Customer Requirements Specification

(Lastenheft)

(TINF19C, SWE I Praxisprojekt 2020/2021)

Project: *Service Registry*

Customer: *Rentschler & Holder*
Rotebühlplatz 41
70178 Stuttgart

Supplier: Team 4 (Daniel Baumann, Tim Diehl, Goran Erdeljan, Serdar Ilhan, Benedict Wetzel)
Rotebühlplatz 41
70178 Stuttgart

Version	Date	Author	Comment
0.1	07.09.2020		created

Allgemeine Hinweise:

Alles, was in dieser blauen Schriftart gesetzt ist, dient nur zur Erläuterung und sollte im fertigen Lastenheft nicht mehr auftauchen!
Der Umfang dieses Dokuments darf sechs Seiten nicht überschreiten.
Ein Lastenheft enthält eine grobe Beschreibung aller fachlichen Anforderungen, die das zu entwickelnde Produkt erfüllen muss. Die Inhalte des Lastenheftes (CRS) dienen als Grundlage für das Pflichtenheft und können -wenn sinnvoll- im Pflichtenheft (SRS) wieder verwendet werden.

CONTENTS

1. Goal

An asset in the meaning of Industry 4.0 can both be a single device and a Docker hosted application. In both cases there must be a way to announce the offered capabilities via a service discovery mechanism. A suitable mechanism for this is DNS-SD [1] [2].

The system will combine the capabilities of DNS-SD and the already existing Service-Registry developed by the OI4-Alliance.
A Service Registry is an application that offers a list of available Services in the network. It makes it possible for Users and other applications to find a service, that matches their requirements. Therefore, a Service-Registry shall also contain Data about the capabilities of the service.
The to be developed application must make it possible to, on the one hand use the DNS-SD mechanism to discover Services, that aren't already known to the Service Registry and on the other hand announce the already known Services via DNS-SD in the Network.

The new interface must be implemented and integrated into an Example project on Linux. The interface must, just like the provided Service Registry, run in a Docker environment. The Docker Environment will also contain a MQTT-Broker, which hosts the OI4-MessageBus. Additionally, there must be test applications, which also run in Docker Containers. These test-applications shall be able to list the services, that were announced over the network, via a simple GUI. The developed applications shall be made publicly available in form of an open-source project.

2. Product Environment

The usage environment of the software consists of the Docker-Application itself, which is connected to the OI4-Service-Registry via a MQTT-Broker. In this context the communication using the MQTT-Protocol is called the OI4-MessageBus. The to be developed Docker-Application shall act as an interface between the Service-Registry and a Service-Discovery mechanism called DNS-SD. This adds an additional way not only of registering new services, but also of announcing already known services to the network.
These services can for example be Industry 4.0 related. Having them registered at a Service-Registry makes them easily accessible.

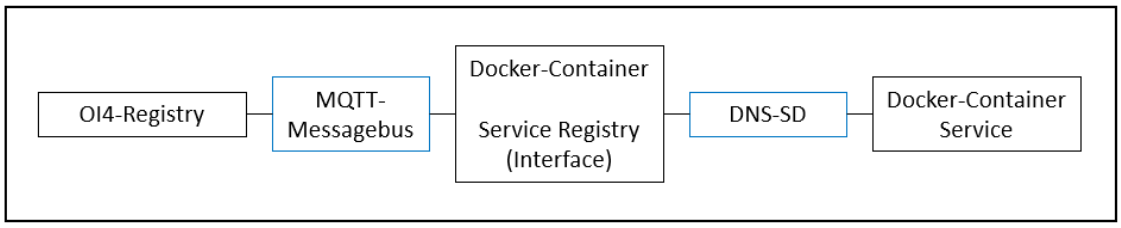

Figure x: Product Environment

3. Product Usage

The following business processes, use cases and features shall be supported by the system.

3.1. Business Processes

Falls notwendig, sind hier die identifizierten Geschäftsprozesse näher zu beschreiben. Jeder von ihnen erhält einen eigenen Unterabschnitt gemäß dem Template. In diesem Abschnitt wird der Ablauf der Geschäftsprozesse des vorigen Abschnittes genauer beschrieben. Diese Abläufe sind es, die das zu entwickelnde System ausschnittsweise unterstützen soll.

3.1.1. <BP.001>: <Grundfunktion>

Triggering Event:	*Is executed as soon as a device or an application communicate compliantly with the Open Industry 4.0*
Result:	OI4 Registry lists all devices and applications and tests all assets. Afterwards, the user receives a feedback
Involved Roles:	*Devices and applications*, employees

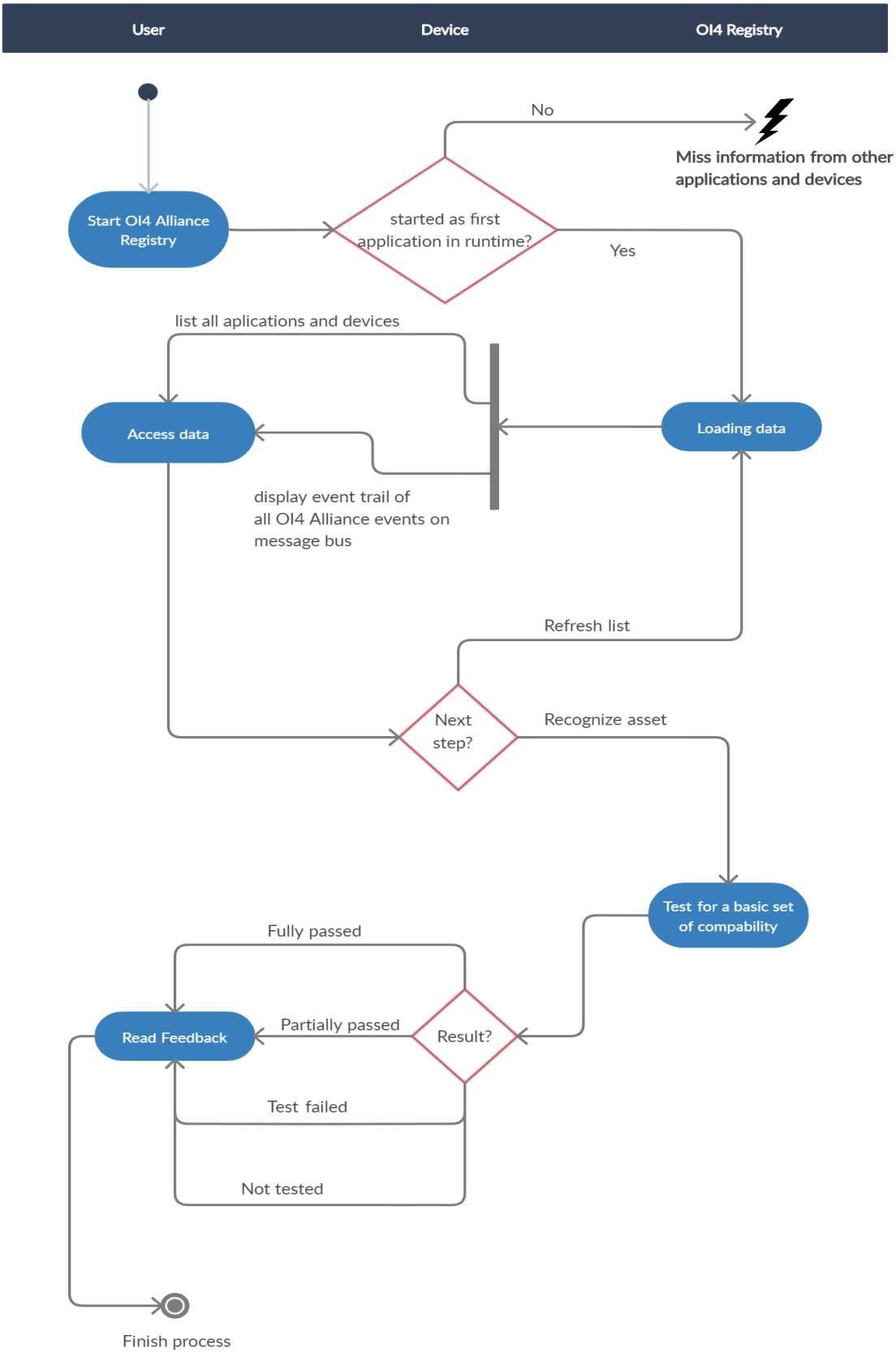
User
Device
OI4 Registry
No
Miss information from other applications and devices
Start OI4 Alliance Registry
started as first application in runtime?
Yes
list all aplications and devices
Access data
Loading data
display event trail of all OI4 Alliance events on message bus
Refresh list
Next step?
Recognize asset
Test for a basic set of compability
Fully passed
Read Feedback
Partially passed
Result?
Test failed
Not tested
Finish process

Figure 2.2: <BP.001> Workflow of general function

3.1.2. <BP.001>: <Publish on OI4 Messagebus>

Triggering Event:	*New DNS-SD entry on the network*
Result:	It is checked whether txt records match the specification. When it matches, it should be published on the OI4 Messagebus. Otherwise the entry will be ignored and the next new entry is waited for.
Involved Roles:	*Devices and applications*, employees

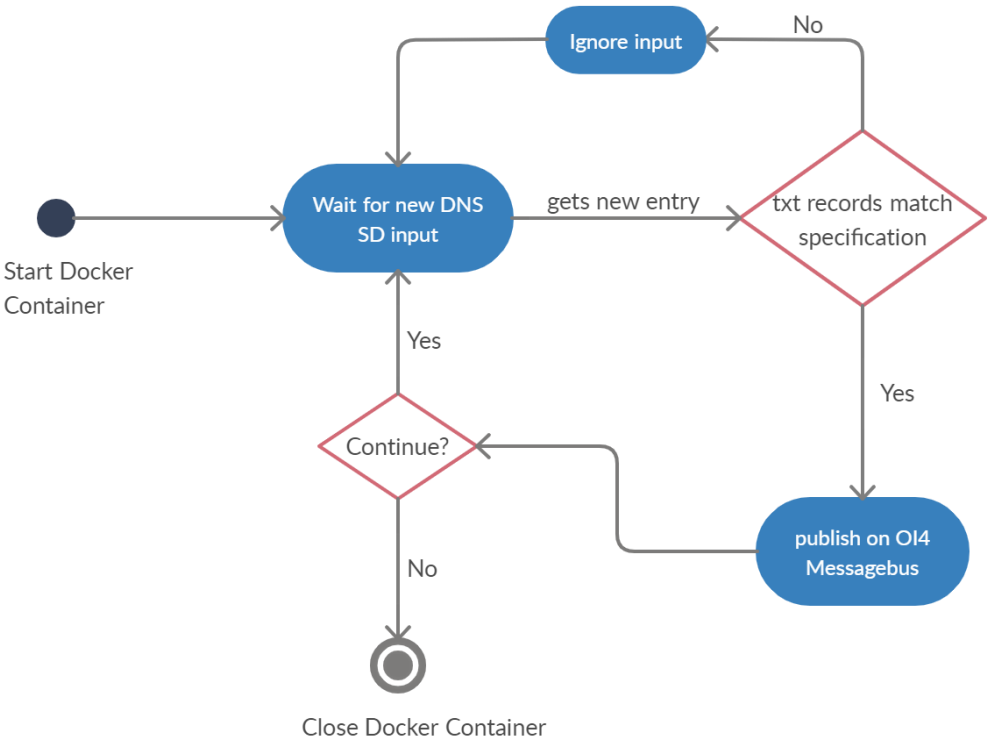

Figure 2.3: <BP.002> Workflow of input handling

3.2. Use Cases

3.2.1. UA-001 Registering Services, that are found using DNS-SD

Related Business Process:	UA-001
Use Cases Objective:	*Registering all available services at the OI4-Service-Registry*
System Boundary:	*OI4-Service-Registry, Services, which announce themselves using DNS-SD, Docker-Application*
Precondition:	*The to be registered services must use TXT-records conform with the specification published by the OI4-Alliance*
Postcondition on success:	*The services will be registered at the OI4-Service-Registry*
Users:	-
Triggering Event:	Any new DNS-SD entries on the network.

3.2.2. UA-002 Registering Services, that are found using DNS-SD

Related Business Process:	UA-001
Use Cases Objective:	*Announcing registered Services on the network*
System Boundary:	*OI4-Service-Registry, Services registered at the OI4-Service-Registry, Docker-Application*
Precondition:	*The services have to be registered at the OI4-Service-Registry*
Postcondition on success:	*The services will be announced to the network using the DNS-SD mechanism.*
Users:	-
Triggering Event:	Any new service registered at the OI4-Service-Registry

3.3. Features

3.3.1. /LF10/ Forwarding Information to the Service-Registry

The Docker Application shall periodically check if there are any new DNS-SD entries and, in case the entry is meant for the Service Registry perform the following reactions: read the DNS-TXT records of the DNS-SD entry, push the received Information to the OI4-MessageBus.

3.3.2. /LF20/ Announcing Services via DNS-SD

The Docker Application shall check if there are any new entries in the Service-Registry and, in case they are not already announced by DNS-SD, perform the following reactions: announce services via DNS-SD.

3.3.3. /LF30/ Announce itself via DNS-SD

The exemplary Docker Application shall check if the user has configured it to announce itself via DNS-SD and perform the following reactions: announce itself via DNS-SD.

3.3.4. /LF40/ Register itself directly at the Service-Registry

The exemplary Docker Application shall check if the user has configured it to register itself directly at the OI4-Service-Registry and perform the following reactions: register itself at the OI4-Service-Register.

3.3.5. /LF50/ Show advertised Services

The exemplary Docker Test Applications shall check if there are any Services registered in the Service Registry and perform the following reactions: show Services on a UI.

3.3.6. /LF60/ Check Conformity

The Docker Application shall check if the received data is conform with the specifications of the OI4-MessageBus.

4. Product Data

4.1. /LD10/ OI4-MessageBus

The OI4-MessageBus is based on the MQTT-Protocol. All Communication via the MessageBus must follow the specifications made by the OI4-Alliance. The OI4-Registry is reached via this interface. Therefore, the MessageBus is used when new DNS-SD entries are discovered and must be added to the Registry but also when the application checks whether there are new Services on the Registry, which might need to be announced via DNS-SD.

4.2. /LD20/ DNS-SD

The DNS-SD system is used to announce Services, that are registered on the Service-Registry, over the network. It is also used to look for other Services, which might not be registered on the Service-Registry. These services are then added to the Service-Registry using the OI4-MessageBus.

5. Other Product Characteristics

This section describes the already known non-functional requirements for the product.

5.1. /NF10/ User-Documentation

The project shall contain an extensive user-documentation. The documentation shall contain instructions on building the Docker-Images and deploying them as Docker-Containers and specifications on how TXT-records are formatted.

5.2. /NF20/ Easy Deployment

An easy deployment shall be guaranteed, as the system is based on Docker-Containers.

5.3. System Environment

This section describes the system environment required to operate the product.
The Project shall be run on any server or computer in a Industry 4.0 environment. In this environment there may be any number of devices, which offer services over a network. These services shall either announce themselves using the DNS-SD mechanism or be directly connected to the OI4-MessageBus running on the MQTT-Broker.

6. References

[1] http://www.ietf.org/rfc/rfc6763.txt
[2] https://blog.stackpath.com/dns-discovery-edge-compute/
[3] https://www.docker.com/
[4] https://en.wikipedia.org/wiki/Edge_computing
https://www.edgelox.com/edgelox-device-service-discovery/