() const`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/field.h
Class:	ara::rest::ogm::Field
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.66: ara::rest::ogm::Field::GetParent

[SWS_REST_02066] **ara::rest::ogm::Field::GetParent** ⌈Table 8.66 describes the interface `ara::rest::ogm::Field::GetParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.8.5 HasParent

Service name:	ara::rest::ogm::Field::HasParent
Type:	`Member function`
Syntax:	`bool ara::rest::ogm::Field::HasParent() const`
Function param:	None
Return value:	true if this node has a structural parent
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/field.h
Class:	ara::rest::ogm::Field
Description:	Denotes whether this node has a structural parent.

Table 8.67: ara::rest::ogm::Field::HasParent

[SWS_REST_02067] **ara::rest::ogm::Field::HasParent** ⌈Table 8.67 describes the interface `ara::rest::ogm::Field::HasParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.8.6 GetName

Service name:	ara::rest::ogm::Field::GetName
Type:	`Member function`
Syntax:	`const StringView& ara::rest::ogm::Field::GetName() const`
Function param:	None
Return value:	a name
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/field.h
Class:	ara::rest::ogm::Field
Description:	Return the name of a Field. Fields names are immutable. To set a different name a new Field must be inserted.

Table 8.68: ara::rest::ogm::Field::GetName

[SWS_REST_02068] **ara::rest::ogm::Field::GetName** ⌈Table 8.68 describes the interface `ara::rest::ogm::Field::GetName`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.8.7 GetValue

Service name:	ara::rest::ogm::Field::GetValue

Type:	`Member function`
Syntax:	`const Value& ara::rest::ogm::Field::GetValue() const`
Function param:	None
Return value:	a reference to the current field value
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/field.h
Class:	ara::rest::ogm::Field
Description:	Returns the value represented by a Field.

Table 8.69: ara::rest::ogm::Field::GetValue

[SWS_REST_02069] ara::rest::ogm::Field::GetValue ⌈Table 8.69 describes the interface `ara::rest::ogm::Field::GetValue`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.8.8 GetValue

Service name:	ara::rest::ogm::Field::GetValue
Type:	`Member function`
Syntax:	`Value& ara::rest::ogm::Field::GetValue()`
Function param:	None
Return value:	a reference to the current field value
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/field.h
Class:	ara::rest::ogm::Field
Description:	Returns the value represented by a Field.

Table 8.70: ara::rest::ogm::Field::GetValue

[SWS_REST_02070] ara::rest::ogm::Field::GetValue ⌈Table 8.70 describes the interface `ara::rest::ogm::Field::GetValue`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.8.9 SetValue

Service name:	ara::rest::ogm::Field::SetValue	
Type:	`Member function`	
Syntax:	`void ara::rest::ogm::Field::SetValue(Pointer< Value > &&v)`	
Function param:	v	a new Value
Return value:	None	
Exceptions:	`noexcept`	
Header file:	ara/rest/ogm/field.h	
Class:	ara::rest::ogm::Field	
Description:	Sets a new value. The previous value is destroyed	

Table 8.71: ara::rest::ogm::Field::SetValue

[SWS_REST_02071] ara::rest::ogm::Field::SetValue ⌈Table 8.71 describes the interface `ara::rest::ogm::Field::SetValue`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.8.10 ReplaceValue

Service name:	ara::rest::ogm::Field::ReplaceValue	
Type:	`Member function`	
Syntax:	`Pointer<Value> ara::rest::ogm::Field::ReplaceValue(Pointer< Value > &&v)`	
Function param:	v	a new Value
Return value:	the old value	
Exceptions:	`noexcept`	
Header file:	ara/rest/ogm/field.h	
Class:	ara::rest::ogm::Field	
Description:	Sets a new value and returns the old one.	

Table 8.72: ara::rest::ogm::Field::ReplaceValue

[SWS_REST_02072] ara::rest::ogm::Field::ReplaceValue ⌈Table 8.72 describes the interface `ara::rest::ogm::Field::ReplaceValue`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.8.11 Make

Service name:	ara::rest::ogm::Field::Make	
Type:	`Member function`	
Syntax:	`template <typename... Ts> static Pointer<SelfType> ara::rest::ogm::Field::Make(Ts &&...ts)`	
Function param:	ts	constructor arguments forwarded to the constructor of this type
Return value:	a pointer to a node of type SelfType	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/field.h	
Class:	ara::rest::ogm::Field	
Description:	Creates a node of type SelfType.	

Table 8.73: ara::rest::ogm::Field::Make

[SWS_REST_02073] ara::rest::ogm::Field::Make ⌈Table 8.73 describes the interface `ara::rest::ogm::Field::Make`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.8.12 Make

Service name:	ara::rest::ogm::Field::Make	
Type:	`Member function`	

Syntax:	`template <typename... Ts>`
	`static Pointer<SelfType>`
	`ara::rest::ogm::Field::Make(Allocator *alloc, Ts &&...ts)`	

Function param:	alloc	an allocator to use to construct this node
Function param:	ts	constructor arguments forwarded to the constructor of this type
Return value:	a pointer to a node of type SelfType	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/field.h	
Class:	ara::rest::ogm::Field	
Description:	Creates a node of type SelfType.	

Table 8.74: ara::rest::ogm::Field::Make

[SWS_REST_02074] ara::rest::ogm::Field::Make ⌈Table 8.74 describes the interface `ara::rest::ogm::Field::Make`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.8.13 Field

Service name:	ara::rest::ogm::Field::Field	
Type:	`Member function`	
Syntax:	`ara::rest::ogm::Field::Field(const String &name, Pointer< Value > &&value)`	
Function param:	name	name of this Field
Function param:	value	value object attached to this Field
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/field.h	
Class:	ara::rest::ogm::Field	
Description:	Constructs a Field.	

Table 8.75: ara::rest::ogm::Field::Field

[SWS_REST_02075] ara::rest::ogm::Field::Field ⌈Table 8.75 describes the interface `ara::rest::ogm::Field::Field`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.8.14 Field

Service name:	ara::rest::ogm::Field::Field
Type:	`Member function`

Syntax:	`ara::rest::ogm::Field::Field(Allocator *alloc, const String &key, Pointer< Value > &&val)`	
Function param:	alloc	an allocator
Function param:	key	a field name
Function param:	val	a field value
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/field.h	
Class:	ara::rest::ogm::Field	
Description:	Constructs field, providing an allocator. The allocator argument may be used for internal allocation purposes.	

Table 8.76: ara::rest::ogm::Field::Field

[SWS_REST_02076] ara::rest::ogm::Field::Field ⌈Table 8.76 describes the interface `ara::rest::ogm::Field::Field`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.9 ara::rest::ogm::Int

[SWS_REST_02077] ⌈`ara::rest::ogm::Int` class shall be declared in the `ara/rest/ogm/int.h` header file:

```
1 class ara::rest::ogm::Int : public ara::rest::ogm::Value;
```

⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307)*

8.9.1 SelfType

Name:	`SelfType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Int::SelfType = Int`
Header file:	ara/rest/ogm/int.h
Class:	ara::rest::ogm::Int
Description:	Its own type.

Table 8.77: ara::rest::ogm::Int::SelfType

[SWS_REST_02078] SelfType ⌈Table 8.77 describes the type alias `ara::rest::ogm::Int::SelfType`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307)*

8.9.2 ParentType

Name:	`ParentType`

Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Int::ParentType = Value`
Header file:	ara/rest/ogm/int.h
Class:	ara::rest::ogm::Int
Description:	Type of its parent in the OGM type hierarchy.

Table 8.78: ara::rest::ogm::Int::ParentType

[SWS_REST_02079] **ParentType** ⌈Table 8.78 describes the type alias `ara::rest::ogm::Int::ParentType`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*)

8.9.3 ValueType

Name:	`ValueType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Int::ValueType = std::int64_t`
Header file:	ara/rest/ogm/int.h
Class:	ara::rest::ogm::Int
Description:	Type of its corresponding C++ data type.

Table 8.79: ara::rest::ogm::Int::ValueType

[SWS_REST_02080] **ValueType** ⌈Table 8.79 describes the type alias `ara::rest::ogm::Int::ValueType`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*)

8.9.4 GetParent

Service name:	ara::rest::ogm::Int::GetParent
Type:	`Member function`
Syntax:	`Node* ara::rest::ogm::Int::GetParent()`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/int.h
Class:	ara::rest::ogm::Int
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.80: ara::rest::ogm::Int::GetParent

[SWS_REST_02081] **ara::rest::ogm::Int::GetParent** ⌈Table 8.80 describes the interface `ara::rest::ogm::Int::GetParent`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.9.5 GetParent

Service name:	ara::rest::ogm::Int::GetParent
Type:	`Member function`
Syntax:	`const Node* ara::rest::ogm::Int::GetParent() const`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/int.h
Class:	ara::rest::ogm::Int
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.81: ara::rest::ogm::Int::GetParent

[SWS_REST_02082] ara::rest::ogm::Int::GetParent ⌈Table 8.81 describes the interface `ara::rest::ogm::Int::GetParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.9.6 HasParent

Service name:	ara::rest::ogm::Int::HasParent
Type:	`Member function`
Syntax:	`bool ara::rest::ogm::Int::HasParent() const`
Function param:	None
Return value:	true if this node has a structural parent
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/int.h
Class:	ara::rest::ogm::Int
Description:	Denotes whether this node has a structural parent.

Table 8.82: ara::rest::ogm::Int::HasParent

[SWS_REST_02083] ara::rest::ogm::Int::HasParent ⌈Table 8.82 describes the interface `ara::rest::ogm::Int::HasParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.9.7 GetValue

Service name:	ara::rest::ogm::Int::GetValue
Type:	`Member function`
Syntax:	`ValueType ara::rest::ogm::Int::GetValue() const`
Function param:	None
Return value:	a value of type ValueType
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/int.h
Class:	ara::rest::ogm::Int
Description:	Returns its value as a C++ data type.

Table 8.83: ara::rest::ogm::Int::GetValue

[SWS_REST_02084] ara::rest::ogm::Int::GetValue ⌈Table 8.83 describes the interface `ara::rest::ogm::Int::GetValue`.⌋ *(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.9.8 SetValue

Service name:	ara::rest::ogm::Int::SetValue	
Type:	`Member function`	
Syntax:	`void ara::rest::ogm::Int::SetValue(ValueType v)`	
Function param:	v	a value
Return value:	None	
Exceptions:	`noexcept`	
Header file:	ara/rest/ogm/int.h	
Class:	ara::rest::ogm::Int	
Description:	Sets the current value from a C++ data type.	

Table 8.84: ara::rest::ogm::Int::SetValue

[SWS_REST_02085] ara::rest::ogm::Int::SetValue ⌈Table 8.84 describes the interface `ara::rest::ogm::Int::SetValue`.⌋ *(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.9.9 Make

Service name:	ara::rest::ogm::Int::Make	
Type:	`Member function`	
Syntax:	`template <typename... Ts>` `static Pointer<SelfType> ara::rest::ogm::Int::Make(Ts &&...ts)`	
Function param:	ts	constructor arguments forwarded to the constructor of this type
Return value:	a pointer to a node of type SelfType	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/int.h	
Class:	ara::rest::ogm::Int	
Description:	Creates a node of type SelfType.	

Table 8.85: ara::rest::ogm::Int::Make

[SWS_REST_02086] ara::rest::ogm::Int::Make ⌈Table 8.85 describes the interface `ara::rest::ogm::Int::Make`.⌋ *(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.9.10 Make

Service name:	ara::rest::ogm::Int::Make	
Type:	`Member function`	

Syntax:	`template <typename... Ts>`
	`static Pointer<SelfType>`
	`ara::rest::ogm::Int::Make(Allocator *alloc, Ts &&...ts)`	

Function param:	alloc	an allocator to use to construct this node
Function param:	ts	constructor arguments forwarded to the constructor of this type
Return value:	a pointer to a node of type SelfType	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/int.h	
Class:	ara::rest::ogm::Int	
Description:	Creates a node of type SelfType.	

Table 8.86: ara::rest::ogm::Int::Make

[SWS_REST_02087] ara::rest::ogm::Int::Make ⌈Table 8.86 describes the interface `ara::rest::ogm::Int::Make`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.9.11 Int

Service name:	ara::rest::ogm::Int::Int	
Type:	`Member function`	
Syntax:	`ara::rest::ogm::Int::Int(ValueType value=ValueType{})`	
Function param:	value	an intial value
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/int.h	
Class:	ara::rest::ogm::Int	
Description:	Connstructs an Int.	

Table 8.87: ara::rest::ogm::Int::Int

[SWS_REST_02088] ara::rest::ogm::Int::Int ⌈Table 8.87 describes the interface `ara::rest::ogm::Int::Int`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.10 ara::rest::ogm::Node

[SWS_REST_02089] ⌈`ara::rest::ogm::Node` class shall be declared in the `ara/rest/ogm/node.h` header file:

```
1 class ara::rest::ogm::Node;
```

⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*)

8.10.1 SelfType

Name:	`SelfType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Node::SelfType = Node`
Header file:	ara/rest/ogm/node.h
Class:	ara::rest::ogm::Node
Description:	Type of this OGM node.

Table 8.88: ara::rest::ogm::Node::SelfType

[SWS_REST_02090] **SelfType** ⌈Table 8.88 describes the type alias `ara::rest::ogm::Node::SelfType`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*)

8.10.2 ParentType

Name:	`ParentType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Node::ParentType = void`
Header file:	ara/rest/ogm/node.h
Class:	ara::rest::ogm::Node
Description:	Type of its parent in the OGM type hierarchy.

Table 8.89: ara::rest::ogm::Node::ParentType

[SWS_REST_02091] **ParentType** ⌈Table 8.89 describes the type alias `ara::rest::ogm::Node::ParentType`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*)

8.10.3 GetParent

Service name:	ara::rest::ogm::Node::GetParent
Type:	`Member function`
Syntax:	`ParentType* ara::rest::ogm::Node::GetParent()`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/node.h
Class:	ara::rest::ogm::Node
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.90: ara::rest::ogm::Node::GetParent

[SWS_REST_02092] ara::rest::ogm::Node::GetParent ⌈Table 8.90 describes the interface `ara::rest::ogm::Node::GetParent`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.10.4 GetParent

Service name:	ara::rest::ogm::Node::GetParent
Type:	`Member function`
Syntax:	`const ParentType* ara::rest::ogm::Node::GetParent() const`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/node.h
Class:	ara::rest::ogm::Node
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.91: ara::rest::ogm::Node::GetParent

[SWS_REST_02093] ara::rest::ogm::Node::GetParent ⌈Table 8.91 describes the interface `ara::rest::ogm::Node::GetParent`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.10.5 HasParent

Service name:	ara::rest::ogm::Node::HasParent
Type:	`Member function`
Syntax:	`bool ara::rest::ogm::Node::HasParent() const`
Function param:	None
Return value:	true of this node has a parent
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/node.h
Class:	ara::rest::ogm::Node
Description:	Denotes whether this node has a structural parent.

Table 8.92: ara::rest::ogm::Node::HasParent

[SWS_REST_02094] ara::rest::ogm::Node::HasParent ⌈Table 8.92 describes the interface `ara::rest::ogm::Node::HasParent`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.10.6 ~Node

Service name:	ara::rest::ogm::Node::~Node
Type:	`Member function`

Syntax:	`virtual ara::rest::ogm::Node::~Node()`
Function param:	None
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/ogm/node.h
Class:	ara::rest::ogm::Node
Description:	Destructor.

Table 8.93: ara::rest::ogm::Node::~Node

[SWS_REST_02095] ara::rest::ogm::Node::~Node ⌈Table 8.93 describes the interface `ara::rest::ogm::Node::~Node`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.10.7 Node

Service name:	ara::rest::ogm::Node::Node
Type:	`Member function`
Syntax:	`ara::rest::ogm::Node::Node(const Node &)=delete`
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/ogm/node.h
Class:	ara::rest::ogm::Node
Description:	Non-copyable; copy with ogm::Copy()

Table 8.94: ara::rest::ogm::Node::Node

[SWS_REST_02096] ara::rest::ogm::Node::Node ⌈Table 8.94 describes the interface `ara::rest::ogm::Node::Node`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.10.8 operator=

Service name:	ara::rest::ogm::Node::operator=
Type:	`Member function`
Syntax:	`Node& ara::rest::ogm::Node::operator=(const Node &)=delete`
Return value:	a value of type `Node &`
Exceptions:	Implementation-defined
Header file:	ara/rest/ogm/node.h
Class:	ara::rest::ogm::Node
Description:	Non-copy-assignable; copy with ogm::Copy()

Table 8.95: ara::rest::ogm::Node::operator=

[SWS_REST_02097] ara::rest::ogm::Node::operator= ⌈Table 8.95 describes the interface `ara::rest::ogm::Node::operator=`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.10.9 GetAllocator

Service name:	ara::rest::ogm::Node::GetAllocator
Type:	`Member function`
Syntax:	`Allocator* ara::rest::ogm::Node::GetAllocator()`
Function param:	None
Return value:	a pointer to an allocator
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/node.h
Class:	ara::rest::ogm::Node
Description:	Returns a pointer to the allocator that manages this subtree.

Table 8.96: ara::rest::ogm::Node::GetAllocator

[SWS_REST_02098] ara::rest::ogm::Node::GetAllocator ⌈Table 8.96 describes the interface `ara::rest::ogm::Node::GetAllocator`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.10.10 GetAllocator

Service name:	ara::rest::ogm::Node::GetAllocator
Type:	`Member function`
Syntax:	`const Allocator* ara::rest::ogm::Node::GetAllocator() const`
Function param:	None
Return value:	a pointer to an allocator
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/node.h
Class:	ara::rest::ogm::Node
Description:	Returns a pointer to the allocator that manages this subtree.

Table 8.97: ara::rest::ogm::Node::GetAllocator

[SWS_REST_02099] ara::rest::ogm::Node::GetAllocator ⌈Table 8.97 describes the interface `ara::rest::ogm::Node::GetAllocator`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.10.11 Node

Service name:	ara::rest::ogm::Node::Node
Type:	`Member function`

Syntax:	`ara::rest::ogm::Node::Node()`
Function param:	None
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/ogm/node.h
Class:	ara::rest::ogm::Node
Description:	Constructs a node. Inaccessible to the user

Table 8.98: ara::rest::ogm::Node::Node

[SWS_REST_02100] ara::rest::ogm::Node::Node ⌈Table 8.98 describes the interface `ara::rest::ogm::Node::Node`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11 ara::rest::ogm::Object

[SWS_REST_02101] ⌈`ara::rest::ogm::Object` class shall be declared in the `ara/rest/ogm/object.h` header file:

```
1 class ara::rest::ogm::Object : public ara::rest::ogm::Value;
```

⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307)*

8.11.1 SelfType

Name:	`SelfType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Object::SelfType = Object`
Header file:	ara/rest/ogm/object.h
Class:	ara::rest::ogm::Object
Description:	Its own type.

Table 8.99: ara::rest::ogm::Object::SelfType

[SWS_REST_02102] SelfType ⌈Table 8.99 describes the type alias `ara::rest::ogm::Object::SelfType`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307)*

8.11.2 ParentType

Name:	`ParentType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Object::ParentType = Value`
Header file:	ara/rest/ogm/object.h
Class:	ara::rest::ogm::Object

Description:	Type of its parent in the OGM type hierarchy.

Table 8.100: ara::rest::ogm::Object::ParentType

[SWS_REST_02103] ParentType ⌈Table 8.100 describes the type alias `ara::rest::ogm::Object::ParentType`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*)

8.11.3 Iterator

Name:	`Iterator`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Object::Iterator = unspecified_iterator_type`
Header file:	ara/rest/ogm/object.h
Class:	ara::rest::ogm::Object
Description:	Value iterator.

Table 8.101: ara::rest::ogm::Object::Iterator

[SWS_REST_02104] Iterator ⌈Table 8.101 describes the type alias `ara::rest::ogm::Object::Iterator`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.11.4 ConstIterator

Name:	`ConstIterator`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Object::ConstIterator = unspecified_iterator_type`
Header file:	ara/rest/ogm/object.h
Class:	ara::rest::ogm::Object
Description:	Value iterator.

Table 8.102: ara::rest::ogm::Object::ConstIterator

[SWS_REST_02105] ConstIterator ⌈Table 8.102 describes the type alias `ara::rest::ogm::Object::ConstIterator`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.11.5 FieldRange

Name:	`FieldRange`
Type:	`Member type alias`

Syntax:	`using ara::rest::ogm::Object::FieldRange = IteratorRange<Iterator>`
Header file:	ara/rest/ogm/object.h
Class:	ara::rest::ogm::Object
Description:	Iterator range.

Table 8.103: ara::rest::ogm::Object::FieldRange

[SWS_REST_02106] FieldRange ⌈Table 8.103 describes the type alias `ara::rest::ogm::Object::FieldRange`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.11.6 ConstFieldRange

Name:	`ConstFieldRange`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Object::ConstFieldRange = IteratorRange<ConstIterator>`
Header file:	ara/rest/ogm/object.h
Class:	ara::rest::ogm::Object
Description:	Iterator range.

Table 8.104: ara::rest::ogm::Object::ConstFieldRange

[SWS_REST_02107] ConstFieldRange ⌈Table 8.104 describes the type alias `ara::rest::ogm::Object::ConstFieldRange`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.11.7 GetParent

Service name:	ara::rest::ogm::Object::GetParent
Type:	`Member function`
Syntax:	`Node* ara::rest::ogm::Object::GetParent()`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/object.h
Class:	ara::rest::ogm::Object
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.105: ara::rest::ogm::Object::GetParent

[SWS_REST_02108] ara::rest::ogm::Object::GetParent ⌈Table 8.105 describes the interface `ara::rest::ogm::Object::GetParent`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.11.8 GetParent

Service name:	ara::rest::ogm::Object::GetParent
Type:	`Member function`
Syntax:	`const Node* ara::rest::ogm::Object::GetParent() const`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/object.h
Class:	ara::rest::ogm::Object
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.106: ara::rest::ogm::Object::GetParent

[SWS_REST_02109] ara::rest::ogm::Object::GetParent ⌈Table 8.106 describes the interface `ara::rest::ogm::Object::GetParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.9 HasParent

Service name:	ara::rest::ogm::Object::HasParent
Type:	`Member function`
Syntax:	`bool ara::rest::ogm::Object::HasParent() const`
Function param:	None
Return value:	true if this node has a structural parent
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/object.h
Class:	ara::rest::ogm::Object
Description:	Denotes whether this node has a structural parent.

Table 8.107: ara::rest::ogm::Object::HasParent

[SWS_REST_02110] ara::rest::ogm::Object::HasParent ⌈Table 8.107 describes the interface `ara::rest::ogm::Object::HasParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.10 GetSize

Service name:	ara::rest::ogm::Object::GetSize
Type:	`Member function`
Syntax:	`std::size_t ara::rest::ogm::Object::GetSize() const`
Function param:	None
Return value:	the number of array elements
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/object.h
Class:	ara::rest::ogm::Object
Description:	Returns the number of elements.

Table 8.108: ara::rest::ogm::Object::GetSize

[SWS_REST_02111] ara::rest::ogm::Object::GetSize ⌈Table 8.108 describes the interface `ara::rest::ogm::Object::GetSize`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.11 IsEmpty

Service name:	ara::rest::ogm::Object::IsEmpty
Type:	`Member function`
Syntax:	`bool ara::rest::ogm::Object::IsEmpty() const`
Function param:	None
Return value:	true if the array holds no elements
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/object.h
Class:	ara::rest::ogm::Object
Description:	Returns whether the object holds no elements.

Table 8.109: ara::rest::ogm::Object::IsEmpty

[SWS_REST_02112] ara::rest::ogm::Object::IsEmpty ⌈Table 8.109 describes the interface `ara::rest::ogm::Object::IsEmpty`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.12 GetFields

Service name:	ara::rest::ogm::Object::GetFields
Type:	`Member function`
Syntax:	`FieldRange ara::rest::ogm::Object::GetFields()`
Function param:	None
Return value:	an iterator range of fields
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/object.h
Class:	ara::rest::ogm::Object
Description:	Returns a range of fields.

Table 8.110: ara::rest::ogm::Object::GetFields

[SWS_REST_02113] ara::rest::ogm::Object::GetFields ⌈Table 8.110 describes the interface `ara::rest::ogm::Object::GetFields`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.13 GetFields

Service name:	ara::rest::ogm::Object::GetFields
Type:	`Member function`
Syntax:	`ConstFieldRange ara::rest::ogm::Object::GetFields() const`
Function param:	None
Return value:	an iterator range of fields
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/object.h
Class:	ara::rest::ogm::Object
Description:	Returns a range of fields.

Table 8.111: ara::rest::ogm::Object::GetFields

[SWS_REST_02114] ara::rest::ogm::Object::GetFields ⌈Table 8.111 describes the interface `ara::rest::ogm::Object::GetFields`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.14 HasField

Service name:	ara::rest::ogm::Object::HasField	
Type:	`Member function`	
Syntax:	`bool ara::rest::ogm::Object::HasField(StringView name) const`	
Function param:	name	of the field to search for
Return value:	true if a field of the given name exists	
Exceptions:	`noexcept`	
Header file:	ara/rest/ogm/object.h	
Class:	ara::rest::ogm::Object	
Description:	Checks whether a field of a given name exists.	

Table 8.112: ara::rest::ogm::Object::HasField

[SWS_REST_02115] ara::rest::ogm::Object::HasField ⌈Table 8.112 describes the interface `ara::rest::ogm::Object::HasField`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.15 Find

Service name:	ara::rest::ogm::Object::Find	
Type:	`Member function`	
Syntax:	`Iterator ara::rest::ogm::Object::Find(StringView name)`	
Function param:	name	field name to look up
Return value:	an iterator pointing to the position of the element.	
Exceptions:	`noexcept`	
Header file:	ara/rest/ogm/object.h	
Class:	ara::rest::ogm::Object	

Description:	Searches for a field of the given name. If the given field name is not found, the return value will be equal to GetFields().end().

Table 8.113: ara::rest::ogm::Object::Find

[SWS_REST_02116] ara::rest::ogm::Object::Find ⌈Table 8.113 describes the interface `ara::rest::ogm::Object::Find`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.16 Find

Service name:	ara::rest::ogm::Object::Find	
Type:	`Member function`	
Syntax:	`ConstIterator ara::rest::ogm::Object::Find(StringView name) const`	
Function param:	name	field name to look up
Return value:	an iterator pointing to the position of the element.	
Exceptions:	`noexcept`	
Header file:	ara/rest/ogm/object.h	
Class:	ara::rest::ogm::Object	
Description:	Searches for a field of the given name. If the given field name is not found, the return value will be equal to GetFields().end().	

Table 8.114: ara::rest::ogm::Object::Find

[SWS_REST_02117] ara::rest::ogm::Object::Find ⌈Table 8.114 describes the interface `ara::rest::ogm::Object::Find`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.17 Insert

Service name:	ara::rest::ogm::Object::Insert	
Type:	`Member function`	
Syntax:	`bool ara::rest::ogm::Object::Insert(Pointer< Field > &&f)`	
Function param:	f	field to insert
Return value:	true if insertion was performed.	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/object.h	
Class:	ara::rest::ogm::Object	
Description:	Inserts a field into the object. If a field of the same name already exists, no insertion is performed. In this case the passed pointer to Field is not invalidated.	

Table 8.115: ara::rest::ogm::Object::Insert

[SWS_REST_02118] ara::rest::ogm::Object::Insert ⌈Table 8.115 describes the interface `ara::rest::ogm::Object::Insert`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.18 Remove

Service name:	ara::rest::ogm::Object::Remove	
Type:	`Member function`	
Syntax:	`Iterator ara::rest::ogm::Object::Remove(Iterator iter)`	
Function param:	iter	an iterator pointing to an element.
Return value:	an iterator pointing to the element following the one just removed.	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/object.h	
Class:	ara::rest::ogm::Object	
Description:	Removes value from the set. Removes an element from the set. Removal invalidates all iterators referencing the respectice element.	

Table 8.116: ara::rest::ogm::Object::Remove

[SWS_REST_02119] ara::rest::ogm::Object::Remove ⌈Table 8.116 describes the interface `ara::rest::ogm::Object::Remove`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.19 Release

Service name:	ara::rest::ogm::Object::Release	
Type:	`Member function`	
Syntax:	`std::pair<Iterator, Pointer<Field> > ara::rest::ogm::Object::Release(Iterator iter)`	
Function param:	iter	an iterator pointing to the element to be removed
Return value:	a pair of the iterator pointing to the element following the one just deleted and a pointer to the element	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/object.h	
Class:	ara::rest::ogm::Object	
Description:	Similar to Remove but does not destroy the removed element. Instead of destroying the removed element, ownership is passed back to the user	

Table 8.117: ara::rest::ogm::Object::Release

[SWS_REST_02120] ara::rest::ogm::Object::Release ⌈Table 8.117 describes the interface `ara::rest::ogm::Object::Release`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.20 Replace

Service name:	ara::rest::ogm::Object::Replace	
Type:	`Member function`	
Syntax:	`Pointer<Field> ara::rest::ogm::Object::Replace(Iterator iter, Pointer< Field > &&field)`	
Function param:	iter	an iterator pointing to the element to be replaced
Function param:	field	Field to replace the current value
Return value:	a Pointer to the old element	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/object.h	
Class:	ara::rest::ogm::Object	
Description:	Replaces an element by a new one without the destroying the old one. Replaces a field without destroying it. Instead the replaced element is returned. Effectively, ownership of the old element is passed back to the user.	

Table 8.118: ara::rest::ogm::Object::Replace

[SWS_REST_02121] ara::rest::ogm::Object::Replace ⌈Table 8.118 describes the interface `ara::rest::ogm::Object::Replace`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.21 Clear

Service name:	ara::rest::ogm::Object::Clear
Type:	`Member function`
Syntax:	`void ara::rest::ogm::Object::Clear()`
Function param:	None
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/ogm/object.h
Class:	ara::rest::ogm::Object
Description:	Removes all elements.

Table 8.119: ara::rest::ogm::Object::Clear

[SWS_REST_02122] ara::rest::ogm::Object::Clear ⌈Table 8.119 describes the interface `ara::rest::ogm::Object::Clear`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.22 Make

Service name:	ara::rest::ogm::Object::Make
Type:	`Member function`

Syntax:	`template <typename... Ts>`
	`static Pointer<SelfType>`
	`ara::rest::ogm::Object::Make(Ts &&...ts)`	

Function param:	ts	constructor arguments forwarded to the constructor of this type
Return value:	a pointer to a node of type SelfType	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/object.h	
Class:	ara::rest::ogm::Object	
Description:	Creates a node of type SelfType.	

Table 8.120: ara::rest::ogm::Object::Make

[SWS_REST_02123] ara::rest::ogm::Object::Make ⌈Table 8.120 describes the interface `ara::rest::ogm::Object::Make`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.23 Make

Service name:	ara::rest::ogm::Object::Make	
Type:	`Member function`	

Syntax:	`template <typename... Ts>`
	`static Pointer<SelfType>`
	`ara::rest::ogm::Object::Make(Allocator *alloc, Ts &&...ts)`	

Function param:	alloc	an allocator to use to construct this node
Function param:	ts	constructor arguments forwarded to the constructor of this type
Return value:	a pointer to a node of type SelfType	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/object.h	
Class:	ara::rest::ogm::Object	
Description:	Creates a node of type SelfType.	

Table 8.121: ara::rest::ogm::Object::Make

[SWS_REST_02124] ara::rest::ogm::Object::Make ⌈Table 8.121 describes the interface `ara::rest::ogm::Object::Make`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.24 Object

Service name:	ara::rest::ogm::Object::Object
Type:	`Member function`
Syntax:	`template <typename... Ts>` `ara::rest::ogm::Object::Object(Pointer< Ts > &&...fields)`

Function param:	fields	Fields to be attached to this object
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/object.h	
Class:	ara::rest::ogm::Object	
Description:	Constructs an Object.	

Table 8.122: ara::rest::ogm::Object::Object

[SWS_REST_02125] ara::rest::ogm::Object::Object ⌈Table 8.122 describes the interface `ara::rest::ogm::Object::Object`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.11.25 Object

Service name:	ara::rest::ogm::Object::Object	
Type:	`Member function`	
Syntax:	`template <typename... Ts>` `ara::rest::ogm::Object::Object(Allocator *alloc, Pointer< Ts > &&...fields)`	
Function param:	alloc	an allocator
Function param:	fields	Fields to be attached to this object
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/object.h	
Class:	ara::rest::ogm::Object	
Description:	Constructs an Object.	

Table 8.123: ara::rest::ogm::Object::Object

[SWS_REST_02126] ara::rest::ogm::Object::Object ⌈Table 8.123 describes the interface `ara::rest::ogm::Object::Object`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.12 ara::rest::ogm::Real

[SWS_REST_02127] ⌈`ara::rest::ogm::Real` class shall be declared in the `ara/rest/ogm/real.h` header file:

```
1 class ara::rest::ogm::Real : public ara::rest::ogm::Value;
```

⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307)*

8.12.1 SelfType

Name:	`SelfType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Real::SelfType = Real`
Header file:	ara/rest/ogm/real.h
Class:	ara::rest::ogm::Real
Description:	Its own type.

Table 8.124: ara::rest::ogm::Real::SelfType

[SWS_REST_02128] **SelfType** ⌈Table 8.124 describes the type alias `ara::rest::ogm::Real::SelfType`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307)*

8.12.2 ParentType

Name:	`ParentType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Real::ParentType = Value`
Header file:	ara/rest/ogm/real.h
Class:	ara::rest::ogm::Real
Description:	Type of its parent in the OGM type hierarchy.

Table 8.125: ara::rest::ogm::Real::ParentType

[SWS_REST_02129] **ParentType** ⌈Table 8.125 describes the type alias `ara::rest::ogm::Real::ParentType`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307)*

8.12.3 ValueType

Name:	`ValueType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Real::ValueType = long double`
Header file:	ara/rest/ogm/real.h
Class:	ara::rest::ogm::Real
Description:	Type of its corresponding C++ data type.

Table 8.126: ara::rest::ogm::Real::ValueType

[SWS_REST_02130] **ValueType** ⌈Table 8.126 describes the type alias `ara::rest::ogm::Real::ValueType`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307)*

8.12.4 GetParent

Service name:	ara::rest::ogm::Real::GetParent
Type:	`Member function`
Syntax:	`Node* ara::rest::ogm::Real::GetParent()`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/real.h
Class:	ara::rest::ogm::Real
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.127: ara::rest::ogm::Real::GetParent

[SWS_REST_02131] ara::rest::ogm::Real::GetParent ⌈Table 8.127 describes the interface `ara::rest::ogm::Real::GetParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.12.5 GetParent

Service name:	ara::rest::ogm::Real::GetParent
Type:	`Member function`
Syntax:	`const Node* ara::rest::ogm::Real::GetParent() const`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/real.h
Class:	ara::rest::ogm::Real
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.128: ara::rest::ogm::Real::GetParent

[SWS_REST_02132] ara::rest::ogm::Real::GetParent ⌈Table 8.128 describes the interface `ara::rest::ogm::Real::GetParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.12.6 HasParent

Service name:	ara::rest::ogm::Real::HasParent
Type:	`Member function`
Syntax:	`bool ara::rest::ogm::Real::HasParent() const`
Function param:	None
Return value:	true if this node has a structural parent
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/real.h
Class:	ara::rest::ogm::Real
Description:	Denotes whether this node has a structural parent.

Table 8.129: ara::rest::ogm::Real::HasParent

[SWS_REST_02133] ara::rest::ogm::Real::HasParent ⌈Table 8.129 describes the interface `ara::rest::ogm::Real::HasParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.12.7 GetValue

Service name:	ara::rest::ogm::Real::GetValue
Type:	`Member function`
Syntax:	`ValueType ara::rest::ogm::Real::GetValue() const`
Function param:	None
Return value:	a value of type ValueType
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/real.h
Class:	ara::rest::ogm::Real
Description:	Returns its value as a C++ data type.

Table 8.130: ara::rest::ogm::Real::GetValue

[SWS_REST_02134] ara::rest::ogm::Real::GetValue ⌈Table 8.130 describes the interface `ara::rest::ogm::Real::GetValue`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.12.8 SetValue

Service name:	ara::rest::ogm::Real::SetValue	
Type:	`Member function`	
Syntax:	`void ara::rest::ogm::Real::SetValue(ValueType v)`	
Function param:	v	a value
Return value:	None	
Exceptions:	`noexcept`	
Header file:	ara/rest/ogm/real.h	
Class:	ara::rest::ogm::Real	
Description:	Sets the current value from a C++ data type.	

Table 8.131: ara::rest::ogm::Real::SetValue

[SWS_REST_02135] ara::rest::ogm::Real::SetValue ⌈Table 8.131 describes the interface `ara::rest::ogm::Real::SetValue`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.12.9 Make

Service name:	ara::rest::ogm::Real::Make
Type:	`Member function`

Syntax:	`template <typename... Ts>`
	`static Pointer<SelfType>`
	`ara::rest::ogm::Real::Make(Ts &&...ts)`	

Function param:	ts	constructor arguments forwarded to the constructor of this type
Return value:	a pointer to a node of type SelfType	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/real.h	
Class:	ara::rest::ogm::Real	
Description:	Creates a node of type SelfType.	

Table 8.132: ara::rest::ogm::Real::Make

[SWS_REST_02136] ara::rest::ogm::Real::Make ⌈Table 8.132 describes the interface `ara::rest::ogm::Real::Make`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.12.10 Make

Service name:	ara::rest::ogm::Real::Make	
Type:	`Member function`	

Syntax:	`template <typename... Ts>`
	`static Pointer<SelfType>`
	`ara::rest::ogm::Real::Make(Allocator *alloc, Ts &&...ts)`	

Function param:	alloc	an allocator to use to construct this node
Function param:	ts	constructor arguments forwarded to the constructor of this type
Return value:	a pointer to a node of type SelfType	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/real.h	
Class:	ara::rest::ogm::Real	
Description:	Creates a node of type SelfType.	

Table 8.133: ara::rest::ogm::Real::Make

[SWS_REST_02137] ara::rest::ogm::Real::Make ⌈Table 8.133 describes the interface `ara::rest::ogm::Real::Make`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.12.11 Real

Service name:	ara::rest::ogm::Real::Real	
Type:	`Member function`	
Syntax:	`ara::rest::ogm::Real::Real(ValueType value=ValueType{})`	
Function param:	value	an intial value

Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/ogm/real.h
Class:	ara::rest::ogm::Real
Description:	Connstructs an Real.

Table 8.134: ara::rest::ogm::Real::Real

[SWS_REST_02138] ara::rest::ogm::Real::Real ⌈Table 8.134 describes the interface `ara::rest::ogm::Real::Real`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.13 ara::rest::ogm::String

[SWS_REST_02139] ⌈`ara::rest::ogm::String` class shall be declared in the `ara/rest/ogm/string.h` header file:

```
1 class ara::rest::ogm::String : public ara::rest::ogm::Value;
```

⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*)

8.13.1 SelfType

Name:	`SelfType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::String::SelfType = String`
Header file:	ara/rest/ogm/string.h
Class:	ara::rest::ogm::String
Description:	Its own type.

Table 8.135: ara::rest::ogm::String::SelfType

[SWS_REST_02140] SelfType ⌈Table 8.135 describes the type alias `ara::rest::ogm::String::SelfType`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*)

8.13.2 ParentType

Name:	`ParentType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::String::ParentType = Value`
Header file:	ara/rest/ogm/string.h
Class:	ara::rest::ogm::String
Description:	Type of its parent in the OGM type hierarchy.

Table 8.136: ara::rest::ogm::String::ParentType

[SWS_REST_02141] **ParentType** ⌈Table 8.136 describes the type alias `ara::rest::ogm::String::ParentType`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*)

8.13.3 ValueType

Name:	`ValueType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::String::ValueType = ara::rest::StringView`
Header file:	ara/rest/ogm/string.h
Class:	ara::rest::ogm::String
Description:	Type of its corresponding C++ data type.

Table 8.137: ara::rest::ogm::String::ValueType

[SWS_REST_02142] **ValueType** ⌈Table 8.137 describes the type alias `ara::rest::ogm::String::ValueType`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*)

8.13.4 GetParent

Service name:	ara::rest::ogm::String::GetParent
Type:	`Member function`
Syntax:	`Node* ara::rest::ogm::String::GetParent()`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/string.h
Class:	ara::rest::ogm::String
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.138: ara::rest::ogm::String::GetParent

[SWS_REST_02143] **ara::rest::ogm::String::GetParent** ⌈Table 8.138 describes the interface `ara::rest::ogm::String::GetParent`.⌋(*RS_CM_00300*, *RS_CM_00305*, *RS_CM_00306*, *RS_CM_00307*, *RS_CM_00308*)

8.13.5 GetParent

Service name:	ara::rest::ogm::String::GetParent
Type:	`Member function`
Syntax:	`const Node* ara::rest::ogm::String::GetParent() const`
Function param:	None
Return value:	a pointer to its parent node

Exceptions:	`noexcept`
Header file:	ara/rest/ogm/string.h
Class:	ara::rest::ogm::String
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.139: ara::rest::ogm::String::GetParent

[SWS_REST_02144] ara::rest::ogm::String::GetParent ⌈Table 8.139 describes the interface `ara::rest::ogm::String::GetParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.13.6 HasParent

Service name:	ara::rest::ogm::String::HasParent
Type:	`Member function`
Syntax:	`bool ara::rest::ogm::String::HasParent() const`
Function param:	None
Return value:	true if this node has a structural parent
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/string.h
Class:	ara::rest::ogm::String
Description:	Denotes whether this node has a structural parent.

Table 8.140: ara::rest::ogm::String::HasParent

[SWS_REST_02145] ara::rest::ogm::String::HasParent ⌈Table 8.140 describes the interface `ara::rest::ogm::String::HasParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.13.7 GetValue

Service name:	ara::rest::ogm::String::GetValue
Type:	`Member function`
Syntax:	`ValueType ara::rest::ogm::String::GetValue() const`
Function param:	None
Return value:	a value of type ValueType
Exceptions:	Implementation-defined
Header file:	ara/rest/ogm/string.h
Class:	ara::rest::ogm::String
Description:	Returns its value as a C++ data type.

Table 8.141: ara::rest::ogm::String::GetValue

[SWS_REST_02146] ara::rest::ogm::String::GetValue ⌈Table 8.141 describes the interface `ara::rest::ogm::String::GetValue`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.13.8 SetValue

Service name:	ara::rest::ogm::String::SetValue	
Type:	`Member function`	
Syntax:	`void ara::rest::ogm::String::SetValue(const ValueType &v)`	
Function param:	v	a value
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/string.h	
Class:	ara::rest::ogm::String	
Description:	Sets the current value from a C++ data type.	

Table 8.142: ara::rest::ogm::String::SetValue

[SWS_REST_02147] ara::rest::ogm::String::SetValue ⌈Table 8.142 describes the interface `ara::rest::ogm::String::SetValue`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.13.9 Make

Service name:	ara::rest::ogm::String::Make	
Type:	`Member function`	

Syntax:	`template <typename... Ts>`
	`static Pointer<SelfType>`
	`ara::rest::ogm::String::Make(Ts &&...ts)`	

Function param:	ts	constructor arguments forwarded to the constructor of this type
Return value:	a pointer to a node of type SelfType	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/string.h	
Class:	ara::rest::ogm::String	
Description:	Creates a node of type SelfType.	

Table 8.143: ara::rest::ogm::String::Make

[SWS_REST_02148] ara::rest::ogm::String::Make ⌈Table 8.143 describes the interface `ara::rest::ogm::String::Make`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.13.10 Make

Service name:	ara::rest::ogm::String::Make
Type:	`Member function`

Syntax:	`template <typename... Ts>`
	`static Pointer<SelfType>`
	`ara::rest::ogm::String::Make(Allocator *alloc, Ts &&...ts)`

Function param:	alloc	an allocator to use to construct this node
Function param:	ts	constructor arguments forwarded to the constructor of this type
Return value:	a pointer to a node of type SelfType	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/string.h	
Class:	ara::rest::ogm::String	
Description:	Creates a node of type SelfType.	

Table 8.144: ara::rest::ogm::String::Make

[SWS_REST_02149] ara::rest::ogm::String::Make ⌈Table 8.144 describes the interface `ara::rest::ogm::String::Make`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.13.11 String

Service name:	ara::rest::ogm::String::String	
Type:	`Member function`	
Syntax:	`ara::rest::ogm::String::String(ValueType value=ValueType{})`	
Function param:	value	an intial value
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/string.h	
Class:	ara::rest::ogm::String	
Description:	Connstructs an String.	

Table 8.145: ara::rest::ogm::String::String

[SWS_REST_02150] ara::rest::ogm::String::String ⌈Table 8.145 describes the interface `ara::rest::ogm::String::String`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.13.12 String

Service name:	ara::rest::ogm::String::String	
Type:	`Member function`	
Syntax:	`ara::rest::ogm::String::String(Allocator *alloc, ValueType value=ValueType{})`	
Function param:	alloc	an allocator
Function param:	value	an intial value
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/string.h	
Class:	ara::rest::ogm::String	
Description:	Connstructs an String.	

Table 8.146: ara::rest::ogm::String::String

[SWS_REST_02151] ara::rest::ogm::String::String ⌈Table 8.146 describes the interface `ara::rest::ogm::String::String`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.14 ara::rest::ogm::Value

[SWS_REST_02152] ⌈`ara::rest::ogm::Value` class shall be declared in the `ara/rest/ogm/value.h` header file:

```
1 class ara::rest::ogm::Value : public ara::rest::ogm::Node;
```

⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307)*

8.14.1 SelfType

Name:	`SelfType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Value::SelfType = Value`
Header file:	ara/rest/ogm/value.h
Class:	ara::rest::ogm::Value
Description:	Its own type.

Table 8.147: ara::rest::ogm::Value::SelfType

[SWS_REST_02153] SelfType ⌈Table 8.147 describes the type alias `ara::rest::ogm::Value::SelfType`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307)*

8.14.2 ParentType

Name:	`ParentType`
Type:	`Member type alias`
Syntax:	`using ara::rest::ogm::Value::ParentType = Node`
Header file:	ara/rest/ogm/value.h
Class:	ara::rest::ogm::Value
Description:	Type of its parent in the OGM type hierarchy.

Table 8.148: ara::rest::ogm::Value::ParentType

[SWS_REST_02154] **ParentType** ⌈Table 8.148 describes the type alias `ara::rest::ogm::Value::ParentType`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307)*

8.14.3 GetParent

Service name:	ara::rest::ogm::Value::GetParent
Type:	`Member function`
Syntax:	`Node* ara::rest::ogm::Value::GetParent()`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/value.h
Class:	ara::rest::ogm::Value
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.149: ara::rest::ogm::Value::GetParent

[SWS_REST_02155] **ara::rest::ogm::Value::GetParent** ⌈Table 8.149 describes the interface `ara::rest::ogm::Value::GetParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.14.4 GetParent

Service name:	ara::rest::ogm::Value::GetParent
Type:	`Member function`
Syntax:	`const Node* ara::rest::ogm::Value::GetParent() const`
Function param:	None
Return value:	a pointer to its parent node
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/value.h
Class:	ara::rest::ogm::Value
Description:	Returns a (strongly-typed) pointer to its parent node.

Table 8.150: ara::rest::ogm::Value::GetParent

[SWS_REST_02156] **ara::rest::ogm::Value::GetParent** ⌈Table 8.150 describes the interface `ara::rest::ogm::Value::GetParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.14.5 HasParent

Service name:	ara::rest::ogm::Value::HasParent
Type:	`Member function`
Syntax:	`bool ara::rest::ogm::Value::HasParent() const`

Function param:	None
Return value:	true if this node has a structural parent
Exceptions:	`noexcept`
Header file:	ara/rest/ogm/value.h
Class:	ara::rest::ogm::Value
Description:	Denotes whether this node has a structural parent.

Table 8.151: ara::rest::ogm::Value::HasParent

[SWS_REST_02157] ara::rest::ogm::Value::HasParent ⌈Table 8.151 describes the interface `ara::rest::ogm::Value::HasParent`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.14.6 Value

Service name:	ara::rest::ogm::Value::Value
Type:	`Member function`
Syntax:	`ara::rest::ogm::Value::Value()`
Function param:	None
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/ogm/value.h
Class:	ara::rest::ogm::Value
Description:	Constructs a node. Inaccessible to the user

Table 8.152: ara::rest::ogm::Value::Value

[SWS_REST_02158] ara::rest::ogm::Value::Value ⌈Table 8.152 describes the interface `ara::rest::ogm::Value::Value`.⌋*(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.15 ara::rest::Pattern

[SWS_REST_02159] ⌈`ara::rest::Pattern` class shall be declared in the `ara/rest/routing.h` header file:

```
1 class ara::rest::Pattern;
```

⌋*(RS_CM_00300, RS_CM_00309)*

8.15.1 Pattern

Service name:	ara::rest::Pattern::Pattern
Type:	`Member function`
Syntax:	`ara::rest::Pattern::Pattern(StringView pat)`

Function param:	pat	a pattern string
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/routing.h	
Class:	ara::rest::Pattern	
Description:	Constructs a Pattern.	

Table 8.153: ara::rest::Pattern::Pattern

[SWS_REST_02160] ara::rest::Pattern::Pattern ⌈Table 8.153 describes the interface `ara::rest::Pattern::Pattern`.⌋*(RS_CM_00300, RS_CM_00309)*

8.15.2 operator==

Service name:	ara::rest::Pattern::operator==	
Type:	`Non-member function`	
Syntax:	`friend bool operator==(const Pattern &a, const Pattern &b)`	
Function param:	a	a Pattern
Function param:	b	a Patern
Return value:	true if arguments compare equal	
Exceptions:	`noexcept`	
Header file:	ara/rest/routing.h	
Namespace:	ara::rest::Pattern	
Description:	Compares patterns for equality.	

Table 8.154: ara::rest::Pattern::operator==

[SWS_REST_02161] ara::rest::Pattern::operator== ⌈Table 8.154 describes the interface `ara::rest::Pattern::operator==`.⌋*(RS_CM_00300, RS_CM_00309)*

8.15.3 operator!=

Service name:	ara::rest::Pattern::operator!=	
Type:	`Non-member function`	
Syntax:	`friend bool operator!=(const Pattern &a, const Pattern &b)`	
Function param:	a	a Pattern
Function param:	b	a Patern
Return value:	true if arguments compare inequal	
Exceptions:	`noexcept`	
Header file:	ara/rest/routing.h	
Namespace:	ara::rest::Pattern	
Description:	Compares patterns for inequality.	

Table 8.155: ara::rest::Pattern::operator!=

[SWS_REST_02162] ara::rest::Pattern::operator!= ⌈Table 8.155 describes the interface `ara::rest::Pattern::operator!=`.⌋*(RS_CM_00300, RS_CM_00309)*

8.15.4 operator<

Service name:	ara::rest::Pattern::operator<	
Type:	`Non-member function`	
Syntax:	`friend bool operator<(const Pattern &a, const Pattern &b)`	
Function param:	a	a Pattern
Function param:	b	a Patern
Return value:	true if 'a' is less-than 'b' accoring to Pattern order critera.	
Exceptions:	`noexcept`	
Header file:	ara/rest/routing.h	
Namespace:	ara::rest::Pattern	
Description:	Compares patterns for order.	

Table 8.156: ara::rest::Pattern::operator<

[SWS_REST_02163] ara::rest::Pattern::operator< ⌈Table 8.156 describes the interface `ara::rest::Pattern::operator<`.⌋*(RS_CM_00300, RS_CM_00309)*

8.16 ara::rest::ReplyHeader

[SWS_REST_02164] ⌈`ara::rest::ReplyHeader` class shall be declared in the `ara/rest/header.h` header file:

```
1 class ara::rest::ReplyHeader;
```

⌋*(RS_CM_00300)*

8.16.1 GetStatus

Service name:	ara::rest::ReplyHeader::GetStatus
Type:	`Member function`
Syntax:	`int ara::rest::ReplyHeader::GetStatus() const`
Function param:	None
Return value:	a status code
Exceptions:	`noexcept`
Header file:	ara/rest/header.h
Class:	ara::rest::ReplyHeader
Description:	Returns the current message status code. Status codes are binding-specific

Table 8.157: ara::rest::ReplyHeader::GetStatus

[SWS_REST_02165] ara::rest::ReplyHeader::GetStatus ⌈Table 8.157 describes the interface `ara::rest::ReplyHeader::GetStatus`.⌋*(RS_CM_00300)*

8.16.2 SetStatus

Service name:	ara::rest::ReplyHeader::SetStatus	
Type:	`Member function`	
Syntax:	`void ara::rest::ReplyHeader::SetStatus(int code) const`	
Function param:	code	an integral status code
Return value:	None	
Exceptions:	`noexcept`	
Header file:	ara/rest/header.h	
Class:	ara::rest::ReplyHeader	
Description:	Sets a message status code. Status codes are binding-specific	

Table 8.158: ara::rest::ReplyHeader::SetStatus

[SWS_REST_02166] ara::rest::ReplyHeader::SetStatus ⌈Table 8.158 describes the interface `ara::rest::ReplyHeader::SetStatus`.⌋*(RS_CM_00300)*

8.16.3 GetUri

Service name:	ara::rest::ReplyHeader::GetUri
Type:	`Member function`
Syntax:	`const Uri& ara::rest::ReplyHeader::GetUri() const`
Function param:	None
Return value:	a Uri
Exceptions:	`noexcept`
Header file:	ara/rest/header.h
Class:	ara::rest::ReplyHeader
Description:	Returns a Uri. It is binding-specific how Uri map to the transport protocol format.

Table 8.159: ara::rest::ReplyHeader::GetUri

[SWS_REST_02167] ara::rest::ReplyHeader::GetUri ⌈Table 8.159 describes the interface `ara::rest::ReplyHeader::GetUri`.⌋*(RS_CM_00300, RS_CM_00304)*

8.16.4 SetUri

Service name:	ara::rest::ReplyHeader::SetUri	
Type:	`Member function`	
Syntax:	`void ara::rest::ReplyHeader::SetUri(const Uri &uri)`	
Function param:	uri	a Uri
Return value:	None	

Exceptions:	Implementation-defined
Header file:	ara/rest/header.h
Class:	ara::rest::ReplyHeader
Description:	Allows to set a Uri. It is binding-specific how Uri map to the transport protocol format.

Table 8.160: ara::rest::ReplyHeader::SetUri

[SWS_REST_02168] ara::rest::ReplyHeader::SetUri ⌈Table 8.160 describes the interface `ara::rest::ReplyHeader::SetUri`.⌋*(RS_CM_00300, RS_CM_00304)*

8.17 ara::rest::Reply

[SWS_REST_02169] ⌈`ara::rest::Reply` class shall be declared in the `ara/rest/client.h` header file:

```
1 class ara::rest::Reply;
```

⌋*(RS_CM_00300)*

8.17.1 Reply

Service name:	ara::rest::Reply::Reply
Type:	`Member function`
Syntax:	`ara::rest::Reply::Reply(const Reply &)=delete`
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/client.h
Class:	ara::rest::Reply
Description:	Non-copyable.

Table 8.161: ara::rest::Reply::Reply

[SWS_REST_02170] ara::rest::Reply::Reply ⌈Table 8.161 describes the interface `ara::rest::Reply::Reply`.⌋*(RS_CM_00300)*

8.17.2 operator=

Service name:	ara::rest::Reply::operator=
Type:	`Member function`
Syntax:	`Reply& ara::rest::Reply::operator=(const Reply &)=delete`
Return value:	a value of type `Reply &`
Exceptions:	Implementation-defined
Header file:	ara/rest/client.h

Class:	ara::rest::Reply
Description:	Non-copy-assignable.

Table 8.162: ara::rest::Reply::operator=

[SWS_REST_02171] ara::rest::Reply::operator= ⌈Table 8.162 describes the interface `ara::rest::Reply::operator=`.⌋*(RS_CM_00300)*

8.17.3 GetHeader

Service name:	ara::rest::Reply::GetHeader
Type:	`Member function`
Syntax:	`ReplyHeader const& ara::rest::Reply::GetHeader() const`
Function param:	None
Return value:	a reference to a ReplyHeader
Exceptions:	Implementation-defined
Header file:	ara/rest/client.h
Class:	ara::rest::Reply
Description:	Obtains the message header. Requests the message header from the endpoint. Accessing the message header is always synchronous.

Table 8.163: ara::rest::Reply::GetHeader

[SWS_REST_02172] ara::rest::Reply::GetHeader ⌈Table 8.163 describes the interface `ara::rest::Reply::GetHeader`.⌋*(RS_CM_00300)*

8.17.4 GetObject

Service name:	ara::rest::Reply::GetObject
Type:	`Member function`
Syntax:	`Task<ogm::Object const&> ara::rest::Reply::GetObject() const`
Function param:	None
Return value:	returns a task waiting for the message payload to be received.
Exceptions:	Implementation-defined
Header file:	ara/rest/client.h
Class:	ara::rest::Reply
Description:	Obtains the reply message payload.

Table 8.164: ara::rest::Reply::GetObject

[SWS_REST_02173] ara::rest::Reply::GetObject ⌈Table 8.164 describes the interface `ara::rest::Reply::GetObject`.⌋*(RS_CM_00300)*

8.17.5 ReleaseObject

Service name:	ara::rest::Reply::ReleaseObject
Type:	`Member function`
Syntax:	`Task<Pointer<ogm::Object> > ara::rest::Reply::ReleaseObject()`
Function param:	None
Return value:	returns a task waiting for the message payload to be received.
Exceptions:	Implementation-defined
Header file:	ara/rest/client.h
Class:	ara::rest::Reply
Description:	Obtains the reply message payload.

Table 8.165: ara::rest::Reply::ReleaseObject

[SWS_REST_02174] ara::rest::Reply::ReleaseObject ⌈Table 8.165 describes the interface `ara::rest::Reply::ReleaseObject`.⌋*(RS_CM_00300)*

8.18 ara::rest::RequestHeader

[SWS_REST_02175] ⌈`ara::rest::RequestHeader` class shall be declared in the `ara/rest/header.h` header file:

```
1 class ara::rest::RequestHeader;
```

⌋*(RS_CM_00300)*

8.18.1 GetMethod

Service name:	ara::rest::RequestHeader::GetMethod
Type:	`Member function`
Syntax:	`RequestMethod ara::rest::RequestHeader::GetMethod() const`
Function param:	None
Return value:	a request method
Exceptions:	`noexcept`
Header file:	ara/rest/header.h
Class:	ara::rest::RequestHeader
Description:	Returns the request method.

Table 8.166: ara::rest::RequestHeader::GetMethod

[SWS_REST_02176] ara::rest::RequestHeader::GetMethod ⌈Table 8.166 describes the interface `ara::rest::RequestHeader::GetMethod`.⌋*(RS_CM_00300)*

8.18.2 SetMethod

Service name:	ara::rest::RequestHeader::SetMethod	
Type:	`Member function`	
Syntax:	`void ara::rest::RequestHeader::SetMethod(RequestMethod met)`	
Function param:	met	a RequestMethod
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/header.h	
Class:	ara::rest::RequestHeader	
Description:	Allows to set the request method.	

Table 8.167: ara::rest::RequestHeader::SetMethod

[SWS_REST_02177] ara::rest::RequestHeader::SetMethod ⌈Table 8.167 describes the interface `ara::rest::RequestHeader::SetMethod`.⌋*(RS_CM_00300)*

8.18.3 GetUri

Service name:	ara::rest::RequestHeader::GetUri
Type:	`Member function`
Syntax:	`const Uri& ara::rest::RequestHeader::GetUri() const`
Function param:	None
Return value:	a Uri
Exceptions:	`noexcept`
Header file:	ara/rest/header.h
Class:	ara::rest::RequestHeader
Description:	Returns a Uri.

Table 8.168: ara::rest::RequestHeader::GetUri

[SWS_REST_02178] ara::rest::RequestHeader::GetUri ⌈Table 8.168 describes the interface `ara::rest::RequestHeader::GetUri`.⌋*(RS_CM_00300, RS_CM_00304)*

8.18.4 SetUri

Service name:	ara::rest::RequestHeader::SetUri	
Type:	`Member function`	
Syntax:	`void ara::rest::RequestHeader::SetUri(const Uri &uri)`	
Function param:	uri	a Uri
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/header.h	
Class:	ara::rest::RequestHeader	
Description:	Allows to set a Uri.	

Table 8.169: ara::rest::RequestHeader::SetUri

[SWS_REST_02179] **ara::rest::RequestHeader::SetUri** ⌈Table 8.169 describes the interface `ara::rest::RequestHeader::SetUri`.⌋*(RS_CM_00300, RS_CM_00304)*

8.19 ara::rest::Request

[SWS_REST_02180] ⌈`ara::rest::Request` class shall be declared in the `ara/rest/client.h` header file:

```
1 class ara::rest::Request;
```

⌋*(RS_CM_00300)*

8.19.1 Request

Service name:	ara::rest::Request::Request
Type:	`Member function`
Syntax:	`ara::rest::Request::Request(const Request &)=delete`
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/client.h
Class:	ara::rest::Request
Description:	Non-copyable.

Table 8.170: ara::rest::Request::Request

[SWS_REST_02181] **ara::rest::Request::Request** ⌈Table 8.170 describes the interface `ara::rest::Request::Request`.⌋*(RS_CM_00300)*

8.19.2 operator=

Service name:	ara::rest::Request::operator=
Type:	`Member function`
Syntax:	`Request& ara::rest::Request::operator=(const Request &)=delete`
Return value:	a value of type `Request &`
Exceptions:	Implementation-defined
Header file:	ara/rest/client.h
Class:	ara::rest::Request
Description:	Non-copy-assignable.

Table 8.171: ara::rest::Request::operator=

[SWS_REST_02182] ara::rest::Request::operator= ⌈Table 8.171 describes the interface `ara::rest::Request::operator=`.⌋*(RS_CM_00300)*

8.19.3 Request

Service name:	ara::rest::Request::Request	
Type:	`Member function`	
Syntax:	`ara::rest::Request::Request(RequestMethod met, const Uri &uri)`	
Function param:	met	one of RequestMethod
Function param:	uri	a Uri
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/client.h	
Class:	ara::rest::Request	
Description:	Constructs a Request.	

Table 8.172: ara::rest::Request::Request

[SWS_REST_02183] ara::rest::Request::Request ⌈Table 8.172 describes the interface `ara::rest::Request::Request`.⌋*(RS_CM_00300)*

8.19.4 Request

Service name:	ara::rest::Request::Request	
Type:	`Member function`	
Syntax:	`ara::rest::Request::Request(RequestMethod met, Uri &&uri)`	
Function param:	met	one of RequestMethod
Function param:	uri	a Uri
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/client.h	
Class:	ara::rest::Request	
Description:	Constructs a Request.	

Table 8.173: ara::rest::Request::Request

[SWS_REST_02184] ara::rest::Request::Request ⌈Table 8.173 describes the interface `ara::rest::Request::Request`.⌋*(RS_CM_00300)*

8.19.5 Request

Service name:	ara::rest::Request::Request
Type:	`Member function`

Syntax:	`ara::rest::Request::Request(RequestMethod met, const Uri &uri, const Pointer< ogm::Object > &obj)`	
Function param:	met	one of RequestMethod
Function param:	uri	a Uri
Function param:	obj	data payload of request message
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/client.h	
Class:	ara::rest::Request	
Description:	Constructs a Request.	

Table 8.174: ara::rest::Request::Request

[SWS_REST_02185] ara::rest::Request::Request ⌈Table 8.174 describes the interface `ara::rest::Request::Request`.⌋*(RS_CM_00300)*

8.19.6 Request

Service name:	ara::rest::Request::Request	
Type:	`Member function`	
Syntax:	`ara::rest::Request::Request(RequestMethod met, const Uri &uri, Pointer< ogm::Object > &&obj)`	
Function param:	met	one of RequestMethod
Function param:	uri	a Uri
Function param:	obj	data payload of request message
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/client.h	
Class:	ara::rest::Request	
Description:	Constructs a Request.	

Table 8.175: ara::rest::Request::Request

[SWS_REST_02186] ara::rest::Request::Request ⌈Table 8.175 describes the interface `ara::rest::Request::Request`.⌋*(RS_CM_00300)*

8.19.7 Request

Service name:	ara::rest::Request::Request	
Type:	`Member function`	
Syntax:	`ara::rest::Request::Request(RequestMethod met, Uri &&uri, const Pointer< ogm::Object > &obj)`	
Function param:	met	one of RequestMethod
Function param:	uri	a Uri
Function param:	obj	data payload of request message
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/client.h	

Class:	ara::rest::Request
Description:	Constructs a Request.

Table 8.176: ara::rest::Request::Request

[SWS_REST_02187] ara::rest::Request::Request ⌈Table 8.176 describes the interface `ara::rest::Request::Request`.⌋*(RS_CM_00300)*

8.19.8 Request

Service name:	ara::rest::Request::Request	
Type:	`Member function`	
Syntax:	`ara::rest::Request::Request(RequestMethod met, Uri &&uri, Pointer< ogm::Object > &&obj)`	
Function param:	met	one of RequestMethod
Function param:	uri	a Uri
Function param:	obj	data payload of request message
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/client.h	
Class:	ara::rest::Request	
Description:	Constructs a Request.	

Table 8.177: ara::rest::Request::Request

[SWS_REST_02188] ara::rest::Request::Request ⌈Table 8.177 describes the interface `ara::rest::Request::Request`.⌋*(RS_CM_00300)*

8.20 ara::rest::Router

[SWS_REST_02189] ⌈`ara::rest::Router` class shall be declared in the `ara/rest/routing.h` header file:

```
1 class ara::rest::Router;
```

⌋*(RS_CM_00300)*

8.20.1 RouteHandlerType

Name:	`RouteHandlerType`
Type:	`Member type alias`
Syntax:	`using ara::rest::Router::RouteHandlerType = void(const ServerRequest&, ServerReply&, const Matches&)`
Header file:	ara/rest/routing.h
Class:	ara::rest::Router

Description:	! User-define route handler function type

Table 8.178: ara::rest::Router::RouteHandlerType

[SWS_REST_02190] RouteHandlerType ⌈Table 8.178 describes the type alias `ara::rest::Router::RouteHandlerType`.⌋*(RS_CM_00300)*

8.20.2 RouteRange

Name:	`RouteRange`
Type:	`Member type alias`
Syntax:	`using ara::rest::Router::RouteRange = IteratorRange<unspecified_iterator_type>`
Header file:	ara/rest/routing.h
Class:	ara::rest::Router
Description:	! Iterator range of routes

Table 8.179: ara::rest::Router::RouteRange

[SWS_REST_02191] RouteRange ⌈Table 8.179 describes the type alias `ara::rest::Router::RouteRange`.⌋*(RS_CM_00300)*

8.20.3 ConstRouteRange

Name:	`ConstRouteRange`
Type:	`Member type alias`
Syntax:	`using ara::rest::Router::ConstRouteRange = IteratorRange<unspecified_iterator_type>`
Header file:	ara/rest/routing.h
Class:	ara::rest::Router
Description:	! Const iterator range of routes

Table 8.180: ara::rest::Router::ConstRouteRange

[SWS_REST_02192] ConstRouteRange ⌈Table 8.180 describes the type alias `ara::rest::Router::ConstRouteRange`.⌋*(RS_CM_00300)*

8.20.4 Router

Service name:	ara::rest::Router::Router	
Type:	`Member function`	
Syntax:	`ara::rest::Router::Router(Allocator *alloc=GetDefaultAllocator())`	
Function param:	alloc	an allocator for all internal dynamic memory requirements

Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/routing.h
Class:	ara::rest::Router
Description:	Constructs an empty Router.

Table 8.181: ara::rest::Router::Router

[SWS_REST_02193] ara::rest::Router::Router ⌈Table 8.181 describes the interface `ara::rest::Router::Router`.⌋ *(RS_CM_00300)*

8.20.5 Router

Service name:	ara::rest::Router::Router	
Type:	`Member function`	
Syntax:	`ara::rest::Router::Router(std::initializer_list< Route > routes, Allocator *alloc=GetDefaultAllocator())`	
Function param:	routes	a list of routes
Function param:	alloc	an allocator for all internal dynamic memory requirements
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/routing.h	
Class:	ara::rest::Router	
Description:	Constructs a router from a given list of routes.	

Table 8.182: ara::rest::Router::Router

[SWS_REST_02194] ara::rest::Router::Router ⌈Table 8.182 describes the interface `ara::rest::Router::Router`.⌋ *(RS_CM_00300)*

8.20.6 operator()

Service name:	ara::rest::Router::operator()	
Type:	`Member function`	
Syntax:	`void ara::rest::Router::operator()(const ServerRequest &req, ServerReply &rep) const`	
Function param:	req	a request
Function param:	rep	a reply
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/routing.h	
Class:	ara::rest::Router	
Description:	Request handler function. This function serves as the user-defined request handler function passed to Server	

Table 8.183: ara::rest::Router::operator()

[SWS_REST_02195] ara::rest::Router::operator() ⌈Table 8.183 describes the interface `ara::rest::Router::operator()`.⌋*(RS_CM_00300)*

8.20.7 InsertRoute

Service name:	ara::rest::Router::InsertRoute	
Type:	`Member function`	
Syntax:	`Router& ara::rest::Router::InsertRoute(const Route &route)`	
Function param:	route	a route
Return value:	a reference to this	
Exceptions:	Implementation-defined	
Header file:	ara/rest/routing.h	
Class:	ara::rest::Router	
Description:	Inserts a route into the set of potential matches. If a route already exists nothing is inserted.	

Table 8.184: ara::rest::Router::InsertRoute

[SWS_REST_02196] ara::rest::Router::InsertRoute ⌈Table 8.184 describes the interface `ara::rest::Router::InsertRoute`.⌋*(RS_CM_00300)*

8.20.8 EmplaceRoute

Service name:	ara::rest::Router::EmplaceRoute	
Type:	`Member function`	
Syntax:	`Router& ara::rest::Router::EmplaceRoute(RequestMethod met, Pattern pat, const Function< RouteHandlerType > &hnd)`	
Function param:	met	a set of request methods
Function param:	pat	a URI Pattern
Function param:	hnd	a user-defined routing handler
Return value:	a reference to this	
Exceptions:	Implementation-defined	
Header file:	ara/rest/routing.h	
Class:	ara::rest::Router	
Description:	Constructs a route in-place Similar to Insert except that the route is constructed in-place. The given arguments are forwarded to the internal Route. If such a route already exists nothing is inserted.	

Table 8.185: ara::rest::Router::EmplaceRoute

[SWS_REST_02197] ara::rest::Router::EmplaceRoute ⌈Table 8.185 describes the interface `ara::rest::Router::EmplaceRoute`.⌋*(RS_CM_00300)*

8.20.9 SetDefaultHandler

Service name:	ara::rest::Router::SetDefaultHandler	
Type:	`Member function`	
Syntax:	`Router& ara::rest::Router::SetDefaultHandler(const Function< Server::RequestHandlerType > &hnd)`	
Function param:	hnd	a user-defined request handler
Return value:	a reference to this.	
Exceptions:	Implementation-defined	
Header file:	ara/rest/routing.h	
Class:	ara::rest::Router	
Description:	Enables a user to set a default request handler The given handler is called if none of the routes matched of it at least once of the routes called Route::Default().	

Table 8.186: ara::rest::Router::SetDefaultHandler

[SWS_REST_02198] ara::rest::Router::SetDefaultHandler ⌈Table 8.186 describes the interface `ara::rest::Router::SetDefaultHandler`.⌋*(RS_CM_00300)*

8.20.10 RouteCount

Service name:	ara::rest::Router::RouteCount
Type:	`Member function`
Syntax:	`std::size_t ara::rest::Router::RouteCount()`
Function param:	None
Return value:	the number of user-defined routes
Exceptions:	Implementation-defined
Header file:	ara/rest/routing.h
Class:	ara::rest::Router
Description:	Returns the number of routes. Returns the number of specified routes, exclusive of the default route.

Table 8.187: ara::rest::Router::RouteCount

[SWS_REST_02199] ara::rest::Router::RouteCount ⌈Table 8.187 describes the interface `ara::rest::Router::RouteCount`.⌋*(RS_CM_00300)*

8.20.11 Routes

Service name:	ara::rest::Router::Routes
Type:	`Member function`
Syntax:	`RouteRange ara::rest::Router::Routes()`
Function param:	None
Return value:	an iterator range of routes
Exceptions:	Implementation-defined
Header file:	ara/rest/routing.h
Class:	ara::rest::Router
Description:	Provides direc access to the set of routes.

Table 8.188: ara::rest::Router::Routes

[SWS_REST_02200] ara::rest::Router::Routes ⌈Table 8.188 describes the interface `ara::rest::Router::Routes`.⌋ *(RS_CM_00300)*

8.20.12 Routes

Service name:	ara::rest::Router::Routes
Type:	`Member function`
Syntax:	`ConstRouteRange ara::rest::Router::Routes() const`
Function param:	None
Return value:	an iterator range of routes
Exceptions:	Implementation-defined
Header file:	ara/rest/routing.h
Class:	ara::rest::Router
Description:	Provides direc access to the set of routes.

Table 8.189: ara::rest::Router::Routes

[SWS_REST_02201] ara::rest::Router::Routes ⌈Table 8.189 describes the interface `ara::rest::Router::Routes`.⌋ *(RS_CM_00300)*

8.20.13 RemoveRoute

Service name:	ara::rest::Router::RemoveRoute	
Type:	`Member function`	
Syntax:	`void ara::rest::Router::RemoveRoute(RouteRange::Iterator iter)`	
Function param:	iter	iterator referencing the route to remove
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/routing.h	
Class:	ara::rest::Router	
Description:	Removes a route from the set.	

Table 8.190: ara::rest::Router::RemoveRoute

[SWS_REST_02202] ara::rest::Router::RemoveRoute ⌈Table 8.190 describes the interface `ara::rest::Router::RemoveRoute`.⌋ *(RS_CM_00300)*

8.20.14 FindRoute

Service name:	ara::rest::Router::FindRoute	
Type:	`Member function`	
Syntax:	`RouteRange::Iterator ara::rest::Router::FindRoute(const Route &route)`	
Function param:	route	route to search for

Return value:	an iterator to the route if it exists in the set or Routes.end() if no such route was found.
Exceptions:	Implementation-defined
Header file:	ara/rest/routing.h
Class:	ara::rest::Router
Description:	Searches for a given route.

Table 8.191: ara::rest::Router::FindRoute

[SWS_REST_02203] ara::rest::Router::FindRoute ⌈Table 8.191 describes the interface `ara::rest::Router::FindRoute`.⌋*(RS_CM_00300)*

8.20.15 Clear

Service name:	ara::rest::Router::Clear
Type:	`Member function`
Syntax:	`void ara::rest::Router::Clear()`
Function param:	None
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/routing.h
Class:	ara::rest::Router
Description:	Removes all routes.

Table 8.192: ara::rest::Router::Clear

[SWS_REST_02204] ara::rest::Router::Clear ⌈Table 8.192 describes the interface `ara::rest::Router::Clear`.⌋*(RS_CM_00300)*

8.21 ara::rest::Route

[SWS_REST_02205] ⌈`ara::rest::Route` class shall be declared in the `ara/rest/routing.h` header file:

```
1 class ara::rest::Route;
```

⌋*(RS_CM_00300)*

8.21.1 Upshot

Name:	`Upshot`	
Type:	`Member enumeration`	

Range:	`Accept`
	`Yield`
	`Default`	

Syntax:	`enum class Upshot { Accept, Yield, Default };`
Header file:	ara/rest/routing.h
Class:	ara::rest::Route
Description:	Instructions for a Router on how to proceed after a route handler functions returns. A route handler function must return one of these values to instruct the router how to proceed after executing the current handler.

Table 8.193: ara::rest::Route::Upshot

[SWS_REST_02206] Upshot ⌈Table 8.193 describes the enumeration datatype `ara::rest::Route::Upshot`.⌋*(RS_CM_00300)*

8.21.2 RouteHandlerType

Name:	`RouteHandlerType`
Type:	`Member type alias`
Syntax:	`using ara::rest::Route::RouteHandlerType = Upshot(const ServerRequest&, ServerReply&, const Matches&)`
Header file:	ara/rest/routing.h
Class:	ara::rest::Route
Description:	The type of the user-define handler function to be invoked if this Route matches the Pattern.

Table 8.194: ara::rest::Route::RouteHandlerType

[SWS_REST_02207] RouteHandlerType ⌈Table 8.194 describes the type alias `ara::rest::Route::RouteHandlerType`.⌋*(RS_CM_00300)*

8.21.3 Route

Service name:	ara::rest::Route::Route	
Type:	`Member function`	
Syntax:	`ara::rest::Route::Route(RequestMethod met, const Pattern &pat, const Function< RouteHandlerType > &hnd)`	
Function param:	met	a disjunction (logical OR) of RequestMethods to match against
Function param:	pat	a URI Pattern to match against
Function param:	hnd	a user-defined handler
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/routing.h	
Class:	ara::rest::Route	
Description:	Constructs a route.	

Table 8.195: ara::rest::Route::Route

[SWS_REST_02208] ara::rest::Route::Route ⌈Table 8.195 describes the interface `ara::rest::Route::Route`.⌋*(RS_CM_00300)*

8.21.4 operator()

Service name:	ara::rest::Route::operator()	
Type:	`Member function`	
Syntax:	`Upshot ara::rest::Route::operator()(const ServerRequest &req, ServerReply &rep) const`	
Function param:	req	a request
Function param:	rep	a reply
Return value:	a value of type `Upshot`	
Exceptions:	`noexcept`	
Header file:	ara/rest/routing.h	
Class:	ara::rest::Route	
Description:	ara::rest::Server compliant handler function This function is invoked by the Router to test the current Route for a match	

Table 8.196: ara::rest::Route::operator()

[SWS_REST_02209] ara::rest::Route::operator() ⌈Table 8.196 describes the interface `ara::rest::Route::operator()`.⌋*(RS_CM_00300)*

8.21.5 GetRequestMethod

Service name:	ara::rest::Route::GetRequestMethod
Type:	`Member function`
Syntax:	`RequestMethod ara::rest::Route::GetRequestMethod() const`
Function param:	None
Return value:	reference to a Pattern
Exceptions:	`noexcept`
Header file:	ara/rest/routing.h
Class:	ara::rest::Route
Description:	Provides access to the underlying Pattern object.

Table 8.197: ara::rest::Route::GetRequestMethod

[SWS_REST_02210] ara::rest::Route::GetRequestMethod ⌈Table 8.197 describes the interface `ara::rest::Route::GetRequestMethod`.⌋*(RS_CM_00300)*

8.21.6 GetPattern

Service name:	ara::rest::Route::GetPattern
Type:	`Member function`
Syntax:	`const Pattern& ara::rest::Route::GetPattern() const`
Function param:	None
Return value:	reference to a Pattern
Exceptions:	`noexcept`
Header file:	ara/rest/routing.h
Class:	ara::rest::Route
Description:	Provides access to the underlying Pattern object.

Table 8.198: ara::rest::Route::GetPattern

[SWS_REST_02211] ara::rest::Route::GetPattern ⌈Table 8.198 describes the interface `ara::rest::Route::GetPattern`.⌋ *(RS_CM_00300)*

8.21.7 operator==

Service name:	ara::rest::Route::operator==	
Type:	`Non-member function`	
Syntax:	`friend bool operator==(const Route &a, const Route &b)`	
Function param:	a	a route
Function param:	b	a route
Return value:	true if equal	
Exceptions:	`noexcept`	
Header file:	ara/rest/routing.h	
Namespace:	ara::rest::Route	
Description:	Tests for equality.	

Table 8.199: ara::rest::Route::operator==

[SWS_REST_02212] ara::rest::Route::operator== ⌈Table 8.199 describes the interface `ara::rest::Route::operator==`.⌋ *(RS_CM_00300)*

8.21.8 operator!=

Service name:	ara::rest::Route::operator!=	
Type:	`Non-member function`	
Syntax:	`friend bool operator!=(const Route &a, const Route &b)`	
Function param:	a	a route
Function param:	b	a route
Return value:	true if unequal	
Exceptions:	`noexcept`	
Header file:	ara/rest/routing.h	
Namespace:	ara::rest::Route	

Description:	Tests for inequality.

Table 8.200: ara::rest::Route::operator!=

[SWS_REST_02213] ara::rest::Route::operator!= ⌈Table 8.200 describes the interface `ara::rest::Route::operator!=`.⌋ *(RS_CM_00300)*

8.21.9 operator<

Service name:	ara::rest::Route::operator<	
Type:	`Non-member function`	
Syntax:	`friend bool operator<(const Route &a, const Route &b)`	
Function param:	a	a route
Function param:	b	a route
Return value:	true if a compares less-than b	
Exceptions:	`noexcept`	
Header file:	ara/rest/routing.h	
Namespace:	ara::rest::Route	
Description:	Tests whether a route is less-than another.	

Table 8.201: ara::rest::Route::operator<

[SWS_REST_02214] ara::rest::Route::operator< ⌈Table 8.201 describes the interface `ara::rest::Route::operator<`.⌋ *(RS_CM_00300)*

8.22 ara::rest::ServerEvent

[SWS_REST_02215] ⌈`ara::rest::ServerEvent` class shall be declared in the `ara/rest/server.h` header file:

```
1 class ara::rest::ServerEvent;
```

⌋ *(RS_CM_00300)*

8.22.1 ServerEvent

Service name:	ara::rest::ServerEvent::ServerEvent
Type:	`Member function`
Syntax:	`ara::rest::ServerEvent::ServerEvent(const ServerEvent &)=delete`
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/server.h
Class:	ara::rest::ServerEvent
Description:	Non-copyable.

Table 8.202: ara::rest::ServerEvent::ServerEvent

[SWS_REST_02216] ara::rest::ServerEvent::ServerEvent ⌈Table 8.202 describes the interface `ara::rest::ServerEvent::ServerEvent`.⌋*(RS_CM_00300)*

8.22.2 operator=

Service name:	ara::rest::ServerEvent::operator=
Type:	`Member function`
Syntax:	`ServerEvent& ara::rest::ServerEvent::operator=(const ServerEvent &)=delete`
Return value:	a value of type `ServerEvent &`
Exceptions:	Implementation-defined
Header file:	ara/rest/server.h
Class:	ara::rest::ServerEvent
Description:	Non-copy-assignable.

Table 8.203: ara::rest::ServerEvent::operator=

[SWS_REST_02217] ara::rest::ServerEvent::operator= ⌈Table 8.203 describes the interface `ara::rest::ServerEvent::operator=`.⌋*(RS_CM_00300)*

8.22.3 Notify

Service name:	ara::rest::ServerEvent::Notify
Type:	`Member function`
Syntax:	`Task<void> ara::rest::ServerEvent::Notify()`
Function param:	None
Return value:	a task waiting for the notification to complete.
Exceptions:	Implementation-defined
Header file:	ara/rest/server.h
Class:	ara::rest::ServerEvent
Description:	Issues a change notification to its corresponding Server. Each server-side event has at least one corresponding client-side event. Notify does NOT notify these clients. It notifies its corresponding server of potential changes to the data referenced by the event (URI).

Table 8.204: ara::rest::ServerEvent::Notify

[SWS_REST_02218] ara::rest::ServerEvent::Notify ⌈Table 8.204 describes the interface `ara::rest::ServerEvent::Notify`.⌋*(RS_CM_00300)*

8.22.4 Unsubscribe

Service name:	ara::rest::ServerEvent::Unsubscribe
Type:	`Member function`
Syntax:	`Task<void> ara::rest::ServerEvent::Unsubscribe()`
Function param:	None
Return value:	a value of type `Task< void >`
Exceptions:	Implementation-defined
Header file:	ara/rest/server.h
Class:	ara::rest::ServerEvent
Description:	Cancels an event subscription by issuing a cancelation request. A subscription can also be terminated (but not canceled) by destroying the correspond Event object.

Table 8.205: ara::rest::ServerEvent::Unsubscribe

[SWS_REST_02219] ara::rest::ServerEvent::Unsubscribe ⌈Table 8.205 describes the interface `ara::rest::ServerEvent::Unsubscribe`.⌋ *(RS_CM_00300)*

8.22.5 GetSubscriptionState

Service name:	ara::rest::ServerEvent::GetSubscriptionState
Type:	`Member function`
Syntax:	`SubscriptionState ara::rest::ServerEvent::GetSubscriptionState() const`
Function param:	None
Return value:	the current subscription state as perceived by the client
Exceptions:	`noexcept`
Header file:	ara/rest/server.h
Class:	ara::rest::ServerEvent
Description:	Denotes the current subscription state.

Table 8.206: ara::rest::ServerEvent::GetSubscriptionState

[SWS_REST_02220] ara::rest::ServerEvent::GetSubscriptionState ⌈Table 8.206 describes the interface `ara::rest::ServerEvent::GetSubscriptionState`.⌋ *(RS_CM_00300)*

8.22.6 GetUri

Service name:	ara::rest::ServerEvent::GetUri
Type:	`Member function`
Syntax:	`const Uri& ara::rest::ServerEvent::GetUri() const`
Function param:	None
Return value:	the Uri corresponding to this event subscription
Exceptions:	`noexcept`
Header file:	ara/rest/server.h
Class:	ara::rest::ServerEvent
Description:	Returns the event Uri.

Table 8.207: ara::rest::ServerEvent::GetUri

[SWS_REST_02221] ara::rest::ServerEvent::GetUri ⌈Table 8.207 describes the interface `ara::rest::ServerEvent::GetUri`.⌋*(RS_CM_00300)*

8.22.7 operator==

Service name:	ara::rest::ServerEvent::operator==	
Type:	`Non-member function`	
Syntax:	`friend bool operator==(const ServerEvent &a, const ServerEvent &b)`	
Function param:	a	an event
Function param:	b	an event
Return value:	true if a and b are equal	
Exceptions:	`noexcept`	
Header file:	ara/rest/server.h	
Namespace:	ara::rest::ServerEvent	
Description:	Tests events for equality.	

Table 8.208: ara::rest::ServerEvent::operator==

[SWS_REST_02222] ara::rest::ServerEvent::operator== ⌈Table 8.208 describes the interface `ara::rest::ServerEvent::operator==`.⌋*(RS_CM_00300)*

8.22.8 operator!=

Service name:	ara::rest::ServerEvent::operator!=	
Type:	`Non-member function`	
Syntax:	`friend bool operator!=(const ServerEvent &a, const ServerEvent &b)`	
Function param:	a	an event
Function param:	b	an event
Return value:	true if a and b are unequal	
Exceptions:	`noexcept`	
Header file:	ara/rest/server.h	
Namespace:	ara::rest::ServerEvent	
Description:	Tests events for inequality.	

Table 8.209: ara::rest::ServerEvent::operator!=

[SWS_REST_02223] ara::rest::ServerEvent::operator!= ⌈Table 8.209 describes the interface `ara::rest::ServerEvent::operator!=`.⌋*(RS_CM_00300)*

8.22.9 operator<

Service name:	ara::rest::ServerEvent::operator<	
Type:	`Non-member function`	
Syntax:	`friend bool operator<(const ServerEvent &a, const ServerEvent &b)`	
Function param:	a	an event
Function param:	b	an event
Return value:	true if a less-than b	
Exceptions:	`noexcept`	
Header file:	ara/rest/server.h	
Namespace:	ara::rest::ServerEvent	
Description:	Tests events for their partial order Order criterion is implementation-defined.	

Table 8.210: ara::rest::ServerEvent::operator<

[SWS_REST_02224] ara::rest::ServerEvent::operator< ⌈Table 8.210 describes the interface `ara::rest::ServerEvent::operator<`.⌋*(RS_CM_00300)*

8.23 ara::rest::ServerReply

[SWS_REST_02225] ⌈`ara::rest::ServerReply` class shall be declared in the `ara/rest/server.h` header file:

```
1 class ara::rest::ServerReply;
```

⌋*(RS_CM_00300)*

8.23.1 ServerReply

Service name:	ara::rest::ServerReply::ServerReply
Type:	`Member function`
Syntax:	`ara::rest::ServerReply::ServerReply(const ServerReply &)=delete`
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/server.h
Class:	ara::rest::ServerReply
Description:	Non-copyable.

Table 8.211: ara::rest::ServerReply::ServerReply

[SWS_REST_02226] ara::rest::ServerReply::ServerReply ⌈Table 8.211 describes the interface `ara::rest::ServerReply::ServerReply`.⌋*(RS_CM_00300)*

8.23.2 operator=

Service name:	ara::rest::ServerReply::operator=
Type:	`Member function`
Syntax:	`ServerReply& ara::rest::ServerReply::operator=(const ServerReply &)=delete`
Return value:	a value of type `ServerReply &`
Exceptions:	Implementation-defined
Header file:	ara/rest/server.h
Class:	ara::rest::ServerReply
Description:	Non-copy-assignable.

Table 8.212: ara::rest::ServerReply::operator=

[SWS_REST_02227] ara::rest::ServerReply::operator= ⌈Table 8.212 describes the interface `ara::rest::ServerReply::operator=`.⌋*(RS_CM_00300)*

8.23.3 GetHeader

Service name:	ara::rest::ServerReply::GetHeader
Type:	`Member function`
Syntax:	`ReplyHeader& ara::rest::ServerReply::GetHeader()`
Function param:	None
Return value:	a reference to a RequestHeader
Exceptions:	Implementation-defined
Header file:	ara/rest/server.h
Class:	ara::rest::ServerReply
Description:	Provides access to the reply message header.

Table 8.213: ara::rest::ServerReply::GetHeader

[SWS_REST_02228] ara::rest::ServerReply::GetHeader ⌈Table 8.213 describes the interface `ara::rest::ServerReply::GetHeader`.⌋*(RS_CM_00300)*

8.23.4 Send

Service name:	ara::rest::ServerReply::Send	
Type:	`Member function`	
Syntax:	`Task<void> ara::rest::ServerReply::Send(const Pointer< ogm::Object > &data={})`	
Function param:	data	payload to be transmitted
Return value:	a task waiting for the transmission to complete	
Exceptions:	Implementation-defined	
Header file:	ara/rest/server.h	
Class:	ara::rest::ServerReply	
Description:	Send a reply to the peer that has issued the request. If this function is not invoked explicitly, the endpoint will transmit a default reply. If Redirect() has been before called, these functions must be used.	

Table 8.214: ara::rest::ServerReply::Send

[SWS_REST_02229] ara::rest::ServerReply::Send ⌈Table 8.214 describes the interface `ara::rest::ServerReply::Send`.⌋ *(RS_CM_00300)*

8.23.5 Send

Service name:	ara::rest::ServerReply::Send	
Type:	`Member function`	
Syntax:	`Task<void> ara::rest::ServerReply::Send(const Pointer< ogm::Object > &&data)`	
Function param:	data	payload to be transmitted
Return value:	a task waiting for the transmission to complete	
Exceptions:	Implementation-defined	
Header file:	ara/rest/server.h	
Class:	ara::rest::ServerReply	
Description:	Send a reply to the peer that has issued the request. Same a other Send(), only with move semantics	

Table 8.215: ara::rest::ServerReply::Send

[SWS_REST_02230] ara::rest::ServerReply::Send ⌈Table 8.215 describes the interface `ara::rest::ServerReply::Send`.⌋ *(RS_CM_00300)*

8.23.6 Redirect

Service name:	ara::rest::ServerReply::Redirect	
Type:	`Member function`	
Syntax:	`Task<void> ara::rest::ServerReply::Redirect(const Uri &uri)`	
Function param:	uri	location to redirect to
Return value:	a value of type `Task< void >`	
Exceptions:	Implementation-defined	
Header file:	ara/rest/server.h	
Class:	ara::rest::ServerReply	
Description:	Issues a redirect command to the connected client. Must not be called after Send().	

Table 8.216: ara::rest::ServerReply::Redirect

[SWS_REST_02231] ara::rest::ServerReply::Redirect ⌈Table 8.216 describes the interface `ara::rest::ServerReply::Redirect`.⌋ *(RS_CM_00300)*

8.24 ara::rest::ServerRequest

[SWS_REST_02232] ⌈`ara::rest::ServerRequest` class shall be declared in the `ara/rest/server.h` header file:

```
1 class ara::rest::ServerRequest;
```

⌋(*RS_CM_00300*)

8.24.1 ServerRequest

Service name:	ara::rest::ServerRequest::ServerRequest
Type:	`Member function`
Syntax:	`ara::rest::ServerRequest::ServerRequest(const ServerRequest &)=delete`
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/server.h
Class:	ara::rest::ServerRequest
Description:	Non-copyable.

Table 8.217: ara::rest::ServerRequest::ServerRequest

[SWS_REST_02233] ara::rest::ServerRequest::ServerRequest ⌈Table 8.217 describes the interface `ara::rest::ServerRequest::ServerRequest`.⌋ (*RS_CM_00300*)

8.24.2 operator=

Service name:	ara::rest::ServerRequest::operator=
Type:	`Member function`
Syntax:	`ServerRequest& ara::rest::ServerRequest::operator=(const ServerRequest &)=delete`
Return value:	a value of type `ServerRequest &`
Exceptions:	Implementation-defined
Header file:	ara/rest/server.h
Class:	ara::rest::ServerRequest
Description:	Non-copy-assignable.

Table 8.218: ara::rest::ServerRequest::operator=

[SWS_REST_02234] ara::rest::ServerRequest::operator= ⌈Table 8.218 describes the interface `ara::rest::ServerRequest::operator=`.⌋(*RS_CM_00300*)

8.24.3 GetHeader

Service name:	ara::rest::ServerRequest::GetHeader
Type:	`Member function`
Syntax:	`RequestHeader const& ara::rest::ServerRequest::GetHeader() const`
Function param:	None
Return value:	a reference to a RequestHeader
Exceptions:	Implementation-defined
Header file:	ara/rest/server.h
Class:	ara::rest::ServerRequest
Description:	Provides access to the message header. Requests the message header from the endpoint. Accessing the message header is always synchronous.

Table 8.219: ara::rest::ServerRequest::GetHeader

[SWS_REST_02235] ara::rest::ServerRequest::GetHeader ⌈Table 8.219 describes the interface `ara::rest::ServerRequest::GetHeader`.⌋*(RS_CM_00300)*

8.24.4 GetObject

Service name:	ara::rest::ServerRequest::GetObject
Type:	`Member function`
Syntax:	`Task<ogm::Object const&> ara::rest::ServerRequest::GetObject() const`
Function param:	None
Return value:	returns a task waiting for the message payload to be received.
Exceptions:	Implementation-defined
Header file:	ara/rest/server.h
Class:	ara::rest::ServerRequest
Description:	Obtains the request message payload.

Table 8.220: ara::rest::ServerRequest::GetObject

[SWS_REST_02236] ara::rest::ServerRequest::GetObject ⌈Table 8.220 describes the interface `ara::rest::ServerRequest::GetObject`.⌋*(RS_CM_00300)*

8.24.5 ReleaseObject

Service name:	ara::rest::ServerRequest::ReleaseObject
Type:	`Member function`
Syntax:	`Task<Pointer<ogm::Object> > ara::rest::ServerRequest::ReleaseObject()`
Function param:	None
Return value:	returns a task waiting for the message payload to be received.
Exceptions:	Implementation-defined
Header file:	ara/rest/server.h

Class:	ara::rest::ServerRequest
Description:	Obtains the reply message payload.

Table 8.221: ara::rest::ServerRequest::ReleaseObject

[SWS_REST_02237] ara::rest::ServerRequest::ReleaseObject ⌈Table 8.221 describes the interface `ara::rest::ServerRequest::ReleaseObject`.⌋ *(RS_CM_00300)*

8.25 ara::rest::Server

[SWS_REST_02238] ⌈`ara::rest::Server` class shall be declared in the `ara/rest/server.h` header file:

```
1 class ara::rest::Server;
```

⌋*(RS_CM_00300, RS_CM_00301)*

8.25.1 RequestHandlerType

Name:	`RequestHandlerType`
Type:	`Member type alias`
Syntax:	`using ara::rest::Server::RequestHandlerType = void(const ServerRequest&, ServerReply&)`
Header file:	ara/rest/server.h
Class:	ara::rest::Server
Description:	Type of user-defined request handlers.

Table 8.222: ara::rest::Server::RequestHandlerType

[SWS_REST_02239] RequestHandlerType ⌈Table 8.222 describes the type alias `ara::rest::Server::RequestHandlerType`.⌋*(RS_CM_00300, RS_CM_00301)*

8.25.2 SubscriptionHandlerType

Name:	`SubscriptionHandlerType`
Type:	`Member type alias`
Syntax:	`using ara::rest::Server::SubscriptionHandlerType = void(ServerEvent)`
Header file:	ara/rest/server.h
Class:	ara::rest::Server
Description:	Denotes a subscription handler type.

Table 8.223: ara::rest::Server::SubscriptionHandlerType

[SWS_REST_02240] SubscriptionHandlerType ⌈Table 8.223 describes the type alias `ara::rest::Server::SubscriptionHandlerType`.⌋*(RS_CM_00300, RS_CM_00301)*

8.25.3 SubscriptionStateHandlerType

Name:	`SubscriptionStateHandlerType`
Type:	`Member type alias`
Syntax:	`using ara::rest::Server::SubscriptionStateHandlerType = void(const ServerEvent&, SubscriptionState)`
Header file:	ara/rest/server.h
Class:	ara::rest::Server
Description:	Denotes a callback to call if subscription status changes.

Table 8.224: ara::rest::Server::SubscriptionStateHandlerType

[SWS_REST_02241] SubscriptionStateHandlerType ⌈Table 8.224 describes the type alias `ara::rest::Server::SubscriptionStateHandlerType`.⌋ *(RS_CM_00300, RS_CM_00301)*

8.25.4 Server

Service name:	ara::rest::Server::Server
Type:	`Member function`
Syntax:	`ara::rest::Server::Server(const Server &)=delete`
Return value:	None
Exceptions:	Implementation-defined
Header file:	ara/rest/server.h
Class:	ara::rest::Server
Description:	Server is non-copy-constructible.

Table 8.225: ara::rest::Server::Server

[SWS_REST_02242] ara::rest::Server::Server ⌈Table 8.225 describes the interface `ara::rest::Server::Server`.⌋*(RS_CM_00300, RS_CM_00301)*

8.25.5 operator=

Service name:	ara::rest::Server::operator=
Type:	`Member function`
Syntax:	`Server& ara::rest::Server::operator=(const Server &)=delete`
Return value:	a value of type `Server &`
Exceptions:	Implementation-defined
Header file:	ara/rest/server.h

Class:	ara::rest::Server
Description:	Server is non-copy-assignable.

Table 8.226: ara::rest::Server::operator=

[SWS_REST_02243] ara::rest::Server::operator= ⌈Table 8.226 describes the interface `ara::rest::Server::operator=`.⌋ *(RS_CM_00300, RS_CM_00301)*

8.25.6 Server

Service name:	ara::rest::Server::Server	
Type:	`Member function`	
Syntax:	`ara::rest::Server::Server(const String &conf, const Function< RequestHandlerType > &hnd, Allocator *alloc=GetDefaultAllocator())`	
Function param:	conf	identifier for a corresponding configuration record
Function param:	hnd	request handler function
Function param:	alloc	allocator for dynamic memory
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/server.h	
Class:	ara::rest::Server	
Description:	Constructs a server.	

Table 8.227: ara::rest::Server::Server

[SWS_REST_02244] ara::rest::Server::Server ⌈Table 8.227 describes the interface `ara::rest::Server::Server`.⌋ *(RS_CM_00300, RS_CM_00301, RS_CM_00310)*

8.25.7 Start

Service name:	ara::rest::Server::Start	
Type:	`Member function`	
Syntax:	`Task<void> ara::rest::Server::Start(StartupPolicy policy=StartupPolicy::kDetached)`	
Function param:	policy	denotes whether caller is blocked or not.
Return value:	a task waiting for Stop() to be invoked	
Exceptions:	Implementation-defined	
Header file:	ara/rest/server.h	
Class:	ara::rest::Server	
Description:	Instruct a server to begin serving clients. A server does not serve anything unless Start() is invoked. A server can be started within the execution context of the caller or within its own execution context (usually this is a thread). If StartupPolicy::kAttached, then Start() blocks its caller and only returns of Stop() is called. If Startuppolicy::kDetached, Start() does not block its caller but returns a task for synchronization. The caller may wait on the task, which blocks until Stop() is invoked.	

Table 8.228: ara::rest::Server::Start

[SWS_REST_02245] ara::rest::Server::Start ⌈Table 8.228 describes the interface `ara::rest::Server::Start`.⌋*(RS_CM_00300, RS_CM_00301)*

8.25.8 Stop

Service name:	ara::rest::Server::Stop	
Type:	`Member function`	
Syntax:	`Task<void> ara::rest::Server::Stop(ShutdownPolicy policy=ShutdownPolicy::kGraceful)`	
Function param:	policy	denotes how server is stopped.
Return value:	return type	
Exceptions:	Implementation-defined	
Header file:	ara/rest/server.h	
Class:	ara::rest::Server	
Description:	Instructs a server to stop serving clients. A client can be stopped either as fast as possible or "gracefully". If ShutdownPolicy::kForced then all connections are terminates instantly and all ongoing processes shall be terminated as fast as possible. If ShutdownPolicy::kGraceful then a server will stop accepting new requests but will wait until all requests have been served.	

Table 8.229: ara::rest::Server::Stop

[SWS_REST_02246] ara::rest::Server::Stop ⌈Table 8.229 describes the interface `ara::rest::Server::Stop`.⌋*(RS_CM_00300, RS_CM_00301)*

8.25.9 ObserveSubscriptions

Service name:	ara::rest::Server::ObserveSubscriptions	
Type:	`Member function`	
Syntax:	`void ara::rest::Server::ObserveSubscriptions(const Function< SubscriptionHandlerType > &shnd, const Function< SubscriptionStateHandlerType > &sshnd)`	
Function param:	shnd	a subscription handler function
Function param:	sshnd	a subscription state handler function
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/server.h	
Class:	ara::rest::Server	
Description:	Registers a user-defined subscription handler. A server can handle event subscriptions by default. Unless a user-defined handler function is registered explicitly, event subscriptions are not visible to the user. This implies that subscriptions with EventPolicy::kTriggered never receive notifications.	

Table 8.230: ara::rest::Server::ObserveSubscriptions

[SWS_REST_02247] ara::rest::Server::ObserveSubscriptions ⌈Table 8.230 describes the interface `ara::rest::Server::ObserveSubscriptions`.⌋ *(RS_CM_00300, RS_CM_00301)*

8.25.10 GetError

Service name:	ara::rest::Server::GetError
Type:	`Member function`
Syntax:	`std::error_code ara::rest::Server::GetError() const`
Function param:	None
Return value:	a reference t the server Status
Exceptions:	`noexcept`
Header file:	ara/rest/server.h
Class:	ara::rest::Server
Description:	Obtain server status.

Table 8.231: ara::rest::Server::GetError

[SWS_REST_02248] ara::rest::Server::GetError ⌈Table 8.231 describes the interface `ara::rest::Server::GetError`.⌋ *(RS_CM_00300, RS_CM_00301)*

8.25.11 ObserveError

Service name:	ara::rest::Server::ObserveError	
Type:	`Member function`	
Syntax:	`void ara::rest::Server::ObserveError(const Function< void(std::error_code)> &hnd)`	
Function param:	hnd	user-defined handler function to to called on status changes
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/server.h	
Class:	ara::rest::Server	
Description:	Observe status changes.	

Table 8.232: ara::rest::Server::ObserveError

[SWS_REST_02249] ara::rest::Server::ObserveError ⌈Table 8.232 describes the interface `ara::rest::Server::ObserveError`.⌋ *(RS_CM_00300, RS_CM_00301)*

8.26 ara::rest::StdAllocator

[SWS_REST_02250] ⌈`ara::rest::StdAllocator` class shall be declared in the `ara/rest/allocator.h` header file:

```
1 template <typename T>
```

```
2 class ara::rest::StdAllocator;
```

⌋(*RS_CM_00300*)

8.26.1 value_type

Name:	`value_type`
Type:	`Member type alias`
Syntax:	`using ara::rest::StdAllocator< T >::value_type = T`
Header file:	ara/rest/allocator.h
Class:	ara::rest::StdAllocator
Description:	Type this allocator is bound to.

Table 8.233: ara::rest::StdAllocator::value_type

[SWS_REST_02251] value_type ⌈Table 8.233 describes the type alias `ara::rest::StdAllocator::value_type`.⌋(*RS_CM_00300*)

8.26.2 StdAllocator

Service name:	ara::rest::StdAllocator::StdAllocator
Type:	`Member function`
Syntax:	`ara::rest::StdAllocator< T >::StdAllocator()`
Function param:	None
Return value:	None
Exceptions:	`noexcept`
Header file:	ara/rest/allocator.h
Class:	ara::rest::StdAllocator
Description:	Default constructs this allocator See std::pmr::polymorphic_allocator documentation for details.

Table 8.234: ara::rest::StdAllocator::StdAllocator

[SWS_REST_02252] ara::rest::StdAllocator::StdAllocator ⌈Table 8.234 describes the interface `ara::rest::StdAllocator::StdAllocator`.⌋(*RS_CM_00300*)

8.26.3 StdAllocator

Service name:	ara::rest::StdAllocator::StdAllocator	
Type:	`Member function`	
Syntax:	`ara::rest::StdAllocator< T >::StdAllocator(Allocator *a)`	
Function param:	a	an allocator
Return value:	None	
Exceptions:	`noexcept`	
Header file:	ara/rest/allocator.h	

Class:	ara::rest::StdAllocator
Description:	Default constructs this allocator See std::pmr::polymorphic_allocator documentation for details.

Table 8.235: ara::rest::StdAllocator::StdAllocator

[SWS_REST_02253] ara::rest::StdAllocator::StdAllocator ⌈Table 8.235 describes the interface `ara::rest::StdAllocator::StdAllocator`.⌋*(RS_CM_00300)*

8.26.4 StdAllocator

Service name:	ara::rest::StdAllocator::StdAllocator	
Type:	`Member function`	
Syntax:	`template <typename U > ara::rest::StdAllocator< T >::StdAllocator(StdAllocator< U > const &do_not_use)`	
Function param:	do_not_use	meaningless.
Return value:	None	
Exceptions:	`noexcept`	
Header file:	ara/rest/allocator.h	
Class:	ara::rest::StdAllocator	
Description:	Do not call. See detailed API description. This function exists only for the sake of 'noexcept'. never invoked directly.	

Table 8.236: ara::rest::StdAllocator::StdAllocator

[SWS_REST_02254] ara::rest::StdAllocator::StdAllocator ⌈Table 8.236 describes the interface `ara::rest::StdAllocator::StdAllocator`.⌋*(RS_CM_00300)*

8.26.5 allocate

Service name:	ara::rest::StdAllocator::allocate	
Type:	`Member function`	
Syntax:	`value_type* ara::rest::StdAllocator< T >::allocate(std::size_t n)`	
Function param:	n	number of bytes to allocator
Return value:	a value of type `value_type *`	
Exceptions:	Implementation-defined	
Header file:	ara/rest/allocator.h	
Class:	ara::rest::StdAllocator	
Description:	Allocate	

Table 8.237: ara::rest::StdAllocator::allocate

[SWS_REST_02255] ara::rest::StdAllocator::allocate ⌈Table 8.237 describes the interface `ara::rest::StdAllocator::allocate`.⌋*(RS_CM_00300)*

8.26.6 deallocate

Service name:	ara::rest::StdAllocator::deallocate	
Type:	`Member function`	
Syntax:	`void ara::rest::StdAllocator< T >::deallocate(value_type *p, std::size_t s)`	
Function param:	p	pointer to allocated memory region
Function param:	s	size of allocated memory region
Return value:	None	
Exceptions:	`noexcept`	
Header file:	ara/rest/allocator.h	
Class:	ara::rest::StdAllocator	
Description:	Deallocate.	

Table 8.238: ara::rest::StdAllocator::deallocate

[SWS_REST_02256] ara::rest::StdAllocator::deallocate ⌈Table 8.238 describes the interface `ara::rest::StdAllocator::deallocate`.⌋ *(RS_CM_00300)*

8.26.7 select_on_container_copy_construction

Service name:	ara::rest::StdAllocator::select_on_container_copy_construction
Type:	`Member function`
Syntax:	`StdAllocator ara::rest::StdAllocator< T >::select_on_container_copy_construction() const`
Function param:	None
Return value:	a value of type `StdAllocator`
Exceptions:	Implementation-defined
Header file:	ara/rest/allocator.h
Class:	ara::rest::StdAllocator
Description:	Returns the allocator to use when a standard container using it is copied. See std::pmr::polymorphic_allocator documentation for details

Table 8.239: ara::rest::StdAllocator::select_on_container_copy_construction

[SWS_REST_02257] ara::rest::StdAllocator::select_on_container_copy_construction ⌈Table 8.239 describes the interface `ara::rest::StdAllocator::select_on_container_co` *(RS_CM_00300)*

8.26.8 resource

Service name:	ara::rest::StdAllocator::resource
Type:	`Member function`
Syntax:	`Allocator* ara::rest::StdAllocator< T >::resource() const`
Function param:	None
Return value:	a value of type `Allocator *`
Exceptions:	`noexcept`

Header file:	ara/rest/allocator.h
Class:	ara::rest::StdAllocator
Description:	Returns the Allocator behind this adapter. See std::pmr::polymorphic_allocator documentation for details

Table 8.240: ara::rest::StdAllocator::resource

[SWS_REST_02258] ara::rest::StdAllocator::resource ⌈Table 8.240 describes the interface `ara::rest::StdAllocator::resource`.⌋*(RS_CM_00300)*

8.27 ara::rest::Uri::Builder

[SWS_REST_02259] ⌈`ara::rest::Uri::Builder` class shall be declared in the `ara/rest/uri.h` header file:

```
1 class ara::rest::Uri::Builder;
```

⌋*(RS_CM_00300, RS_CM_00304)*

8.27.1 Builder

Service name:	ara::rest::Uri::Builder::Builder	
Type:	`Member function`	
Syntax:	`ara::rest::Uri::Builder::Builder(Allocator *alloc=GetDefaultAllocator())`	
Function param:	alloc	an allocator
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Default-constructs a builder.	

Table 8.241: ara::rest::Uri::Builder::Builder

[SWS_REST_02260] ara::rest::Uri::Builder::Builder ⌈Table 8.241 describes the interface `ara::rest::Uri::Builder::Builder`.⌋*(RS_CM_00300, RS_CM_00304)*

8.27.2 Builder

Service name:	ara::rest::Uri::Builder::Builder	
Type:	`Member function`	
Syntax:	`ara::rest::Uri::Builder::Builder(StringView uri, Allocator *alloc=GetDefaultAllocator())`	
Function param:	uri	an URI to initiazlize from

Function param:	alloc	an allocator
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Parses a URI in string format.	

Table 8.242: ara::rest::Uri::Builder::Builder

[SWS_REST_02261] **ara::rest::Uri::Builder::Builder** ⌈Table 8.242 describes the interface `ara::rest::Uri::Builder::Builder`.⌋*(RS_CM_00300, RS_CM_00304)*

8.27.3 Builder

Service name:	ara::rest::Uri::Builder::Builder	
Type:	`Member function`	
Syntax:	`ara::rest::Uri::Builder::Builder(const Uri &uri, Allocator *alloc=GetDefaultAllocator())`	
Function param:	uri	an URI to initiazlize from
Function param:	alloc	an allocator
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Initializes this builder with an existing Uri.	

Table 8.243: ara::rest::Uri::Builder::Builder

[SWS_REST_02262] **ara::rest::Uri::Builder::Builder** ⌈Table 8.243 describes the interface `ara::rest::Uri::Builder::Builder`.⌋*(RS_CM_00300, RS_CM_00304)*

8.27.4 Builder

Service name:	ara::rest::Uri::Builder::Builder	
Type:	`Member function`	
Syntax:	`ara::rest::Uri::Builder::Builder(Uri &&uri, Allocator *alloc=GetDefaultAllocator())`	
Function param:	uri	an URI to initiazlize from
Function param:	alloc	an allocator
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Initializes this builder with an existing Uri.	

Table 8.244: ara::rest::Uri::Builder::Builder

[SWS_REST_02263] **ara::rest::Uri::Builder::Builder** ⌈Table 8.244 describes the interface `ara::rest::Uri::Builder::Builder`.⌋*(RS_CM_00300, RS_CM_00304)*

8.27.5 Scheme

Service name:	ara::rest::Uri::Builder::Scheme	
Type:	`Member function`	
Syntax:	`template <typename T >` `Builder& ara::rest::Uri::Builder::Scheme(T &&value)`	
Function param:	value	a value of an output-streamable type
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Set scheme by parsing the given argument Throws std::invalid_argument if parsing fails.	

Table 8.245: ara::rest::Uri::Builder::Scheme

[SWS_REST_02264] **ara::rest::Uri::Builder::Scheme** ⌈Table 8.245 describes the interface `ara::rest::Uri::Builder::Scheme`.⌋*(RS_CM_00300, RS_CM_00304)*

8.27.6 UserInfo

Service name:	ara::rest::Uri::Builder::UserInfo	
Type:	`Member function`	
Syntax:	`template <typename T >` `Builder& ara::rest::Uri::Builder::UserInfo(T &&value)`	
Function param:	value	a value of an output-streamable type
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Set user info by parsing the given argument Throws std::invalid_argument if parsing fails.	

Table 8.246: ara::rest::Uri::Builder::UserInfo

[SWS_REST_02265] **ara::rest::Uri::Builder::UserInfo** ⌈Table 8.246 describes the interface `ara::rest::Uri::Builder::UserInfo`.⌋*(RS_CM_00300, RS_CM_00304)*

8.27.7 Host

Service name:	ara::rest::Uri::Builder::Host

Type:	`Member function`	
Syntax:	`template <typename T > Builder& ara::rest::Uri::Builder::Host(T &&value)`	
Function param:	value	a value of an output-streamable type
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Sets host by parsing the given argument Throws std::invalid_argument if parsing fails.	

Table 8.247: ara::rest::Uri::Builder::Host

[SWS_REST_02266] ara::rest::Uri::Builder::Host ⌈Table 8.247 describes the interface `ara::rest::Uri::Builder::Host`.⌋*(RS_CM_00300, RS_CM_00304)*

8.27.8 Port

Service name:	ara::rest::Uri::Builder::Port	
Type:	`Member function`	
Syntax:	`template <typename T > Builder& ara::rest::Uri::Builder::Port(T &&value)`	
Function param:	value	a value of an output-streamable type
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Sets the the Uri port from the given argument Throws std::invalid_argument if parsing fails.	

Table 8.248: ara::rest::Uri::Builder::Port

[SWS_REST_02267] ara::rest::Uri::Builder::Port ⌈Table 8.248 describes the interface `ara::rest::Uri::Builder::Port`.⌋*(RS_CM_00300, RS_CM_00304)*

8.27.9 Path

Service name:	ara::rest::Uri::Builder::Path	
Type:	`Member function`	
Syntax:	`Builder& ara::rest::Uri::Builder::Path(StringView value)`	
Function param:	value	a value of an output-streamable type
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Sets the URI path by parsing the given argument Throws std::invalid_argument, if parsing fails.	

Table 8.249: ara::rest::Uri::Builder::Path

[SWS_REST_02268] ara::rest::Uri::Builder::Path ⌈Table 8.249 describes the interface `ara::rest::Uri::Builder::Path`.⌋*(RS_CM_00300, RS_CM_00304)*

8.27.10 Path

Service name:	ara::rest::Uri::Builder::Path	
Type:	`Member function`	
Syntax:	`Builder& ara::rest::Uri::Builder::Path(const Uri::Path &value)`	
Function param:	value	a value of an output-streamable type
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Sets a path from an existing Uri::Path components Throws std::invalid_argument if parsing fails.	

Table 8.250: ara::rest::Uri::Builder::Path

[SWS_REST_02269] ara::rest::Uri::Builder::Path ⌈Table 8.250 describes the interface `ara::rest::Uri::Builder::Path`.⌋*(RS_CM_00300, RS_CM_00304)*

8.27.11 PathSegments

Service name:	ara::rest::Uri::Builder::PathSegments	
Type:	`Member function`	
Syntax:	`template <typename... Ts> Builder& ara::rest::Uri::Builder::PathSegments(Ts &&...values)`	
Function param:	values	values of output-streamable types
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Constructs a path from the given function arguments Throws std::invalid_argument if parsing fails.	

Table 8.251: ara::rest::Uri::Builder::PathSegments

[SWS_REST_02270] ara::rest::Uri::Builder::PathSegments ⌈Table 8.251 describes the interface `ara::rest::Uri::Builder::PathSegments`.⌋*(RS_CM_00300, RS_CM_00304)*

8.27.12 PathSegmentsFrom

Service name:	ara::rest::Uri::Builder::PathSegmentsFrom	
Type:	`Member function`	
Syntax:	`template <typename... Ts> Builder& ara::rest::Uri::Builder::PathSegmentsFrom(std::size_t pos, Ts &&...values)`	
Function param:	pos	index to start from
Function param:	values	values of output-streamable type
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Constructs a path from the given function arguments starting from the n'th path segment.	

Table 8.252: ara::rest::Uri::Builder::PathSegmentsFrom

[SWS_REST_02271] ara::rest::Uri::Builder::PathSegmentsFrom ⌈Table 8.252 describes the interface `ara::rest::Uri::Builder::PathSegmentsFrom`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.27.13 Query

Service name:	ara::rest::Uri::Builder::Query	
Type:	`Member function`	
Syntax:	`Builder& ara::rest::Uri::Builder::Query(const Uri::Query &q)`	
Function param:	q	a query
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Needs documentation.	

Table 8.253: ara::rest::Uri::Builder::Query

[SWS_REST_02272] ara::rest::Uri::Builder::Query ⌈Table 8.253 describes the interface `ara::rest::Uri::Builder::Query`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.27.14 QueryParameter

Service name:	ara::rest::Uri::Builder::QueryParameter
Type:	`Member function`
Syntax:	`template <typename T > Builder& ara::rest::Uri::Builder::QueryParameter(T &&key)`

Function param:	key	of a query parameter
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Inserts a query parameter (key only) If the given key already exists, no action is performed.	

Table 8.254: ara::rest::Uri::Builder::QueryParameter

[SWS_REST_02273] **ara::rest::Uri::Builder::QueryParameter** ⌈Table 8.254 describes the interface `ara::rest::Uri::Builder::QueryParameter`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.27.15 QueryParameter

Service name:	ara::rest::Uri::Builder::QueryParameter	
Type:	`Member function`	
Syntax:	`template <typename T , typename U > Builder& ara::rest::Uri::Builder::QueryParameter(T &&key, U &&value)`	
Function param:	key	a key
Function param:	value	a value
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Inserts a query parameter (key and value) If the given key already exists, no action is performed.	

Table 8.255: ara::rest::Uri::Builder::QueryParameter

[SWS_REST_02274] **ara::rest::Uri::Builder::QueryParameter** ⌈Table 8.255 describes the interface `ara::rest::Uri::Builder::QueryParameter`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.27.16 QueryParameterAt

Service name:	ara::rest::Uri::Builder::QueryParameterAt	
Type:	`Member function`	
Syntax:	`template <typename T , typename U > Builder& ara::rest::Uri::Builder::QueryParameterAt(T &&oldkey, U &&newkey)`	
Function param:	oldkey	a key
Function param:	newkey	a value
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	

Header file:	ara/rest/uri.h
Class:	ara::rest::Uri::Builder
Description:	Replaces an existing paramater (key only) If old exists, then it is replaced by new. Otherwise no action is performed. If the existing key is part of a key/value pair, the entire pair is replaced.

Table 8.256: ara::rest::Uri::Builder::QueryParameterAt

[SWS_REST_02275] **ara::rest::Uri::Builder::QueryParameterAt** ⌈Table 8.256 describes the interface `ara::rest::Uri::Builder::QueryParameterAt`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.27.17 QueryParameterAt

Service name:	ara::rest::Uri::Builder::QueryParameterAt	
Type:	`Member function`	
Syntax:	`template <typename T > Builder& ara::rest::Uri::Builder::QueryParameterAt(T &&key)`	
Function param:	key	a key
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Removes a query parameter (by key) If key exists, removes it. Otherwise no action is performed. If key belong to a key/value pair, the pair is removed. Throws std::invalid_argument if parsing fails.	

Table 8.257: ara::rest::Uri::Builder::QueryParameterAt

[SWS_REST_02276] **ara::rest::Uri::Builder::QueryParameterAt** ⌈Table 8.257 describes the interface `ara::rest::Uri::Builder::QueryParameterAt`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.27.18 QueryParameterAt

Service name:	ara::rest::Uri::Builder::QueryParameterAt	
Type:	`Member function`	
Syntax:	`template <typename T , typename U , typename V > Builder& ara::rest::Uri::Builder::QueryParameterAt(T &&oldkey, U &&newkey, V &&newvalue)`	
Function param:	oldkey	a key
Function param:	newkey	a value
Function param:	newvalue	a value
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	

Class:	ara::rest::Uri::Builder
Description:	Replaces an existing paramater (key + value) If oldkey exists, then it is replaced (including its value) by newkey and newvalue. Otherwise no action is performed. If no old value exists, it is set.

Table 8.258: ara::rest::Uri::Builder::QueryParameterAt

[SWS_REST_02277] ara::rest::Uri::Builder::QueryParameterAt ⌈Table 8.258 describes the interface `ara::rest::Uri::Builder::QueryParameterAt`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.27.19 Fragment

Service name:	ara::rest::Uri::Builder::Fragment	
Type:	`Member function`	
Syntax:	`template <typename T >` `Builder& ara::rest::Uri::Builder::Fragment(T &&value)`	
Function param:	value	a value of an output-streamable type
Return value:	a reference to this builder	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Builder	
Description:	Sets the fragment component of a URI Throws std::invalid_argument if parsing fails.	

Table 8.259: ara::rest::Uri::Builder::Fragment

[SWS_REST_02278] ara::rest::Uri::Builder::Fragment ⌈Table 8.259 describes the interface `ara::rest::Uri::Builder::Fragment`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.27.20 Fragment

Service name:	ara::rest::Uri::Builder::Fragment
Type:	`Member function`
Syntax:	`Builder& ara::rest::Uri::Builder::Fragment()`
Function param:	None
Return value:	a reference to this builder
Exceptions:	Implementation-defined
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri::Builder
Description:	Clears the fragment component.

Table 8.260: ara::rest::Uri::Builder::Fragment

[SWS_REST_02279] ara::rest::Uri::Builder::Fragment ⌈Table 8.260 describes the interface `ara::rest::Uri::Builder::Fragment`.⌋*(RS_CM_00300, RS_CM_00304)*

8.27.21 ToUri

Service name:	ara::rest::Uri::Builder::ToUri
Type:	`Member function`
Syntax:	`Uri ara::rest::Uri::Builder::ToUri() const`
Function param:	None
Return value:	a value of type `Uri`
Exceptions:	Implementation-defined
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri::Builder
Description:	Returns a Uri.

Table 8.261: ara::rest::Uri::Builder::ToUri

[SWS_REST_02280] ara::rest::Uri::Builder::ToUri ⌈Table 8.261 describes the interface `ara::rest::Uri::Builder::ToUri`.⌋*(RS_CM_00300, RS_CM_00304)*

8.28 ara::rest::Uri::Path::Segment

[SWS_REST_02281] ⌈`ara::rest::Uri::Path::Segment` class shall be declared in the `ara/rest/uri.h` header file:

```
1 class ara::rest::Uri::Path::Segment;
```

⌋*(RS_CM_00300, RS_CM_00304)*

8.28.1 Get

Service name:	ara::rest::Uri::Path::Segment::Get
Type:	`Member function`
Syntax:	`StringView ara::rest::Uri::Path::Segment::Get()`
Function param:	None
Return value:	a string representation of this path segment
Exceptions:	`noexcept(std::is_nothrow_constructible< StringView >::value)`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri::Path::Segment
Description:	Returns a string representation of this path segment. The representation is non-percent-encoded

Table 8.262: ara::rest::Uri::Path::Segment::Get

[SWS_REST_02282] ara::rest::Uri::Path::Segment::Get ⌈Table 8.262 describes the interface `ara::rest::Uri::Path::Segment::Get`.⌋*(RS_CM_00300, RS_CM_00304)*

8.28.2 GetAs

Service name:	ara::rest::Uri::Path::Segment::GetAs	
Type:	`Member function template`	

Syntax:	`template <typename T >`
	`T ara::rest::Uri::Path::Segment::GetAs(T &&def={})`
	`const`	

Function param:	def	a default value
Return value:	an instance of type T that represents this URI component.	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Path::Segment	
Description:	Returns this segment converted to a user-defined type. The conversion result is assigned to the function argument which is subsequently returned. If conversion fails the function argument is returned unchanged. So either form is valid: GetAs<string>(), GetAs(string{my_allocator}), GetAs<string>("conversion failed")	

Table 8.263: ara::rest::Uri::Path::Segment::GetAs

[SWS_REST_02283] ara::rest::Uri::Path::Segment::GetAs ⌈Table 8.263 describes the interface `ara::rest::Uri::Path::Segment::GetAs`.⌋*(RS_CM_00300, RS_CM_00304)*

8.28.3 operator==

Service name:	ara::rest::Uri::Path::Segment::operator==	
Type:	`Non-member function`	
Syntax:	`friend bool operator==(const Segment &a, const Segment &b)`	
Function param:	a	object to compare
Function param:	b	object to compare
Return value:	true if segments compare equal	
Exceptions:	`noexcept`	
Header file:	ara/rest/uri.h	
Namespace:	ara::rest::Uri::Path::Segment	
Description:	Tests two segments for equality.	

Table 8.264: ara::rest::Uri::Path::Segment::operator==

[SWS_REST_02284] ara::rest::Uri::Path::Segment::operator== ⌈Table 8.264 describes the interface `ara::rest::Uri::Path::Segment::operator==`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.28.4 operator!=

Service name:	ara::rest::Uri::Path::Segment::operator!=	
Type:	`Non-member function`	
Syntax:	`friend bool operator!=(const Segment &a, const Segment &b)`	
Function param:	a	object to compare
Function param:	b	object to compare
Return value:	true if segments compare unequal	
Exceptions:	`noexcept`	
Header file:	ara/rest/uri.h	
Namespace:	ara::rest::Uri::Path::Segment	
Description:	Tests two segments for inequality.	

Table 8.265: ara::rest::Uri::Path::Segment::operator!=

[SWS_REST_02285] ara::rest::Uri::Path::Segment::operator!= ⌈Table 8.265 describes the interface `ara::rest::Uri::Path::Segment::operator!=`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.28.5 operator<

Service name:	ara::rest::Uri::Path::Segment::operator<	
Type:	`Non-member function`	
Syntax:	`friend bool operator<(const Segment &a, const Segment &b)`	
Function param:	a	object to compare
Function param:	b	object to compare
Return value:	true if a compares less-than b	
Exceptions:	`noexcept`	
Header file:	ara/rest/uri.h	
Namespace:	ara::rest::Uri::Path::Segment	
Description:	Compares two path segments according to their lexicographical order.	

Table 8.266: ara::rest::Uri::Path::Segment::operator<

[SWS_REST_02286] ara::rest::Uri::Path::Segment::operator< ⌈Table 8.266 describes the interface `ara::rest::Uri::Path::Segment::operator<`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.29 ara::rest::Uri::Path

[SWS_REST_02287] ⌈`ara::rest::Uri::Path` class shall be declared in the `ara/rest/uri.h` header file:

```
1 class ara::rest::Uri::Path;
```

⌋*(RS_CM_00300, RS_CM_00304)*

8.29.1 IteratorRange

Name:	`IteratorRange`
Type:	`Member type alias`
Syntax:	`using ara::rest::Uri::Path::IteratorRange = ara::rest::IteratorRange<unspecified_iterator_type>`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri::Path
Description:	Iterator range of path segments.

Table 8.267: ara::rest::Uri::Path::IteratorRange

[SWS_REST_02288] IteratorRange ⌈Table 8.267 describes the type alias `ara::rest::Uri::Path::IteratorRange`.⌋*(RS_CM_00300, RS_CM_00304)*

8.29.2 NumSegments

Service name:	ara::rest::Uri::Path::NumSegments
Type:	`Member function`
Syntax:	`std::size_t ara::rest::Uri::Path::NumSegments() const`
Function param:	None
Return value:	a number of path segmeents
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri::Path
Description:	Returns the number of path segments.

Table 8.268: ara::rest::Uri::Path::NumSegments

[SWS_REST_02289] ara::rest::Uri::Path::NumSegments ⌈Table 8.268 describes the interface `ara::rest::Uri::Path::NumSegments`.⌋*(RS_CM_00300, RS_CM_00304)*

8.29.3 GetSegments

Service name:	ara::rest::Uri::Path::GetSegments
Type:	`Member function`
Syntax:	`IteratorRange ara::rest::Uri::Path::GetSegments() const`
Function param:	None
Return value:	an iterator range of path segments
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri::Path
Description:	Returns a range of path segments.

Table 8.269: ara::rest::Uri::Path::GetSegments

[SWS_REST_02290] ara::rest::Uri::Path::GetSegments ⌈Table 8.269 describes the interface `ara::rest::Uri::Path::GetSegments`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.29.4 operator==

Service name:	ara::rest::Uri::Path::operator==	
Type:	`Non-member function`	
Syntax:	`friend bool operator==(const Path &a, const Path &b)`	
Function param:	a	object to compare
Function param:	b	object to compare
Return value:	true if equal	
Exceptions:	`noexcept`	
Header file:	ara/rest/uri.h	
Namespace:	ara::rest::Uri::Path	
Description:	Tests two paths for equality.	

Table 8.270: ara::rest::Uri::Path::operator==

[SWS_REST_02291] ara::rest::Uri::Path::operator== ⌈Table 8.270 describes the interface `ara::rest::Uri::Path::operator==`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.29.5 operator!=

Service name:	ara::rest::Uri::Path::operator!=	
Type:	`Non-member function`	
Syntax:	`friend bool operator!=(const Path &a, const Path &b)`	
Function param:	a	object to compare
Function param:	b	object to compare
Return value:	true if not equal	
Exceptions:	`noexcept`	
Header file:	ara/rest/uri.h	
Namespace:	ara::rest::Uri::Path	
Description:	Tests two paths for inequality.	

Table 8.271: ara::rest::Uri::Path::operator!=

[SWS_REST_02292] ara::rest::Uri::Path::operator!= ⌈Table 8.271 describes the interface `ara::rest::Uri::Path::operator!=`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.29.6 operator<

Service name:	ara::rest::Uri::Path::operator<

Type:	`Non-member function`	
Syntax:	`friend bool operator<(const Path &a, const Path &b)`	
Function param:	a	object to compare
Function param:	b	object to compare
Return value:	true if a compares less-than b	
Exceptions:	`noexcept`	
Header file:	ara/rest/uri.h	
Namespace:	ara::rest::Uri::Path	
Description:	Relates two paths according to their lexicographical order.	

Table 8.272: ara::rest::Uri::Path::operator<

[SWS_REST_02293] ara::rest::Uri::Path::operator< ⌈Table 8.272 describes the interface `ara::rest::Uri::Path::operator<`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.30 ara::rest::Uri::Query::Parameter

[SWS_REST_02294] ⌈`ara::rest::Uri::Query::Parameter` class shall be declared in the `ara/rest/uri.h` header file:

```
1 class ara::rest::Uri::Query::Parameter;
```

⌋ *(RS_CM_00300, RS_CM_00304)*

8.30.1 GetKey

Service name:	ara::rest::Uri::Query::Parameter::GetKey
Type:	`Member function`
Syntax:	`StringView ara::rest::Uri::Query::Parameter::GetKey() const`
Function param:	None
Return value:	a string representation
Exceptions:	`noexcept(std::is_nothrow_constructible< StringView >::value)`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri::Query::Parameter
Description:	Returns a string representation of the parameter key The representation is non-percent-encoded.

Table 8.273: ara::rest::Uri::Query::Parameter::GetKey

[SWS_REST_02295] ara::rest::Uri::Query::Parameter::GetKey ⌈Table 8.273 describes the interface `ara::rest::Uri::Query::Parameter::GetKey`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.30.2 GetKeyAs

Service name:	ara::rest::Uri::Query::Parameter::GetKeyAs
Type:	`Member function template`
Syntax:	`template <typename T > T ara::rest::Uri::Query::Parameter::GetKeyAs(T &&def={}) const`
Function param:	def — a default value
Return value:	an instance of type T that represents this URI component.
Exceptions:	Implementation-defined
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri::Query::Parameter
Description:	Converts a query parameter key to the specified type. The conversion result is assigned to the function argument which is subsequently returned. If conversion fails the function argument is returned unchanged. So either form is valid: GetAs<string>(), GetAs(string{my_allocator}), GetAs<string>("conversion failed")

Table 8.274: ara::rest::Uri::Query::Parameter::GetKeyAs

[SWS_REST_02296] ara::rest::Uri::Query::Parameter::GetKeyAs ⌈Table 8.274 describes the interface `ara::rest::Uri::Query::Parameter::GetKeyAs`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.30.3 HasValue

Service name:	ara::rest::Uri::Query::Parameter::HasValue
Type:	`Member function`
Syntax:	`bool ara::rest::Uri::Query::Parameter::HasValue() const`
Function param:	None
Return value:	true if this query paramater has a value component
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri::Query::Parameter
Description:	Needs documentation.

Table 8.275: ara::rest::Uri::Query::Parameter::HasValue

[SWS_REST_02297] ara::rest::Uri::Query::Parameter::HasValue ⌈Table 8.275 describes the interface `ara::rest::Uri::Query::Parameter::HasValue`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.30.4 GetValue

Service name:	ara::rest::Uri::Query::Parameter::GetValue
Type:	`Member function`

Syntax:	`StringView ara::rest::Uri::Query::Parameter::GetValue() const`
Function param:	None
Return value:	a string representation of the value
Exceptions:	`noexcept(std::is_nothrow_constructible< StringView >::value)`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri::Query::Parameter
Description:	Obtains the value of a query parameter If none exists the result is undefined.

Table 8.276: ara::rest::Uri::Query::Parameter::GetValue

[SWS_REST_02298] **ara::rest::Uri::Query::Parameter::GetValue** ⌈Table 8.276 describes the interface `ara::rest::Uri::Query::Parameter::GetValue`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.30.5 GetValueAs

Service name:	ara::rest::Uri::Query::Parameter::GetValueAs	
Type:	`Member function template`	
Syntax:	`template <typename T > T ara::rest::Uri::Query::Parameter::GetValueAs(T &&def={}) const`	
Function param:	def	a default value
Return value:	an instance of type T that represents this URI component.	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Query::Parameter	
Description:	Converts a query parameter value to the specified type. The conversion result is assigned to the function argument which is subsequently returned. If conversion fails the function argument is returned unchanged. So either form is valid: GetAs<string>(), GetAs(string{my_allocator}), GetAs<string>("conversion failed")	

Table 8.277: ara::rest::Uri::Query::Parameter::GetValueAs

[SWS_REST_02299] **ara::rest::Uri::Query::Parameter::GetValueAs** ⌈Table 8.277 describes the interface `ara::rest::Uri::Query::Parameter::GetValueAs`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.31 ara::rest::Uri::Query

[SWS_REST_02300] ⌈`ara::rest::Uri::Query` class shall be declared in the `ara/rest/uri.h` header file:

```
1 class ara::rest::Uri::Query;
```

⌋(*RS_CM_00300*, *RS_CM_00304*)

8.31.1 IteratorRange

Name:	`IteratorRange`
Type:	`Member type alias`
Syntax:	`using ara::rest::Uri::Query::IteratorRange = ara::rest::IteratorRange<unspecified_iterator_type>`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri::Query
Description:	A range of query parameters.

Table 8.278: ara::rest::Uri::Query::IteratorRange

[SWS_REST_02301] IteratorRange ⌈Table 8.278 describes the type alias `ara::rest::Uri::Query::IteratorRange`.⌋(*RS_CM_00300*, *RS_CM_00304*)

8.31.2 NumParameters

Service name:	ara::rest::Uri::Query::NumParameters
Type:	`Member function`
Syntax:	`std::size_t ara::rest::Uri::Query::NumParameters() const`
Function param:	None
Return value:	the number of query parameters
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri::Query
Description:	Returns the number of query parameters.

Table 8.279: ara::rest::Uri::Query::NumParameters

[SWS_REST_02302] ara::rest::Uri::Query::NumParameters ⌈Table 8.279 describes the interface `ara::rest::Uri::Query::NumParameters`.⌋(*RS_CM_00300*, *RS_CM_00304*)

8.31.3 GetParameters

Service name:	ara::rest::Uri::Query::GetParameters
Type:	`Member function`
Syntax:	`IteratorRange ara::rest::Uri::Query::GetParameters() const`
Function param:	None
Return value:	an iterator range of query parameters
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h

Class:	ara::rest::Uri::Query
Description:	Returns the range of all query parameters.

Table 8.280: ara::rest::Uri::Query::GetParameters

[SWS_REST_02303] ara::rest::Uri::Query::GetParameters ⌈Table 8.280 describes the interface `ara::rest::Uri::Query::GetParameters`.⌋*(RS_CM_00300, RS_CM_00304)*

8.31.4 GetParameter

Service name:	ara::rest::Uri::Query::GetParameter	
Type:	`Member function`	
Syntax:	`const Parameter& ara::rest::Uri::Query::GetParameter(std::size_t i) const`	
Function param:	i	an index
Return value:	a reference to the query parameter	
Exceptions:	`noexcept`	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Query	
Description:	Returns a specific query parameter by index.	

Table 8.281: ara::rest::Uri::Query::GetParameter

[SWS_REST_02304] ara::rest::Uri::Query::GetParameter ⌈Table 8.281 describes the interface `ara::rest::Uri::Query::GetParameter`.⌋*(RS_CM_00300, RS_CM_00304)*

8.31.5 Find

Service name:	ara::rest::Uri::Query::Find	
Type:	`Member function`	
Syntax:	`IteratorRange::Iterator ara::rest::Uri::Query::Find(StringView key) const`	
Function param:	key	a key
Return value:	an iterator to the respective query parameter	
Exceptions:	`noexcept`	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Query	
Description:	Searches for a query parameter by key.	

Table 8.282: ara::rest::Uri::Query::Find

[SWS_REST_02305] ara::rest::Uri::Query::Find ⌈Table 8.282 describes the interface `ara::rest::Uri::Query::Find`.⌋*(RS_CM_00300, RS_CM_00304)*

8.31.6 HasKey

Service name:	ara::rest::Uri::Query::HasKey	
Type:	`Member function`	
Syntax:	`bool ara::rest::Uri::Query::HasKey(StringView key)`	
Function param:	key	a key
Return value:	true of key exists	
Exceptions:	`noexcept`	
Header file:	ara/rest/uri.h	
Class:	ara::rest::Uri::Query	
Description:	Tests whether a query parameter of a given key exists.	

Table 8.283: ara::rest::Uri::Query::HasKey

[SWS_REST_02306] ara::rest::Uri::Query::HasKey ⌈Table 8.283 describes the interface `ara::rest::Uri::Query::HasKey`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32 ara::rest::Uri

[SWS_REST_02307] ⌈`ara::rest::Uri` class shall be declared in the `ara/rest/uri.h` header file:

```
1 class ara::rest::Uri;
```

⌋*(RS_CM_00300, RS_CM_00304)*

8.32.1 Part

Name:	`Part`	
Type:	`Member enumeration`	
Range:	`Scheme`	`= 1 « 1`
	`UserInfo`	`= 1 « 2`
	`Host`	`= 1 « 3`
	`Port`	`= 1 « 4`
	`Path`	`= 1 « 5`
	`Query`	`= 1 « 6`
	`Fragment`	`= 1 « 7`
	`PathAndQuery`	`= Part::Path | Part::Query`
	`PathEtc`	`= Part::Path | Part::Query | Part::Fragment`
	`All`	`= ~std::underlying_type<Part>::type{0}`

Syntax:	`enum class Part : std::uint32_t { Scheme = 1 « 1, UserInfo = 1 « 2, Host = 1 « 3, Port = 1 « 4, Path = 1 « 5, Query = 1 « 6, Fragment = 1 « 7, PathAndQuery = Part::Path |Part::Query, PathEtc = Part::Path |Part::Query |Part::Fragment, All = ~std::underlying_type<Part>::type{0} };`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	Used to specify a subset of a URI. Part defines components of a

Table 8.284: ara::rest::Uri::Part

[SWS_REST_02308] Part ⌈Table 8.284 describes the enumeration datatype `ara::rest::Uri::Part`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.2 LENGTH_MAX

Name:	`LENGTH_MAX`
Type:	`Member variable`
Syntax:	`static constexpr std::size_t LENGTH_MAX = 2048;`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	The maximum length of a URI. The suggested length maximum is around 2000 characters which roughly conforms to the typical limit that mainstream webbrowsers support. A bound is specified to enable optimization potential in the internal encoding.

Table 8.285: ara::rest::Uri::LENGTH_MAX

[SWS_REST_02309] ara::rest::Uri::LENGTH_MAX ⌈Table 8.285 describes the variable `ara::rest::Uri::LENGTH_MAX`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.3 operator|

Service name:	ara::rest::Uri::operator|	
Type:	`Non-member function`	
Syntax:	`friend constexpr Part operator|(Part a, Part b)`	
Function param:	a	(set of) Part enumerator(s)
Function param:	b	(set of) Part enumerator(s)
Return value:	a set of Part enumerators	
Exceptions:	`noexcept`	
Header file:	ara/rest/uri.h	

Namespace:	ara::rest::Uri
Description:	Computes a set of Part enumerators.

Table 8.286: ara::rest::Uri::operator|

[SWS_REST_02310] ara::rest::Uri::operator| ⌈Table 8.286 describes the interface `ara::rest::Uri::operator|`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.4 Uri

Service name:	ara::rest::Uri::Uri
Type:	`Member function`
Syntax:	`ara::rest::Uri::Uri() =default`
Function param:	None
Return value:	None
Exceptions:	`noexcept=default`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	Constructs a default URI. A default-constructed URI is empty. To populate an existing URI, Uri::Builder must be used. Uri member functions must not throw unless in those cases where StringView is used and StringView is not 'nothrow_constructible'.

Table 8.287: ara::rest::Uri::Uri

[SWS_REST_02311] ara::rest::Uri::Uri ⌈Table 8.287 describes the interface `ara::rest::Uri::Uri`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.5 HasScheme

Service name:	ara::rest::Uri::HasScheme
Type:	`Member function`
Syntax:	`bool ara::rest::Uri::HasScheme() const`
Function param:	None
Return value:	a value of type `bool`
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	Has scheme.

Table 8.288: ara::rest::Uri::HasScheme

[SWS_REST_02312] ara::rest::Uri::HasScheme ⌈Table 8.288 describes the interface `ara::rest::Uri::HasScheme`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.6 GetScheme

Service name:	ara::rest::Uri::GetScheme
Type:	`Member function`
Syntax:	`StringView ara::rest::Uri::GetScheme() const`
Function param:	None
Return value:	a value of type `StringView`
Exceptions:	`noexcept(std::is_nothrow_constructible< StringView >::value)`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	gets scheme.

Table 8.289: ara::rest::Uri::GetScheme

[SWS_REST_02313] ara::rest::Uri::GetScheme ⌈Table 8.289 describes the interface `ara::rest::Uri::GetScheme`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.32.7 HasUserInfo

Service name:	ara::rest::Uri::HasUserInfo
Type:	`Member function`
Syntax:	`bool ara::rest::Uri::HasUserInfo() const`
Function param:	None
Return value:	a value of type `bool`
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	has user info.

Table 8.290: ara::rest::Uri::HasUserInfo

[SWS_REST_02314] ara::rest::Uri::HasUserInfo ⌈Table 8.290 describes the interface `ara::rest::Uri::HasUserInfo`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.32.8 GetUserinfo

Service name:	ara::rest::Uri::GetUserinfo
Type:	`Member function`
Syntax:	`StringView ara::rest::Uri::GetUserinfo() const`
Function param:	None
Return value:	a value of type `StringView`
Exceptions:	`noexcept(std::is_nothrow_constructible< StringView >::value)`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	get user info.

Table 8.291: ara::rest::Uri::GetUserinfo

[SWS_REST_02315] ara::rest::Uri::GetUserinfo ⌈Table 8.291 describes the interface `ara::rest::Uri::GetUserinfo`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.9 HasHost

Service name:	ara::rest::Uri::HasHost
Type:	`Member function`
Syntax:	`bool ara::rest::Uri::HasHost() const`
Function param:	None
Return value:	a value of type `bool`
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	has host.

Table 8.292: ara::rest::Uri::HasHost

[SWS_REST_02316] ara::rest::Uri::HasHost ⌈Table 8.292 describes the interface `ara::rest::Uri::HasHost`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.10 GetHost

Service name:	ara::rest::Uri::GetHost
Type:	`Member function`
Syntax:	`StringView ara::rest::Uri::GetHost() const`
Function param:	None
Return value:	a value of type `StringView`
Exceptions:	`noexcept(std::is_nothrow_constructible< StringView >::value)`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	get host.

Table 8.293: ara::rest::Uri::GetHost

[SWS_REST_02317] ara::rest::Uri::GetHost ⌈Table 8.293 describes the interface `ara::rest::Uri::GetHost`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.11 HasPort

Service name:	ara::rest::Uri::HasPort
Type:	`Member function`
Syntax:	`bool ara::rest::Uri::HasPort() const`
Function param:	None
Return value:	a value of type `bool`
Exceptions:	`noexcept`

Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	has host.

Table 8.294: ara::rest::Uri::HasPort

[SWS_REST_02318] ara::rest::Uri::HasPort ⌈Table 8.294 describes the interface `ara::rest::Uri::HasPort`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.12 GetPort

Service name:	ara::rest::Uri::GetPort
Type:	`Member function`
Syntax:	`int ara::rest::Uri::GetPort() const`
Function param:	None
Return value:	a value of type `int`
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	get port.

Table 8.295: ara::rest::Uri::GetPort

[SWS_REST_02319] ara::rest::Uri::GetPort ⌈Table 8.295 describes the interface `ara::rest::Uri::GetPort`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.13 HasPath

Service name:	ara::rest::Uri::HasPath
Type:	`Member function`
Syntax:	`bool ara::rest::Uri::HasPath() const`
Function param:	None
Return value:	a value of type `bool`
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	has path.

Table 8.296: ara::rest::Uri::HasPath

[SWS_REST_02320] ara::rest::Uri::HasPath ⌈Table 8.296 describes the interface `ara::rest::Uri::HasPath`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.14 GetPath

Service name:	ara::rest::Uri::GetPath
Type:	`Member function`
Syntax:	`const Path& ara::rest::Uri::GetPath() const`
Function param:	None
Return value:	a value of type `const Path &`
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	get path.

Table 8.297: ara::rest::Uri::GetPath

[SWS_REST_02321] ara::rest::Uri::GetPath ⌈Table 8.297 describes the interface `ara::rest::Uri::GetPath`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.15 HasQuery

Service name:	ara::rest::Uri::HasQuery
Type:	`Member function`
Syntax:	`bool ara::rest::Uri::HasQuery() const`
Function param:	None
Return value:	a value of type `bool`
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	has query.

Table 8.298: ara::rest::Uri::HasQuery

[SWS_REST_02322] ara::rest::Uri::HasQuery ⌈Table 8.298 describes the interface `ara::rest::Uri::HasQuery`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.16 GetQuery

Service name:	ara::rest::Uri::GetQuery
Type:	`Member function`
Syntax:	`const Query& ara::rest::Uri::GetQuery() const`
Function param:	None
Return value:	a value of type `const Query &`
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	get query.

Table 8.299: ara::rest::Uri::GetQuery

[SWS_REST_02323] ara::rest::Uri::GetQuery ⌈Table 8.299 describes the interface `ara::rest::Uri::GetQuery`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.17 HasFragment

Service name:	ara::rest::Uri::HasFragment
Type:	`Member function`
Syntax:	`bool ara::rest::Uri::HasFragment() const`
Function param:	None
Return value:	a value of type `bool`
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	Has Fragment.

Table 8.300: ara::rest::Uri::HasFragment

[SWS_REST_02324] ara::rest::Uri::HasFragment ⌈Table 8.300 describes the interface `ara::rest::Uri::HasFragment`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.18 GetFragment

Service name:	ara::rest::Uri::GetFragment
Type:	`Member function`
Syntax:	`StringView ara::rest::Uri::GetFragment() const`
Function param:	None
Return value:	a value of type `StringView`
Exceptions:	`noexcept(std::is_nothrow_constructible< StringView >::value)`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	Get Fragment as string.

Table 8.301: ara::rest::Uri::GetFragment

[SWS_REST_02325] ara::rest::Uri::GetFragment ⌈Table 8.301 describes the interface `ara::rest::Uri::GetFragment`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.19 GetFragmentAs

Service name:	ara::rest::Uri::GetFragmentAs	
Type:	`Member function template`	
Syntax:	`template <typename T > T ara::rest::Uri::GetFragmentAs(T &&def={}) const`	
Function param:	def	a default value
Return value:	an instance of type T that represents this URI component	

Exceptions:	Implementation-defined
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	Converts a URI fragment part to a given type. The conversion result is assigned to the function argument which is subsequently returned. If conversion fails the function argument is returned unchanged. So either form is valid: GetFragmentAs<string>(), GetFragmentAs(string{my_allocator}), GetFragmentAs<string>("conversion failed")

Table 8.302: ara::rest::Uri::GetFragmentAs

[SWS_REST_02326] ara::rest::Uri::GetFragmentAs ⌈Table 8.302 describes the interface `ara::rest::Uri::GetFragmentAs`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.20 IsEmpty

Service name:	ara::rest::Uri::IsEmpty
Type:	`Member function`
Syntax:	`bool ara::rest::Uri::IsEmpty() const`
Function param:	None
Return value:	true if empty
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	Is URI empty.

Table 8.303: ara::rest::Uri::IsEmpty

[SWS_REST_02327] ara::rest::Uri::IsEmpty ⌈Table 8.303 describes the interface `ara::rest::Uri::IsEmpty`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.21 IsRelative

Service name:	ara::rest::Uri::IsRelative
Type:	`Member function`
Syntax:	`bool ara::rest::Uri::IsRelative() const`
Function param:	None
Return value:	true if relative
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	Is URI relative. An URI is relative if it does not starts with a scheme.

Table 8.304: ara::rest::Uri::IsRelative

[SWS_REST_02328] ara::rest::Uri::IsRelative ⌈Table 8.304 describes the interface `ara::rest::Uri::IsRelative`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.22 isOpaque

Service name:	ara::rest::Uri::isOpaque
Type:	`Member function`
Syntax:	`bool ara::rest::Uri::isOpaque() const`
Function param:	None
Return value:	true if this URI is opaque
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	Denotes whether the URI is opaque. An opaque URI is an absolute URI whose scheme-specific part does not begin with a slash character ('/').

Table 8.305: ara::rest::Uri::isOpaque

[SWS_REST_02329] ara::rest::Uri::isOpaque ⌈Table 8.305 describes the interface `ara::rest::Uri::isOpaque`.⌋*(RS_CM_00300, RS_CM_00304)*

8.32.23 isHierarchical

Service name:	ara::rest::Uri::isHierarchical
Type:	`Member function`
Syntax:	`bool ara::rest::Uri::isHierarchical() const`
Function param:	None
Return value:	true if this URI is hierarchical
Exceptions:	`noexcept`
Header file:	ara/rest/uri.h
Class:	ara::rest::Uri
Description:	Denotes wether this URI is hierarchical. A hierarchical URI is either an absolute URI whose scheme-specific part begins with a slash character, or a relative URI, that is, a URI that does not specify a scheme.

Table 8.306: ara::rest::Uri::isHierarchical

[SWS_REST_02330] ara::rest::Uri::isHierarchical ⌈Table 8.306 describes the interface `ara::rest::Uri::isHierarchical`.⌋*(RS_CM_00300, RS_CM_00304)*

8.33 ara::rest::Uuid

[SWS_REST_02331] ⌈`ara::rest::Uuid` class shall be declared in the `ara/rest/uuid.h` header file:

```
1 class ara::rest::Uuid;
```

⌋(*RS_CM_00300*)

8.33.1 Uuid

Service name:	ara::rest::Uuid::Uuid
Type:	`Member function`
Syntax:	`ara::rest::Uuid::Uuid() =default`
Function param:	None
Return value:	None
Exceptions:	`noexcept=default`
Header file:	ara/rest/uuid.h
Class:	ara::rest::Uuid
Description:	Default constructs a Uuid.

Table 8.307: ara::rest::Uuid::Uuid

[SWS_REST_02332] ara::rest::Uuid::Uuid ⌈Table 8.307 describes the interface `ara::rest::Uuid::Uuid`.⌋(*RS_CM_00300*)

8.33.2 Uuid

Service name:	ara::rest::Uuid::Uuid	
Type:	`Member function`	
Syntax:	`ara::rest::Uuid::Uuid(StringView id)`	
Function param:	id	a UUID in RFC4122 format
Return value:	None	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uuid.h	
Class:	ara::rest::Uuid	
Description:	Constructs a Uuid from a string representation. Throws an std::invalid_argument if parsing fails.	

Table 8.308: ara::rest::Uuid::Uuid

[SWS_REST_02333] ara::rest::Uuid::Uuid ⌈Table 8.308 describes the interface `ara::rest::Uuid::Uuid`.⌋(*RS_CM_00300*)

8.33.3 Uuid

Service name:	ara::rest::Uuid::Uuid	
Type:	`Member function`	
Syntax:	`ara::rest::Uuid::Uuid(std::uint32_t timeLow, std::uint16_t timeMid, std::uint16_t timeHighAndVersion, std::uint16_t clockSeq, std::uint64_t node)`	
Function param:	timeLow	see RFC 4122

Function param:	timeMid	see RFC 4122
Function param:	timeHighAndVersion	see RFC 4122
Function param:	clockSeq	see RFC 4122
Function param:	node	see RFC 4122
Return value:	None	
Exceptions:	`noexcept`	
Header file:	ara/rest/uuid.h	
Class:	ara::rest::Uuid	
Description:	Constructs a Uuid from its components explicitly.	

Table 8.309: ara::rest::Uuid::Uuid

[SWS_REST_02334] ara::rest::Uuid::Uuid ⌈Table 8.309 describes the interface `ara::rest::Uuid::Uuid`.⌋*(RS_CM_00300)*

8.33.4 GetTimeLow

Service name:	ara::rest::Uuid::GetTimeLow
Type:	`Member function`
Syntax:	`std::uint32_t ara::rest::Uuid::GetTimeLow() const`
Function param:	None
Return value:	a numeric value
Exceptions:	`noexcept`
Header file:	ara/rest/uuid.h
Class:	ara::rest::Uuid
Description:	Returns time low.

Table 8.310: ara::rest::Uuid::GetTimeLow

[SWS_REST_02335] ara::rest::Uuid::GetTimeLow ⌈Table 8.310 describes the interface `ara::rest::Uuid::GetTimeLow`.⌋*(RS_CM_00300)*

8.33.5 GetTimeMid

Service name:	ara::rest::Uuid::GetTimeMid
Type:	`Member function`
Syntax:	`std::uint16_t ara::rest::Uuid::GetTimeMid() const`
Function param:	None
Return value:	a numeric value
Exceptions:	`noexcept`
Header file:	ara/rest/uuid.h
Class:	ara::rest::Uuid
Description:	Returns time mid.

Table 8.311: ara::rest::Uuid::GetTimeMid

[SWS_REST_02336] ara::rest::Uuid::GetTimeMid ⌈Table 8.311 describes the interface `ara::rest::Uuid::GetTimeMid`.⌋ *(RS_CM_00300)*

8.33.6 GetTimeHighAndVersion

Service name:	ara::rest::Uuid::GetTimeHighAndVersion
Type:	`Member function`
Syntax:	`std::uint16_t ara::rest::Uuid::GetTimeHighAndVersion() const`
Function param:	None
Return value:	a numeric value
Exceptions:	`noexcept`
Header file:	ara/rest/uuid.h
Class:	ara::rest::Uuid
Description:	Returns time high and version.

Table 8.312: ara::rest::Uuid::GetTimeHighAndVersion

[SWS_REST_02337] ara::rest::Uuid::GetTimeHighAndVersion ⌈Table 8.312 describes the interface `ara::rest::Uuid::GetTimeHighAndVersion`.⌋ *(RS_CM_00300)*

8.33.7 GetClockSeq

Service name:	ara::rest::Uuid::GetClockSeq
Type:	`Member function`
Syntax:	`std::uint16_t ara::rest::Uuid::GetClockSeq() const`
Function param:	None
Return value:	a numeric value
Exceptions:	`noexcept`
Header file:	ara/rest/uuid.h
Class:	ara::rest::Uuid
Description:	Returns clock sequence count.

Table 8.313: ara::rest::Uuid::GetClockSeq

[SWS_REST_02338] ara::rest::Uuid::GetClockSeq ⌈Table 8.313 describes the interface `ara::rest::Uuid::GetClockSeq`.⌋ *(RS_CM_00300)*

8.33.8 GetNode

Service name:	ara::rest::Uuid::GetNode
Type:	`Member function`
Syntax:	`std::uint64_t ara::rest::Uuid::GetNode() const`
Function param:	None
Return value:	return type

Exceptions:	noexcept
Header file:	ara/rest/uuid.h
Class:	ara::rest::Uuid
Description:	Returns node value.

Table 8.314: ara::rest::Uuid::GetNode

[SWS_REST_02339] ara::rest::Uuid::GetNode ⌈Table 8.314 describes the interface ara::rest::Uuid::GetNode.⌋*(RS_CM_00300)*

8.33.9 operator==

Service name:	ara::rest::Uuid::operator==	
Type:	Non-member function	
Syntax:	friend bool operator==(const Uuid &a, const Uuid &b)	
Function param:	a	a uuid
Function param:	b	a uuid
Return value:	true if equal	
Exceptions:	noexcept	
Header file:	ara/rest/uuid.h	
Namespace:	ara::rest::Uuid	
Description:	Compares UUIDs.	

Table 8.315: ara::rest::Uuid::operator==

[SWS_REST_02340] ara::rest::Uuid::operator== ⌈Table 8.315 describes the interface ara::rest::Uuid::operator==.⌋*(RS_CM_00300)*

8.33.10 operator!=

Service name:	ara::rest::Uuid::operator!=	
Type:	Non-member function	
Syntax:	friend bool operator!=(const Uuid &a, const Uuid &b)	
Function param:	a	a uuid
Function param:	b	a uuid
Return value:	true if unequal	
Exceptions:	noexcept	
Header file:	ara/rest/uuid.h	
Namespace:	ara::rest::Uuid	
Description:	Compares UUIDs.	

Table 8.316: ara::rest::Uuid::operator!=

[SWS_REST_02341] ara::rest::Uuid::operator!= ⌈Table 8.316 describes the interface ara::rest::Uuid::operator!=.⌋*(RS_CM_00300)*

8.33.11 operator<

Service name:	ara::rest::Uuid::operator<	
Type:	`Non-member function`	
Syntax:	`friend bool operator<(const Uuid &a, const Uuid &b)`	
Function param:	a	a uuid
Function param:	b	a uuid
Return value:	true if uuid compares less than lexicographically less by component	
Exceptions:	`noexcept`	
Header file:	ara/rest/uuid.h	
Namespace:	ara::rest::Uuid	
Description:	Compares UUIDs.	

Table 8.317: ara::rest::Uuid::operator<

[SWS_REST_02342] ara::rest::Uuid::operator< ⌈Table 8.317 describes the interface `ara::rest::Uuid::operator<`.⌋ *(RS_CM_00300)*

8.34 ara::rest::ogm

8.34.1 Copy

Service name:	Copy	
Type:	`Non-member function`	
Syntax:	`template <typename T > Pointer<T> ara::rest::ogm::Copy(const T &g, Allocator *alloc=GetDefaultAllocator())`	
Function param:	g	OGM graph to copy
Function param:	alloc	allocator to use for the copy
Return value:	a copy	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/copy.h	
Namespace:	ara::rest::ogm	
Description:	Copies an object graph. Performs a deep copy of the argument	

Table 8.318: ara::rest::ogm::Copy

[SWS_REST_02343] Copy ⌈Table 8.318 describes the interface `Copy`.⌋ *(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.34.2 Copy

Service name:	Copy
Type:	`Non-member function`
Syntax:	`template <typename T > Pointer<T> ara::rest::ogm::Copy(const Pointer< T > &g, Allocator *alloc=GetDefaultAllocator())`

Function param:	g	OGM graph to copy
Function param:	alloc	allocator to use for the copy
Return value:	a copy	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/copy.h	
Namespace:	ara::rest::ogm	
Description:	Copies an object graph. Performs a deep copy of the argument	

Table 8.319: ara::rest::ogm::Copy

[SWS_REST_02344] Copy ⌈Table 8.319 describes the interface `Copy`.⌋ *(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.34.3 Visit

Service name:	Visit	
Type:	`Non-member function`	
Syntax:	`template <typename NodeT , typename... Visitors> void ara::rest::ogm::Visit(const NodeT &u, Visitors &&...vis)`	
Function param:	u	OGM node to resolve
Function param:	vis	a set of functors
Return value:	a copy	
Exceptions:	`noexcept`	
Header file:	ara/rest/ogm/visit.h	
Namespace:	ara::rest::ogm	
Description:	Resolves the exact type of the OGM node passed to it. The function accepts a set of functors to call with the exact type of the graph node argument. Overload resolution shall apply here. If no visitor matches, the function silently returns without calling any visitor.	

Table 8.320: ara::rest::ogm::Visit

[SWS_REST_02345] Visit ⌈Table 8.320 describes the interface `Visit`.⌋ *(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.34.4 Visit

Service name:	Visit	
Type:	`Non-member function`	
Syntax:	`template <typename NodeT , typename... Visitors> void ara::rest::ogm::Visit(const Pointer< NodeT > &u, Visitors &&...vis)`	
Function param:	u	OGM node to resolve
Function param:	vis	a set of functors
Return value:	a copy	
Exceptions:	`noexcept`	
Header file:	ara/rest/ogm/visit.h	

Namespace:	ara::rest::ogm
Description:	See documentation of void Visit(const Node&, Visitors&&);.

Table 8.321: ara::rest::ogm::Visit

[SWS_REST_02346] Visit ⌈Table 8.321 describes the interface `Visit`.⌋ *(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.34.5 Visit

Service name:	Visit	
Type:	`Non-member function`	
Syntax:	`template <typename NodeT , typename... Visitors> void ara::rest::ogm::Visit(NodeT &u, Visitors &&...vis)`	
Function param:	u	OGM node to resolve
Function param:	vis	a set of functors
Return value:	a copy	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/visit.h	
Namespace:	ara::rest::ogm	
Description:	See documentation of void Visit(const Node&, Visitors&&);.	

Table 8.322: ara::rest::ogm::Visit

[SWS_REST_02347] Visit ⌈Table 8.322 describes the interface `Visit`.⌋ *(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.34.6 Visit

Service name:	Visit	
Type:	`Non-member function`	
Syntax:	`template <typename NodeT , typename... Visitors> void ara::rest::ogm::Visit(Pointer< NodeT > &u, Visitors &&...vis)`	
Function param:	u	OGM node to resolve
Function param:	vis	a set of functors
Return value:	a copy	
Exceptions:	Implementation-defined	
Header file:	ara/rest/ogm/visit.h	
Namespace:	ara::rest::ogm	
Description:	See documentation of void Visit(const Node&, Visitors&&);.	

Table 8.323: ara::rest::ogm::Visit

[SWS_REST_02348] Visit ⌈Table 8.323 describes the interface `Visit`.⌋ *(RS_CM_00300, RS_CM_00305, RS_CM_00306, RS_CM_00307, RS_CM_00308)*

8.35 ara::rest

8.35.1 RequestMethod

Name:	`RequestMethod`	
Type:	`Non-member enumeration`	

Range:	`kGet`
	`kPost`
	`kPut`
	`kDelete`
	`kOptions`
	`kAny`
		`= 1 « 0`
		`= 1 « 1`
		`= 1 « 2`
		`= 1 « 3`
		`= 1 « 4`
		`= ~std::underlying_type<RequestMethod>::type{0}`

Syntax:	`enum class RequestMethod : std::uint32_t { kGet = 1 « 0, kPost = 1 « 1, kPut = 1 « 2, kDelete = 1 « 3, kOptions = 1 « 4, kAny = ~std::underlying_type<RequestMethod>::type{0} };`	
Header file:	ara/rest/endpoint.h	
Namespace:	ara::rest	
Description:	Specifies a set possible API access methods. RequestMethod largely corresponds to typical RESTful API access methods.	

Table 8.324: ara::rest::RequestMethod

[SWS_REST_02349] RequestMethod ⌈Table 8.324 describes the enumeration datatype `ara::rest::RequestMethod`.⌋ *(RS_CM_00300)*

8.35.2 SubscriptionState

Name:	`SubscriptionState`	
Type:	`Non-member enumeration`	

Range:	`kSubscribed`
	`kCanceled`
	`kInvalid`	

Syntax:	`enum class SubscriptionState { kSubscribed, kCanceled, kInvalid };`	
Header file:	ara/rest/endpoint.h	
Namespace:	ara::rest	
Description:	Denotes the state of the subscription relation represented by an Event. The enumerators have the following meaning:	

Table 8.325: ara::rest::SubscriptionState

[SWS_REST_02350] SubscriptionState ⌈Table 8.325 describes the enumeration datatype `ara::rest::SubscriptionState`.⌋*(RS_CM_00300)*

8.35.3 EventPolicy

Name:	`EventPolicy`
Type:	`Non-member enumeration`
Range:	`kTriggered` = `1u << 0` `kPeriodic` = `1u << 1`
Syntax:	`enum class EventPolicy : std::uint32_t { kTriggered = 1u << 0, kPeriodic = 1u << 1 };`
Header file:	ara/rest/endpoint.h
Namespace:	ara::rest
Description:	Mode of operation for event subscriptions. Defines the mode of operation for event subscriptions. The modes have the following semantics:

Table 8.326: ara::rest::EventPolicy

[SWS_REST_02351] EventPolicy ⌈Table 8.326 describes the enumeration datatype `ara::rest::EventPolicy`.⌋*(RS_CM_00300)*

8.35.4 ShutdownPolicy

Name:	`ShutdownPolicy`
Type:	`Non-member enumeration`
Range:	`kForced` `kGraceful`
Syntax:	`enum class ShutdownPolicy : std::uint32_t { kForced, kGraceful };`
Header file:	ara/rest/endpoint.h
Namespace:	ara::rest
Description:	Specifies shutdown behavior of endpoints. Endpoints can shut down "gracefully", which allows all oongoing transactions to finish while blocking the caller. A forced shutdown must cancel or terminate all transations as fast as possible does not block the caller for "unreasonably" long period of time. During a forced shutdown I/O is not allowed. Precise semantics of these policies are implementation defined.

Table 8.327: ara::rest::ShutdownPolicy

[SWS_REST_02352] ShutdownPolicy ⌈Table 8.327 describes the enumeration datatype `ara::rest::ShutdownPolicy`.⌋*(RS_CM_00300)*

8.35.5 StartupPolicy

Name:	`StartupPolicy`
Type:	`Non-member enumeration`
Range:	`kDetached` `kAttached`
Syntax:	`enum class StartupPolicy : std::uint32_t { kDetached, kAttached };`
Header file:	ara/rest/endpoint.h
Namespace:	ara::rest
Description:	Specifies whether a server will detach itself from its owning context. If a server is started "detached" then are::rest::Server::Start() does not block. Effecticely it will request a separate exection context (such as a thread) from

Table 8.328: ara::rest::StartupPolicy

[SWS_REST_02353] StartupPolicy ⌈Table 8.328 describes the enumeration datatype `ara::rest::StartupPolicy`.⌋ *(RS_CM_00300)*

8.35.6 Function

Name:	`Function`
Type:	`Non-member type alias`
Syntax:	`template <typename T>` `using ara::rest::Function = std::function<T>`
Header file:	ara/rest/function.h
Namespace:	ara::rest
Description:	A generalized function pointer.

Table 8.329: ara::rest::Function

[SWS_REST_02354] Function ⌈Table 8.329 describes the type alias `ara::rest::Function`.⌋ *(RS_CM_00300, RS_CM_00311)*

8.35.7 Pointer

Name:	`Pointer`
Type:	`Non-member type alias`
Syntax:	`template<typename T>` `using ara::rest::Pointer = std::unique_ptr<T>`
Header file:	ara/rest/pointer.h
Namespace:	ara::rest
Description:	The equivalent if std::unique_ptr for ara::rest internal uses.

Table 8.330: ara::rest::Pointer

[SWS_REST_02355] **Pointer** ⌈Table 8.330 describes the type alias `ara::rest::Pointer`.⌋*(RS_CM_00300, RS_CM_00311)*

8.35.8 BasicString

Name:	`BasicString`
Type:	`Non-member type alias`
Syntax:	`template<typename CharT, typename TraitsT = std::char_traits<CharT> > using ara::rest::BasicString = std::basic_string<CharT, TraitsT, ara::rest::StdAllocator<CharT> >`
Header file:	ara/rest/string.h
Namespace:	ara::rest
Description:	Equivalent of std::basic_string but with a custom ara::rest::Allocator adaptor.

Table 8.331: ara::rest::BasicString

[SWS_REST_02356] **BasicString** ⌈Table 8.331 describes the type alias `ara::rest::BasicString`.⌋*(RS_CM_00300, RS_CM_00311)*

8.35.9 String

Name:	`String`
Type:	`Non-member type alias`
Syntax:	`using ara::rest::String = BasicString<char>`
Header file:	ara/rest/string.h
Namespace:	ara::rest
Description:	typedef for string.

Table 8.332: ara::rest::String

[SWS_REST_02357] **String** ⌈Table 8.332 describes the type alias `ara::rest::String`.⌋*(RS_CM_00300, RS_CM_00311)*

8.35.10 BasicStringView

Name:	`BasicStringView`
Type:	`Non-member type alias`
Syntax:	`template<typename CharT, typename TraitsT = std::char_traits<CharT> > using ara::rest::BasicStringView = BasicString<CharT, TraitsT>`
Header file:	ara/rest/string.h
Namespace:	ara::rest
Description:	typedef for StringView.

Table 8.333: ara::rest::BasicStringView

[SWS_REST_02358] BasicStringView ⌈Table 8.333 describes the type alias `ara::rest::BasicStringView`.⌋*(RS_CM_00300, RS_CM_00311)*

8.35.11 StringView

Name:	`StringView`
Type:	`Non-member type alias`
Syntax:	`using ara::rest::StringView = BasicStringView<char>`
Header file:	ara/rest/string.h
Namespace:	ara::rest
Description:	typedef for StringView.

Table 8.334: ara::rest::StringView

[SWS_REST_02359] StringView ⌈Table 8.334 describes the type alias `ara::rest::StringView`.⌋*(RS_CM_00300, RS_CM_00311)*

8.35.12 Task

Name:	`Task`
Type:	`Non-member type alias`
Syntax:	`template<typename T>` `using ara::rest::Task = std::future<T>`
Header file:	ara/rest/task.h
Namespace:	ara::rest
Description:	Represents an asynchronous task for which a user might want to wait for.

Table 8.335: ara::rest::Task

[SWS_REST_02360] Task ⌈Table 8.335 describes the type alias `ara::rest::Task`.⌋*(RS_CM_00300, RS_CM_00311)*

8.35.13 duration_t

Name:	`duration_t`
Type:	`Non-member type alias`
Syntax:	`using ara::rest::duration_t = std::chrono::microseconds`
Header file:	ara/rest/types.h
Namespace:	ara::rest
Description:	Specifies an amount of time of granularity of at least microseconds.

Table 8.336: ara::rest::duration_t

[SWS_REST_02361] duration_t ⌈Table 8.336 describes the type alias `ara::rest::duration_t`.⌋*(RS_CM_00300)*

8.35.14 operator==

Service name:	operator==	
Type:	`Non-member function`	
Syntax:	`bool ara::rest::operator==(const Allocator &a, const Allocator &b)`	
Function param:	a	an allocator
Function param:	b	an allocator
Return value:	true allocators compare equal	
Exceptions:	Implementation-defined	
Header file:	ara/rest/allocator.h	
Namespace:	ara::rest	
Description:	Tests two allocators for equality.	

Table 8.337: ara::rest::operator==

[SWS_REST_02362] operator== ⌈Table 8.337 describes the interface `operator==`.⌋ *(RS_CM_00300)*

8.35.15 operator!=

Service name:	operator!=	
Type:	`Non-member function`	
Syntax:	`bool ara::rest::operator!=(const Allocator &a, const Allocator &b)`	
Function param:	a	an allocator
Function param:	b	an allocator
Return value:	true allocators compare unequal	
Exceptions:	Implementation-defined	
Header file:	ara/rest/allocator.h	
Namespace:	ara::rest	
Description:	Tests two allocators for inequality.	

Table 8.338: ara::rest::operator!=

[SWS_REST_02363] operator!= ⌈Table 8.338 describes the interface `operator!=`.⌋ *(RS_CM_00300)*

8.35.16 NewDeleteAllocator

Service name:	NewDeleteAllocator
Type:	`Non-member function`
Syntax:	`Allocator* ara::rest::NewDeleteAllocator()`
Function param:	None
Return value:	a pointer to a NewDeleteAllocator
Exceptions:	`noexcept`
Header file:	ara/rest/allocator.h
Namespace:	ara::rest

Description:	Identical to std::pmr::new_delete_resource.

Table 8.339: ara::rest::NewDeleteAllocator

[SWS_REST_02364] NewDeleteAllocator ⌈Table 8.339 describes the interface `NewDeleteAllocator`.⌋*(RS_CM_00300)*

8.35.17 GetDefaultAllocator

Service name:	GetDefaultAllocator
Type:	`Non-member function`
Syntax:	`Allocator* ara::rest::GetDefaultAllocator()`
Function param:	None
Return value:	a pointer to the default allocator
Exceptions:	`noexcept`
Header file:	ara/rest/allocator.h
Namespace:	ara::rest
Description:	See std::pmr::get_default_allocator for details.

Table 8.340: ara::rest::GetDefaultAllocator

[SWS_REST_02365] GetDefaultAllocator ⌈Table 8.340 describes the interface `GetDefaultAllocator`.⌋*(RS_CM_00300)*

8.35.18 SetDefaultAllocator

Service name:	SetDefaultAllocator	
Type:	`Non-member function`	
Syntax:	`Allocator* ara::rest::SetDefaultAllocator(Allocator *a)`	
Function param:	a	an allocator
Return value:	a pointer to the allocator just set	
Exceptions:	`noexcept`	
Header file:	ara/rest/allocator.h	
Namespace:	ara::rest	
Description:	See std::pmr::set_default_allocator for details.	

Table 8.341: ara::rest::SetDefaultAllocator

[SWS_REST_02366] SetDefaultAllocator ⌈Table 8.341 describes the interface `SetDefaultAllocator`.⌋*(RS_CM_00300)*

8.35.19 operator==

Service name:	operator==

Type:	`Non-member function`	
Syntax:	`template <typename T , typename U > bool ara::rest::operator==(const StdAllocator< T > &a, const StdAllocator< U > &b)`	
Function param:	a	an allocator
Function param:	b	an allocator
Return value:	true if memory allocated in one can be freed via other	
Exceptions:	`noexcept`	
Header file:	ara/rest/allocator.h	
Namespace:	ara::rest	
Description:	Tests allocators for equality.	

Table 8.342: ara::rest::operator==

[SWS_REST_02367] operator== ⌈Table 8.342 describes the interface `operator==`.⌋ *(RS_CM_00300)*

8.35.20 operator!=

Service name:	operator!=	
Type:	`Non-member function`	
Syntax:	`template <typename T , typename U > bool ara::rest::operator!=(StdAllocator< T > const &x, StdAllocator< U > const &y)`	
Function param:	x	an allocator
Function param:	y	an allocator
Return value:	true if memory allocated in x cannot be freed via y	
Exceptions:	`noexcept`	
Header file:	ara/rest/allocator.h	
Namespace:	ara::rest	
Description:	Tests allocators for inequality.	

Table 8.343: ara::rest::operator!=

[SWS_REST_02368] operator!= ⌈Table 8.343 describes the interface `operator!=`.⌋ *(RS_CM_00300)*

8.35.21 operator|

Service name:	operator|	
Type:	`Non-member function`	
Syntax:	`constexpr RequestMethod ara::rest::operator|(RequestMethod a, RequestMethod b)`	
Function param:	a	a (set of) request method enumerator(s)
Function param:	b	a (set of) request method enumerator(s)
Return value:	a set of request method enumerator(s)	
Exceptions:	`noexcept`	

Header file:	ara/rest/endpoint.h
Namespace:	ara::rest
Description:	Computes a set of RequestMethod enumerators.

Table 8.344: ara::rest::operator|

[SWS_REST_02369] operator| ⌈Table 8.344 describes the interface `operator|`.⌋ *(RS_CM_00300)*

8.35.22 operator|

Service name:	operator|	
Type:	`Non-member function`	
Syntax:	`constexpr EventPolicy ara::rest::operator|(EventPolicy a, EventPolicy b)`	
Function param:	a	a (set of) request event policy enumerator(s)
Function param:	b	a (set of) request event policy enumerator(s)
Return value:	a set of request event policy enumerator(s)	
Exceptions:	`noexcept`	
Header file:	ara/rest/endpoint.h	
Namespace:	ara::rest	
Description:	Computes a set of EventPolicy enumerators.	

Table 8.345: ara::rest::operator|

[SWS_REST_02370] operator| ⌈Table 8.345 describes the interface `operator|`.⌋ *(RS_CM_00300)*

8.35.23 MakeIteratorRange

Service name:	MakeIteratorRange	
Type:	`Non-member function`	
Syntax:	`template <typename IterT > IteratorRange<IterT> ara::rest::MakeIteratorRange(IterT a, IterT b)`	
Function param:	a	iterator that denotes the start of the sequence
Function param:	b	iterator that denotes the end of the sequence
Return value:	an IteratorRange	
Exceptions:	Implementation-defined	
Header file:	ara/rest/iterator.h	
Namespace:	ara::rest	
Description:	Helper for type deduction to construct an IteratorRange.	

Table 8.346: ara::rest::MakeIteratorRange

[SWS_REST_02371] MakeIteratorRange ⌈Table 8.346 describes the interface `MakeIteratorRange`.⌋ *(RS_CM_00300)*

8.35.24 Resolve

Service name:	Resolve	
Type:	`Non-member function`	
Syntax:	`Uri ara::rest::Resolve(const Uri &base, const Uri &rel, Allocator *alloc=GetDefaultAllocator())`	
Function param:	base	the URI to resolve against
Function param:	rel	a relative URI
Function param:	alloc	an allocator
Return value:	a resolved URI	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Namespace:	ara::rest	
Description:	Resolves a relative URI against a base URI. See section 5.2 of RFC 3986 for the algorithm used.	

Table 8.347: ara::rest::Resolve

[SWS_REST_02372] Resolve ⌈Table 8.347 describes the interface `Resolve`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.35.25 Normalize

Service name:	Normalize	
Type:	`Non-member function`	
Syntax:	`Uri ara::rest::Normalize(const Uri &uri, Allocator *alloc=GetDefaultAllocator())`	
Function param:	uri	URI to normalize
Function param:	alloc	an allocator
Return value:	a noralized URI	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Namespace:	ara::rest	
Description:	Normalizes a given URI.	

Table 8.348: ara::rest::Normalize

[SWS_REST_02373] Normalize ⌈Table 8.348 describes the interface `Normalize`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.35.26 Relativize

Service name:	Relativize	
Type:	`Non-member function`	
Syntax:	`Uri ara::rest::Relativize(const Uri &base, const Uri &uri, Allocator *alloc=GetDefaultAllocator())`	
Function param:	base	a base URI as reference
Function param:	uri	a URI to relativize

Function param:	alloc	an allocator
Return value:	a relative URI	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Namespace:	ara::rest	
Description:	Relativizes a URI with respect to a given base URI. The relativization of the given URI against this URI is computed as follows:	

Table 8.349: ara::rest::Relativize

[SWS_REST_02374] Relativize ⌈Table 8.349 describes the interface `Relativize`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.35.27 ToString

Service name:	ToString	
Type:	`Non-member function`	
Syntax:	`String ara::rest::ToString(const Uri &uri, Uri::Part part, bool encode, Allocator *alloc=GetDefaultAllocator())`	
Function param:	uri	URI to encode
Function param:	part	denotes which components of a URI should be encoded
Function param:	encode	if true, then the string will be percent-encoded. If false, the string must not be string encoded.
Function param:	alloc	a user-defined allocator passed to the string object being returned
Return value:	the encoded URI	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Namespace:	ara::rest	
Description:	Returns a string representation of a Uri.	

Table 8.350: ara::rest::ToString

[SWS_REST_02375] ToString ⌈Table 8.350 describes the interface `ToString`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.35.28 ToString

Service name:	ToString	
Type:	`Non-member function`	
Syntax:	`String ara::rest::ToString(const Uri &uri, Uri::Part part, Allocator *alloc=GetDefaultAllocator())`	
Function param:	uri	URI to encode
Function param:	part	denotes which components of a URI should be encoded

Function param:	alloc	a user-defined allocator passed to the string object being returned
Return value:	the encoded URI	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Namespace:	ara::rest	
Description:	Returns a string representation of a Uri.	

Table 8.351: ara::rest::ToString

[SWS_REST_02376] ToString ⌈Table 8.351 describes the interface `ToString`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.35.29 ToString

Service name:	ToString	
Type:	`Non-member function`	
Syntax:	`String ara::rest::ToString(const Uri &uri, Allocator *alloc=GetDefaultAllocator())`	
Function param:	uri	URI to encode
Function param:	alloc	a user-defined allocator passed to the string object being returned
Return value:	the encoded URI	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Namespace:	ara::rest	
Description:	Returns a string representation of a Uri.	

Table 8.352: ara::rest::ToString

[SWS_REST_02377] ToString ⌈Table 8.352 describes the interface `ToString`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.35.30 ToString

Service name:	ToString	
Type:	`Non-member function`	
Syntax:	`String ara::rest::ToString(Uri &&uri, Uri::Part part, bool encode, Allocator *alloc=GetDefaultAllocator())`	
Function param:	uri	URI to encode
Function param:	part	denotes which components of a URI should be encoded
Function param:	encode	if true, then the string will be percent-encoded. If false, the string must not be string encoded.
Function param:	alloc	a user-defined allocator passed to the string object being returned
Return value:	the encoded URI	
Exceptions:	Implementation-defined	

Header file:	ara/rest/uri.h
Namespace:	ara::rest
Description:	Returns a string representation of a Uri.

Table 8.353: ara::rest::ToString

[SWS_REST_02378] ToString ⌈Table 8.353 describes the interface `ToString`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.35.31 ToString

Service name:	ToString	
Type:	`Non-member function`	
Syntax:	`String ara::rest::ToString(Uri &&uri, Uri::Part part, Allocator *alloc=GetDefaultAllocator())`	
Function param:	uri	URI to encode
Function param:	part	denotes which components of a URI should be encoded
Function param:	alloc	a user-defined allocator passed to the string object being returned
Return value:	the encoded URI	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Namespace:	ara::rest	
Description:	Returns a string representation of a Uri.	

Table 8.354: ara::rest::ToString

[SWS_REST_02379] ToString ⌈Table 8.354 describes the interface `ToString`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.35.32 ToString

Service name:	ToString	
Type:	`Non-member function`	
Syntax:	`String ara::rest::ToString(Uri &&uri, Allocator *alloc=GetDefaultAllocator())`	
Function param:	uri	URI to encode
Function param:	alloc	a user-defined allocator passed to the string object being returned
Return value:	the encoded URI	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uri.h	
Namespace:	ara::rest	
Description:	Returns a string representation of a Uri.	

Table 8.355: ara::rest::ToString

[SWS_REST_02380] ToString ⌈Table 8.355 describes the interface `ToString`.⌋ *(RS_CM_00300, RS_CM_00304)*

8.35.33 ToString

Service name:	ToString	
Type:	`Non-member function`	
Syntax:	`String ara::rest::ToString(const Uuid &uuid, Allocator *alloc=GetDefaultAllocator())`	
Function param:	uuid	a UUID
Function param:	alloc	an allocator
Return value:	its canonic textual representation	
Exceptions:	Implementation-defined	
Header file:	ara/rest/uuid.h	
Namespace:	ara::rest	
Description:	Converts Uuid into its canonical textual representation.	

Table 8.356: ara::rest::ToString

[SWS_REST_02381] ToString ⌈Table 8.356 describes the interface `ToString`.⌋ *(RS_CM_00300, RS_CM_00304)*

A Mentioned Class Tables

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.