RIC - Reference Implementation

The VWS vernetzt-Testbeds

V1.0 RC01

Technical Report

15. April 2021

Universität Magdeburg
Fakultät für Elektrotechnik und Informationstechnik
Institut für Automatisierungstechnik
Postfach 4120, D-39016 Magdeburg
Germany

Index

1 Abbreviation

AAS	Asset Administration Shell
I4.0	Industrie 4.0
RIC	Registry Infrastructure Component
VWS	Verwaltungsschale
ALP	Application Layer Protocol
RIC	Registry Infrastructure Component
RICSS	RIC Smart Space

2 Relevant Documentation

- VDI 2193 Blatt 1: Sprache für I4.0-Komponenten - Struktur von Nachrichten
- FIPA ACL Message Structure Specification (http://www.fipa.org/specs/fipa00061/SC00061G.html)
- AASiD Part 1: The exchange of information between partners in the value chain of Industrie 4.0 (Version 2.0.1)
- AASiD Part 2: Interoperability at Runtime – Exchanging Information via Application Programming Interfaces (V1.0RC01 for Review)

3 Introduction

AASiD Part 1 firstly introduces the term Asset Administration Shell as digital representation of a real-world physical asset. Secondly it presents a meta model for structuring the information about an asset in terms of submodels, submodel properties, assets, and concept dictionaries. The entire data structure representing the asset can be packaged in either AASx, JSON or XML format and can be exchanged between the trading partners (Type 1 AAS).

As AAS being a digital representation of the physical asset, AASiD part 2 specifies the standards for platform independent programming interfaces that AAS needs to support for making information accessible about itself available in the digital world (Type 2 AAS). The AASiD part 2 also introduces the concept of the registry that accepts, stores and provides information about any AAS. Along with the registry, AASiD part 2 defines the endpoint descriptor data structure for representing the information about the accessibility of an AAS in the digital world. An AAS would need to create its descriptor information and publish it to the registry, so that it could be accessible by any other AAS or applications.

AASiD part 2 mentions about Type 3 AAS that are expected to communicate with each other probably to solve a real-world problem or to perform a specific task. A standard message format for exchange of information would reduce the complexity of interpreting it. VDI/VDE 2193 Part 1 and Part 2 provide guidelines for structuring the information and introduces a new standard consisting frame and interactionElements. The frame part represents the sender and receiver information, message type, and interaction protocol information. Having a common message format addresses the needs of semantic interoperability.

This document defines an interaction protocol for registration of an AAS with the registry. The AAS could utilize the protocols for registering with the registry. In addition, this document defines the HeartBeat interaction protocol so that registry can maintain active status of all the AAS that are associated with it.

4 Registry Infrastructure Component (RIC)

This specification utilizes standards and concepts introduced in the AASiD part 1 and 2, proposes architecture for an infrastructure component termed as Registry Infrastructure Component (RIC). The RIC implements the registry concept and also acts as a message broker (not described in this document). The RIC is a system consisting of a data base and multiple interfaces. Each interface is implemented as own plugin which is comprised of a specific ALP server and the client. The RIC server implementations actively listen to all the inbound messages at the different interfaces, categorize them and assign a unique thread.

The **Fehler! Verweisquelle konnte nicht gefunden werden.** presents the interaction between RIC and an AAS. AAS 1 is registering itself and AAS 2 gets the endpoint and other information from the RIC. The HeartBeat interaction should happen only after the registration of the AAS is successfully completed. The subsections 6.1 and 0 present in detail about these interaction protocols and also the composition of the relevant I4.0 messages.

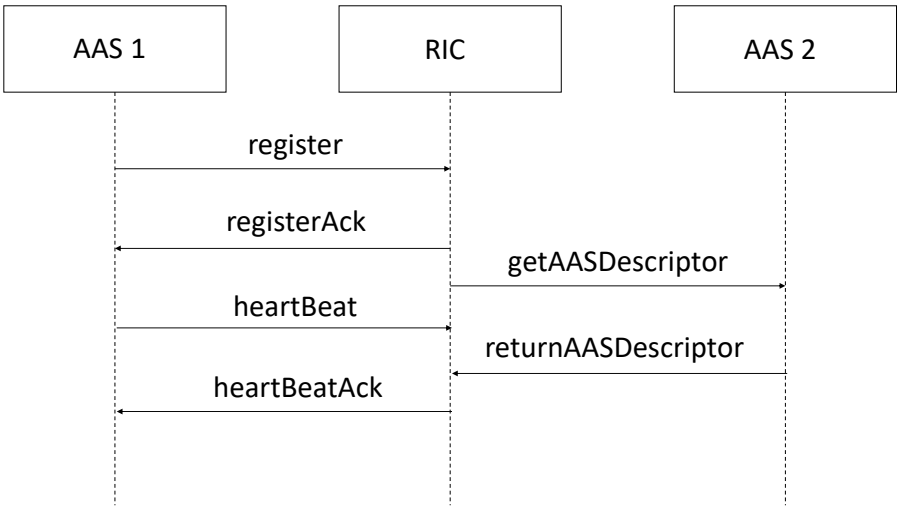

Figure 1 Interactions between AAS and RIC

Figure 2 shows an overview about the interfaces of the registry in detail. The registry has three communication interfaces: http, MQTT and COAP (also https behind firewalls and proxies). Additionally, there are two different registry services types: one is related to AASiD part 2 which is only usable with the http communication interface and the other is using the I4.0 Language (according VDI/VDE 2103-1) which is usable for all three communication interfaces (http, MQTT and COAP). Using the http interface the status of AAS is got which is based on the heart beat of the AAS. Figure 2 also shows the details of the endpoints and name spaces.

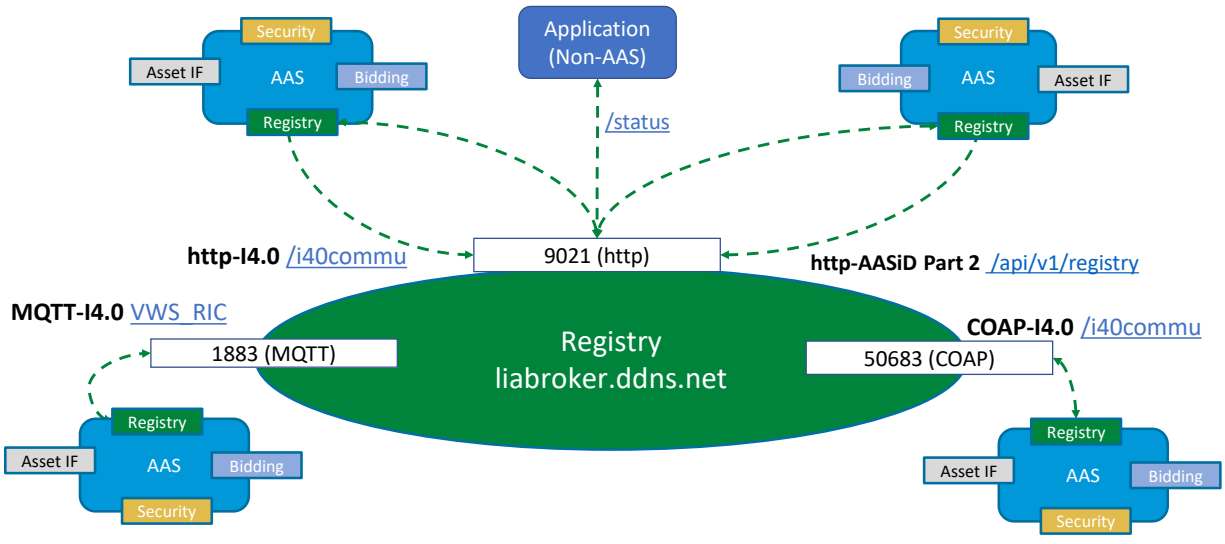

Figure 2: Registry overview

Chapter 5 define the interface related to the http-AASiD Part 2 definition using the according descriptor. This is available in http only. Chapter 6 describes the registry interaction using the I4.0 language according to VDI/VDE 2193. This interface can be used for http, MQTT and COAP.

5 RIC as Registry according to AASiD part 2 and http communication protocol

The RIC implements a registry service as specified in the AASiD part 2. The Table 1 lists all the HTTP URI's that the reference implementation of RIC provides. The table also lists the type of rest services attributed with each of the URI. The Table 2, Table 3, Table 4, Table 5 present the appropriate responses for each service associated with the HTTP URI's.

S No.	HTTP URI	Type	Description
1	http://liabroker.ddns.net:9021/api/v1/registry	GET	Retrieve the entire registry details
2	http://liabroker.ddns.net:9021/api/v1/registry/{aasId}	GET	Retrieve descriptor details of the AAS with id **aasId**
3	http://liabroker.ddns.net:9021/api/v1/registry/{aasId}	PUT	Insert a new AAS descriptor with id aasId. The descriptor data should be sent as a JSON along with the request
4	http://liabroker.ddns.net:9021/api/v1/registry/{aasId}	DELETE	Delete an existing descriptor of the AAS with id aasId
5	http://liabroker.ddns.net:9021/api/v1/registry/{aasId}/submodels	GET	Retrieve all the submodel descriptors of the AAS with id **aasId**
6	http://liabroker.ddns.net:9021/api/v1/registry/{aasId}/submodels/{submodelId}	GET	Retrieve descriptor of the submodel with id **submodelId** of the AAS with id **aasId**
7	http://liabroker.ddns.net:9021/api/v1/registry/{aasId}/submodels/{submodelId}	PUT	Insert a submodel descriptor with id submodelId of the particular AAS with id aasId
8	http://liabroker.ddns.net:9021/api/v1/registry/{aasId}/submodels/{submodelId}	DELETE	Delete a submodel descriptor with id submodelId of the particular AAS with id aasId

Table 1 RIC namespaces list for AAS Descriptor Information as per AASiD part 2

HTTP responses for different REST interfaces

1. http://liabroker.ddns.net:9021/api/v1/registry

Operation	Condition	Response Message	Status Code
GET	Registry contains atleast one entry	Returns a list all the AAS descriptors that have registered with the registry	200
GET	No entries	No Asset Administration Shell descriptors are yet registered	404
GET	Internal Error	Unexpected Internal Server Error	500

Table 2 HTTP Responses for listing the entire Registry

2. http://liabroker.ddns.net:9021/api/v1/registry/{aasId}

Operation	Condition	Response Message	Status Code
GET	Registry contains the relevant entry	Returns the relevant AAS descriptor	200
GET	Relevant descriptor is not present	No Asset Administration Shell with passed id found	404
PUT	Relevant AAS descriptor is not present	The Asset Administration Shell's registration was successful	200
PUT	Relevant AAS descriptor is already present	The Asset Administration Shell's registration was successfully renewed	200
PUT	AAS descriptor not well formed	The syntax of the passed Asset Administration Shell descriptor is not valid or malformed request	400
PUT	AAS ID present in the descriptor and URI do not match	The namspace AASID value and the IdShort value do not match	500
DELETE	Relevant AAS descriptor is not present	No Asset Administration Shell Descriptor with passed id found	404
DELETE	Relevant AAS descriptor is already present	The Asset Administration Shell Descriptor was deleted successfully	200
GET / PUT / DELETE	Internal Error	Unexpected Internal Server Error	500

Table 3 HTTP Responses for different REST services pertaining to one AAS

3. http://liabroker.ddns.net:9021/api/v1/registry/{aasId}/submodels

Operation	Condition	Response Message	Status Code
GET	Registry contains the relevant entry	Returns the list of all the submodel descriptors	200
GET	Relevant AAS descriptor is not present	No Asset Administration Shell descriptors are yet registered	404
GET	Relevant AAS does not have any submodel descriptor	The Asset Administration Shell Descriptors does not have any submodel descriptors	404
GET	Internal Error	Unexpected Internal Server Error	500

Table 4 HTTP Responses for different REST services pertaining to all submodels of one AAS

4. http://liabroker.ddns.net:9021/api/v1/registry/{aasId}/submodels/{submodelId}

Operation	Condition	Response Message	Status Code
GET	Relevant AAS descriptor is not present	No Asset Administration Shell Descriptor with passed id found	404
GET	Relevant submodel descriptor is not present	Submodel with passed id not found	404
GET	Relevant submodel descriptor is found	Submodel descriptor of the relevant submodel	200
PUT	Relevant AAS descriptor is not present	No Asset Administration Shell Descriptor with passed id found	404
PUT	Relevant submodel descriptor is already present	The Submodel descriptor was successfully renewed	200
PUT	Relevant submodel descriptor is not present	The Submodel descriptor was created successfully	200
PUT	Submodel descriptor not well formed	The syntax of the passed Asset Administration Shell is not valid or malformed request	400
PUT	SubmodelId present in the descriptor and the id passed in the uri does not match	The Namespace SubmodelId value and the IdShort value in the data are not matching	500
DELETE	Relevant submodel Iis not present	Submodel Descriptor with passed id not found	404
DELETE	Relevant submodel descriptor is present	The Submodel Descriptor was successfully unregistered	200
GET / PUT / DELETE	Internal Error	Unexpected Internal Server Error	500

Table 5 HTTP Responses for different REST services pertaining to one submodel of an AAS

6 Interactions with AAS using the I4.0 language

There are two new interaction protocols which are registration and heartbeat. Each of the protocol consists of two message types, the message and its acknowledgement. These interactions are presented in the **Fehler! Verweisquelle konnte nicht gefunden werden.** register/registerAck, heartbeat/heartBeatAck along with getAASDescriptor and returnAASDescriptor. The Heart Beat interaction should happen only after the registration of the AAS is successfully completed. The subsections 6.1 and 0 presents in detail about these interaction protocols and also the composition of the relevant I4.0 messages.

6.1 Registration

Every AAS firstly needs to get itself registered with the RIC. It needs to embed its descriptor information within an I4.0 message packet of type “register” and send it to the RIC. RIC firstly validates the message, extracts the descriptor information and assigns it to an internal handler. The internal handler

firstly validates the descriptor against the standard descriptor schema and returns a negative acknowledgement in case of malformed data. Otherwise, the handler inserts the descriptor into the database and returns a positive acknowledgment. In case of any other internal errors, the RIC would return a negative acknowledgment.

Acknowledgment from the RIC will be an I4.0 message packet of type "registerack". RIC embeds a submodel **StatusResponse** into the interactionElements part of the registerack message. StatusResponse submodel is a collection of three properties **Status**, **Code** and **Message,** Values for these properties vary depending on the type of response. Table 6 StatusResponse Submodel Properties lists down values for the Submodel properties under different acknowledgements by the RIC for a register message.

Property Type	Descriptor Invalid	Internal Error	Registration Success	Registration Success (Updated)
Status	E	E	S	S
Code	400	500	200	200
Message	The syntax of the passed Asset Administration Shell descriptor is not valid or malformed request	Unexpected Internal Server Error	The Asset Administration Shell's registration was successful	The Asset Administration Shell's registration was successfully renewed

Table 6 StatusResponse Submodel Properties

Table 7presents relevant values for the I4.0 frame properties in case of register, registerack and HeartBeat messages. A point to note, **replyBy** and **replyTo** fields have one of the ALP RESTAPI(HTTP), MQTT or COAP. The replyTo indicates **"**the receiver AAS preferred ALP**"** and the replyBy property indicates **"**the ALP of the AAS that has created the message**"** (it indicates that the receiver AAS needs to send back the relevant message type as part of the interaction protocol involved over this ALP).

frame Element	Register I4.0 packet	RegisterAck I4.0 packet	HeartBeat I4.0 packet
semanticProtocol/keys/ type	GlobalReference	GlobalReference	GlobalReference
semanticProtocol/keys/ local	local	local	local
semanticProtocol/keys/ value	www.admin-shell.io/interaction/registration	www.admin-shell.io/interaction/registration	www.admin-shell.io/interaction/heartbeat
semanticProtocol/keys/ idType	False	False	False
type	register	registerack	HeartBeat
messageId	Set by the AAS	Set by the AAS	Set by the AAS
sender/identification/id	AAS IdShort	VWS_RIC	AAS IdShort
sender/identification/id Type	URI	URI	URI
sender/role/name	Register	RegistryHandler	AASHeartBeatHandler
receiver/identification/ id	VWS_RIC	AAS IdShort	VWS_RIC
receiver/identification/ idType	URI	URI	URI
receiver/role/name	RegistryHandler	Register	HeartBeatHandler

replyBy	RESTAPI / MQTT / COAP	RESTAPI / MQTT / COAP	RESTAPI / MQTT / COAP
replyTo	RESTAPI / MQTT / COAP	RESTAPI / MQTT / COAP	RESTAPI / MQTT / COAP
conversationId	Set by the AAS	Set by the AAS	Set by the AAS
interactionElements	AAS Descriptor as specified in the AASiD details part 2	As described in Table 6	NA

Table 7 I4.0 Frame details for register, registerack, HeartBeat, HeartBeatAck

The “receiver/identification/id” and the “receiver/role/name” for register and registerack messages are to be adhered as specified in the Table 7 An AAS agent can choose one of the MQTT, RESTAPI or COAP protocols for publishing the message to the RIC, relevant technical details are provided in the section 7

6.2 HeartBeat

An AAS can send an I4.0 message of type HeartBeat at regular intervals to RIC, however this is not mandated. Whenever the RIC receives a HeartBeat message, firstly it looks-up into the internal database whether the AAS is already registered or not. If the AAS is not registered, it would inform the AAS to get its descriptors registered. In case the AAS is registered, its active status is updated with latest timestamp. The active status of all the AAS can be retrieved from the RIC.

Protocol Type	Response Status Message	Status Code
HTTP	success	200
COAP	success	2.04

Table 8 Positive responses for HeartBeat I4.0 message

HeartBeat Acknowledgment varies according to the application layer protocol. In case of synchronous protocols I4.0 HeartBeatAck message packet is not sent, instead a relevant protocol status message is sent. In case of asynchronous protocol, there is no reply back to the AAS (Changes to be made in the implementation, in case the AAS is not registered, an HeartBeatAck message indicating the need for registration is to be sent). Table 8 and Table 9 highlight responses from the RIC in case of HeartBeat Message.

Protocol Type	Response Status Message	Status Code
HTTP	The AAS is not registered, please provide descriptors	500
COAP	The AAS is not registered, please provide descriptors	5.00

Table 9 Negative responses for HeartBeat I4.0 message

The Figure 3 presents the state machine of the RIC from the perspective of registry and heartbeat interactions. The complex internal mechanism is summarized into a simpler system so that it would be easy for the reader of this document to better understand the internal process. The registry message is

firstly validated, next the descriptor within it is validated, late the descriptor it is stored into the internal database and lastly returning an appropriate acknowledgement.

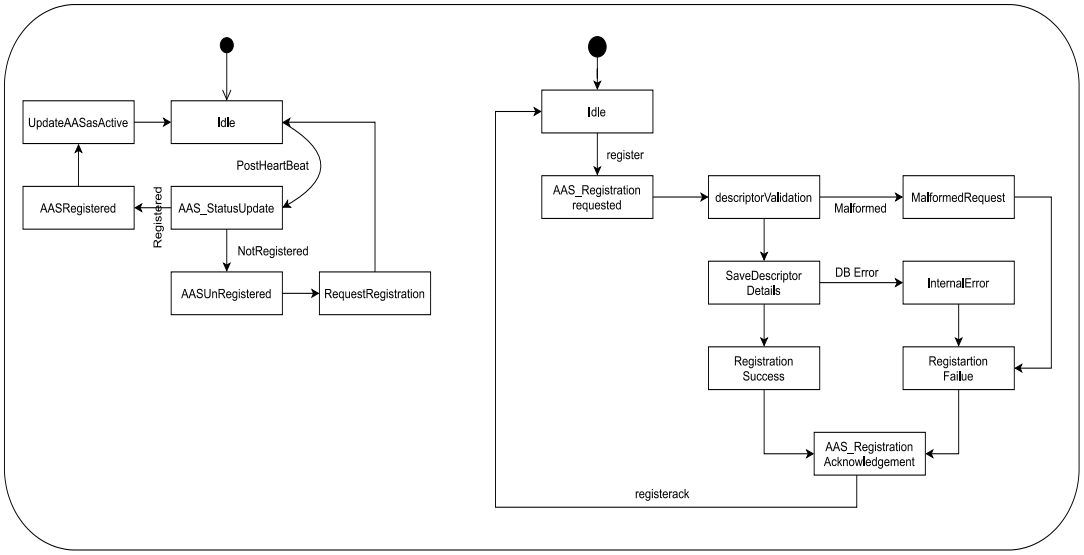

Figure 3 State Machine from the point of view of RIC

7 RIC interface access

RIC hosts different ALP plugins, it operates them parallelly, synchronizes their behaviour. A reference implementation of the RIC is hosted on the "**liabroker.ddns.net**" cloud, in this documentation the cloud space termed as RIC smart space. This cloud hosts the mosquito MQTT server (AMQP server is under consideration) and the RIC implementation itself hosts COAP and HTTP servers. The following sub-sections 7.1 to 7.3 present the technical details about the related servers.

7.1 RIC MQTT Server

Server Host: liabroker.ddns.net

Server Port: 1883

The client application that is part of the RIC MQTT plugin subscribes to the Topic "**VWS_RIC**". An AAS that prefers MQTT as the ALP should subscribe to the topic name same as it's "**idShort**", all the messages inbound to this AAS need to be published to this topic.

An AAS that just supports MQTT as the ALP, would need to publish the register I4.0 message with the frame details as presented in Table 7 I4.0 Frame details for register, registerack, HeartBeat, HeartBeatAckto the topic "**VWS_RIC**". The client application as the part of the MQTT plugin when it receives a new register message, allocates an internal handler over a separate to it. The Internal handler prepares the appropriate registerack message and submits to the mqtt client. The MQTT client then publishes the registerack message to the topic of the topic of the relevant AAS. Similarly, as AAS can publish its HeartBeat messages at regular intervals to the "**VWS_RIC**" topic.

7.2 RIC HTTP Server

Server Host: liabroker.ddns.net

Server Port: 9021

RIC implements a HTTP server over the port 9021, this HTTP server provides multiple namespaces. The Table 10 presents the list of URI along with the relevant rest operation. The status URI provides the active status of all the AAS agent at a particular for a GET request.

Name	S.No	HTTP URI	Type	Description
Status	1	http://liabroker.ddns.net:9021/status	GET	Retrieve status of all the connected AAS
Communication	2	http://liabroker.ddns.net:9021/i40commu	POST	Post an I4.0 message packet to RIC

Table 10 RIC HTTP URI's

An AAS can post its register I4.0 message packet over the communication URI. The client application part of the RIC HTTP plugin, accepts the inbound message and allocates an internal handler. Once the internal handler prepares and submits the registerack message, the client applications sends back a synchronous response. Similarly, the AAS can post the heartbeat messages over the communication and URI and expect responses as indicated in Table 8 and Table 9.

7.3 RIC COAP Server

Server Host: liabroker.ddns.net

Server Port: 50683

Similar to HTTP server, RIC also implements a COAP server. An AAS can post the register and heartbeat messages the communication URI as presented in Table 11 RIC COAP URI's

Name	HTTP URI	Type	Description
Communication	coap://liabroker.ddns.net:50683/i40commu	POST	Post an I4.0 message packet to RIC

Table 11 RIC COAP URI's

8 Conclusion

This document outlines technical specifications of the reference implementation of RIC and it is only from perspective of RIC as a registry. A RIC also acts a medium for exchange of I4.0 message packets between any two AASs, another document is published that details how an AAS can utilize the services of RIC for delivering I4.0 messages to another AAS implement a different application layer protocol.