Virtual Coaching Activities for Rehabilitation in Elderly

Call: H2020-SC1-2016-2017

Grant Agreement Number: 769807

Deliverable

D5.3 Coaching services description

Deliverable type:	Other
WP number and title:	WP5: Definition of coaching services
Dissemination level:	Public
Due date:	Month 30 – 28 February 2020
Lead beneficiary:	AIT
Lead author(s):	Johannes Kropf (AIT)
Co-Author(s):	Lukas Roedl (AIT), Emanuel Sandner (AIT), Niklas Hungerländer (AIT), Karl Krainer (AIT), Vito Nitti (IMA)
Reviewers:	Luc Nicolas (EHTEL), Alvaro Martinez (MYS)

This project vCare has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769807.

EXECUTIVE SUMMARY

Building on the requirements defined in WP1 and WP7, the functional description of coaching services in D5.1 and their interfaces described in D5.2, this deliverable provides a comprehensive technical description of the services, associated functionalities, user interaction modalities and interfaces. Starting from a functional description, it also lists and describes the required personal data and details of the integration in the overall architecture. The content of this deliverable is the result of the first phase of tasks 5.3 and 5.4, dealing with the implementation of the coaching and supporting services.

The service-specific architecture shows a seamless integration within the vCare system by leveraging the MQTT protocol and its publish-subscribe based messaging mechanism. This provides a high level of flexibility and extensibility of vCare's core system. The connection of new components or services to the MQTT broker is thus greatly facilitated allowing extended use of functionalities and features. Likewise, the concept of the vCare-as-a-Service paradigm enables third parties to provide additional features to their platform, effectively extending their service portfolio by using the vCare-as-a-Service REST API. This concept is useful for exploiting vCare as it shows its openness to other applications. Most importantly, this document provides a detailed specification and description of the message (event) format together with event topics and event categories, which is a reference for all vCare components developers on how to communicate with other modules through the MQTT broker.

This specification includes a clear description of the events, event topics and event categories, what they are used for and how they look like. Additionally, examples are given on how JSON encoded events will look like within the coaching services layer of the overall architecture. Therefore, this document provides essential information for further software development and integration phase. Finally, an overview of the authentication and authorization mechanism within the vCare system is given, by using the well-established JWT standard. This deliverable is the base for the ongoing development and forms the technical specification of the coaching services in terms of logic, interaction with the user and data exchange with other modules.

CONTENTS

DOCUMENT HISTORY

Version	Date	Author	Description
0.1	15.01.2020	Johannes Kropf	Initial version
0.2	10.02.2020	Johannes Kropf	Generation of service diagrams
0.3	20.02.2020	Johannes Kropf	Completion of message topics
0.4	25.02.2020	Lukas Roedl	Added chapter about internal messaging
0.5	26.02.2020	Johannes Kropf	Added chapter about authorization and authentication
0.6	03.03.2020	Karl Kreiner, Vito Nitti, Patrick Philipp	Delivery of requested input
0.7	04.03.2020	Johannes Kropf, Lukas Roedl	Merged partner inputs, completion of contents
0.8	05.03.2020	George Matikas, Akis Idrizi, Alexandru Vulpe, Razvan Craciunescu	Update authorizaion and authentication, initial version of vCaaS API
0.9	05.03.2020	Johannes Kropf	Compiled pre-final version for internal review
0.91	10.03.2020	Johannes Kropf	Integration of feedback and review comments from FZI, EHTEL and TUD
0.92	12.03.2020	Johannes Kropf	Integration of feedback and review comments MYS and EHTEL
1.0	13.03.2020	Johannes Kropf	Integration of second feedback round, final version

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AAL:	Ambient Assisted Living
CC2U:	CloudCare2U, an IoT health care platform by iSprint
CS-x:	Coaching service number x
HADS:	anxiety and depression questionnaire
HF:	Heart failure
IHD:	Ischaemic heart disease
JSON:	JavaScript Object Notation
JWT:	Json Web Token
KeyCloak:	An authorization broker software
KRF:	Knowledge Representation Framework
MQTT:	A standardized messaging protocol
OAUTH2:	A standardized protocol for Authorization
PAM:	Patient Activation Measure
PD:	Parkinson’s disease
SD:	Stroke disease
TTM:	Trans-theoretical Model
UC-x:	Use case number x
RDF:	Resource description framework
REST:	Representational State Transfer, a paradigm for remote data exchange based on HTTP
Rx-y:	Requirement number x-y
SS-x:	Supporting service number x
RPC:	Remote Procedure Call
vCaaS:	vCare as a Service

1. INTRODUCTION

This document provides a detailed description of how the coaching services are going to be implemented and reflects the work that is performed in Task 5.3 *Development and integration of services to support daily activities* and Task 5.4 *Development and integration of services supporting the clinical rehabilitation pathways* as part of work package 5. The coaching services are the key contribution of vCare towards ICT supported rehabilitation in home environments after the release of an event-induced stationary rehabilitation in a clinic for one of the following indications: heart failure, stroke, Parkinson's disease or ischaemic heart disease. Coaching services within vCare are software modules driven by clinical pathways which provide content and dedicated support for the patient. Those services are meant to induce a quicker recovery, a healthier lifestyle and a reduction of the re-hospitalization rate.

The development of the coaching services is still ongoing and planned to be finished in month 42, but the functional and technical specification and inter-module communication including messages definition and interfaces are finalized and provided in this deliverable.

In more details, this deliverable provides the following specifications and documentation:

- Specification of the inter-module communication including a detailed description of the mechanisms, protocols and data formats
- Specification of the interaction flows between the patient and the coaching services, represented in sequence diagrams
- Specification of the internal logic of the coaching services in the form of activity diagrams
- Comprehensive documentation of how developers of modules can interact with other components of the vCare system
- Comprehensive documentation for third-party developers following the vCare-as-a-service paradigm for being able to connect third-party software to vCare

The list of services discussed in this document was defined fulfilling the requirements defined by the clinical partners in the vCare consortium. This doesn't mean that the functionalities provided in this document can not be extended until the end of the project and beyond. The same holds for the defined messages for data exchange. Hence, this document might be updated at a later stage in the sense of an extension and without losing the validity of what is stated in the current version.

1.1. DOCUMENT SCOPE

Although the title of this document is restricted to coaching services, which are defined as services related to a clinical pathway, supporting services are covered in this document as well. The data services, which are also listed in D5.1 and D5.2 are not specified in detail in this document since there is a separate deliverable foreseen for this purpose (D2.3 *Home context-aware algorithms and resulting metric*).

Thus, this document provides a detailed specification of the coaching and supporting services including a more detailed functional description, interface definition and details about the implementation of each service and the surrounding container. The document is a technical description of the components of the services layer as defined in the overall architecture (D7.4) and is written as a continuation of D5.2. In addition to the service implementation information, this deliverable describes also the interaction models (if applicable) in more detail with the support of sequence diagrams. Being an integral part of the vCare system's functionalities, a description of how the identification and authentication mechanism is realized is also provided in this deliverable.

1.2. METHODOLOGICAL APPROACH

The approach for creating this document, as illustrated in Figure 1, is to create a technical description of the coaching services as defined in D5.1 by updating and extending the contents of D5.2. In D5.1, a functional description of the services have been provided and D5.2 the interfaces of the services have been described. Based on that, D5.3 provides a more technical description of the services and an update of the interface specification of D5.2. In the end, the document provides a detailed technical description of the coaching services for internal use but also for third-parties following the vCare-as-a-service paradigm.

Based on the multi-layered architecture and the high number of components a common, easy-to-use and standardized messaging mechanisms are required. Additionally, a central authentication and authorization broker has to be provided. The choice on MQTT [Oasis, 2020] as an open standard for the common messaging has been made according to the good experience made in previous projects, the availability of open-source servers and client implementation and the fact that it is extremely lightweight. KeyCloak [Keycloack Team] has been chosen to form the central authorization and authentication broker since it supports the most features required in terms of user management and supports the state-of-the-art OAuth2 protocol [Hardt, 2012] ensuring privacy and security. Details are given in Sections 4 and 5.

The services itself are developed partly in Java, partly in Python and are partly provided as a closed solution (the REHABILITY serious gaming solution). AIT has years of experience in developing Java applications and is partly using components previously developed in Java in former projects. Certain components are developed in Python due to the machine learning capabilities Python provides. Due to the messaging mechanism as described before, seamless integration of modules written in different programming languages is guaranteed.

Since this document is strongly related to the actual implementation of the services, relevant parts of this document are automatically extracted and generated from the source code and its inherent documentation. The synchronization of source code and documentation is therefore guaranteed. This also means that every release of the coaching services will generate a new version of this document.

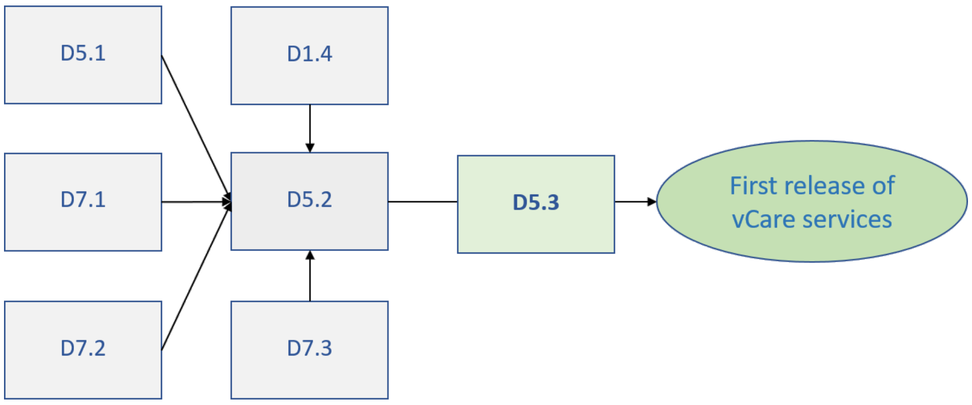


Figure 1. Methodological approach

1.3. DOCUMENT STRUCTURE

The following document is structured as follows: Section 2 describes in detail the concrete implementation of the interactions between the services and the user, specifies the dependencies with other modules and describes the processing of the data flows. In Section 3, the same is provided for the supporting services following the same structure as in Section 2. Section 4 tackles the authentication and authorization concept of vCare using a centralized authentication broker following the OAuth2 standard. Section 5 describes the communication between services and modules in more details. Here, a complete list of all message topics used including their purpose is described. Additionally, examples of the data format of the messages are given. Section 6 concludes the document and gives a brief outlook of future work.

2. COACHING SERVICES

This section describes each coaching service listed in Table 1 with a detailed functional description, interface definition and interaction sequence diagram. The original definition of coaching services can be found in D5.1. For the interactions, the services provide multiple variants of the output texts which are chosen randomly by the system for a more multifaceted speech interaction with the avatar. The following sections describe the separate coaching services.

Table 1. Coaching services overview

#	Name	Description	Requirement	Partner	Implementation	Deployment	Section
CS-1	Physical training	Provides motor exercises giving feedback using a depth camera. Additionally, motivational elements help the user staying engaged (rewards, positive feedback etc.).	R5-8	IMA /AIT	Java / Unity / C#	Local device /cloud	2.3
CS-2	Health status	Provides the user with an overview of his personal health status together with the progress of the rehabilitation treatment.	R5-5	AIT/ MYS	Java	cloud	2.4
CS-3	Cognitive training	The cognitive training service provides gamified cognitive exercises to the patient. Different levels of difficulty and types of exercises are provided as well as motivational elements through rewards.	R5-9	IMA /AIT	Java / Unity / C#	Local device /cloud	2.5
CS-4	E-learning	Provides general information about health and a healthy lifestyle aiming to increase the patient's health literacy.	R5-2	AIT	Java	cloud	2.6
CS-5	Rehabilitation coach	The rehabilitation coach assists the patient with his therapy and tries to motivate and support him to reach his personal goals.	R3-6, R4-6	AIT	Python	cloud	2.7

#	Name	Description	Requirement	Partner	Implementation	Deployment	Section
CS-6	Intelligent notification/ scheduler	Provides information to the user about pathway relevant actions using a context-aware intelligent notification service. The notification can be provided via multiple modalities (UI, avatar, speakers, light etc.).	R2-7, R2-8, R3-1, R3-6, R5-3, R5-4	AIT/ MYS	Java	cloud	2.8
CS-7	Medical questionnaires	Based on validated and standardized questionnaires, data related to the health status of the patient are collected to support the monitoring of health status progression together with the diagnosis of co-morbidities.	R5-7	AIT	Android/ Web technologies	Local/cloud	2.9
CS-8	User feeling	The avatar asks the user about his or her subjective health status (physically and mentally). This is defined in the personalized pathway.	R5-6	AIT	Java	cloud	2.10
CS-9	Speech and swallowing therapy	vCare shall provides digital support for therapy of dysphonia, dysarthria, oropharyngeal dysphagia via serious games.	R5-10	AIT	Java/Unity	local	2.11

2.1. COACHING SERVICES CONTAINER

The coaching services are hosted in a framework that is developed in Java and follows the OSGi (Open Services Gateway initiative [OSGi Alliance, 2013] specification. This framework is based on Apache Karaf [Apache Soft. Found.], an open-source project provided by the Apache Foundation. The advantage of OSGi, in this case, is the modularization of the whole application into several smaller bundles which can provide services to each other. The OSGi specification describes a service platform that provides a complete and dynamic component model, which allows separate modules to be installed, started, stopped, updated and uninstalled at runtime without requiring a reboot. A service registry allows bundles to detect the addition of new services, or the removal of services, and adapt accordingly.

Within vCare this means that the separate coaching services are implemented as separate bundles that use common modules also running in the same framework via this service registry. If some bundles rely on functionality provided by another bundle they are notified when this functionality is available and can react accordingly. A more dynamic and event-oriented approach is offered by the OSGi Event Admin specification [OSGi Alliance a] which describes a method to send events around in the framework and bundles that are related to special kinds of events are able to register for those events.

As mentioned earlier, the vCare Coaching Services Container runs in the Apache Karaf

Framework which sits on top of the OSGi container. Apache Karaf itself adds additional functionality to OSGi via add-on features. One of those additional features consists in the control of the container through the direct shell, SSH or a web console. It supports provisioning and deployment via file system, Maven repository [Zyl & Fox, 2020], archive (e.g. tar.gz), OBR (OSGi Bundle Repository [Sonatype]) and configuration via the file system, web console and JMX [OSGi Alliance b]. Also, dependency injection using different techniques (like SCR or Blueprint) is available in Apache Karaf.

For further information about Apache Karaf please have a look at [Apache Soft. Found.] and [Edstrom et al, 2013].

2.2. HOW TO READ THE SERVICES DESCRIPTIONS

In the following sections, technical descriptions of the coaching services (Section 2) and supporting services (Section 3) are provided. All descriptions follow the same structure: after a general description of the service, the interaction design describing how the user interacts with the service and how the service interacts with other components of the vCare system is explained.

Then, in the *Speech Interaction definition* sections, example dialogs between avatar and system are given. These examples will be extended by additional options what to say and what the avatar responds, but the structure of the dialogs will remain the same.

In the *requirements on personal data* sections, the data which are necessary to provide the service is provided.

In the following section *Implementation*, details about how this service is implemented is given, including an activity diagram depicting the internal logic of the service.

In the last section *Interface description*, the message topics which are consumed and published by the service are stated. For the inter-module communication within vCare, messages will be used following a publish-subscribe mechanism. Modules which are running within the services container communicate over the OSGi EventAdmin which is described in more detail in Section 5, communication with other components is realized using a centralized message broker (the MQTT server). Services or components register at the central message broker (the MQTT server or the OSGi EventAdmin) with so-called topics they would like to consume. These topics are listed in this section and are relevant for developers who want to use the service.

Even though two brokers are used, for the services and other components this makes no difference since they are connected and the messages remain the same.

2.3. CS-1: PHYSICAL TRAINING

2.3.1. GENERAL DESCRIPTION

The service includes two elements: the first one is a dedicated vCare service responsible for the initiation of physical training sessions and the second one is a commercial product deployed by the project partner IMA. The IMA product is called REHABILITY and is a rehabilitation system based on motion tracking through 3D depth sensors, providing game-based personalized exercises with immediate feedback to the user and data collection for health care professionals.

The serious games are triggered either by the user or by the avatar via a reminder. The physical exercise is then invoked automatically after the user has switched on the TV set and the Set-top box connected to it. The REHABILITY solution is described in more details already in deliverable D5.1 - *Definition of Coaching Services*. Some physical exercises cover outdoor activities and are therefore not handled by the REHABILITY system. A specific game plan is available after configuration on the caregiver-interface and can be retrieved through the event topics as defined below.

2.3.2. INTERACTION DESIGN

The service is triggered by the system according to the schedule of the patient's pathway. The interaction with the user and the message exchanges with other components is depicted in Figure 2. The service does not directly provide the training. Instead, it just reminds the user to turn on the REHABILITY system which is a closed solution providing the serious games running on a separate device.

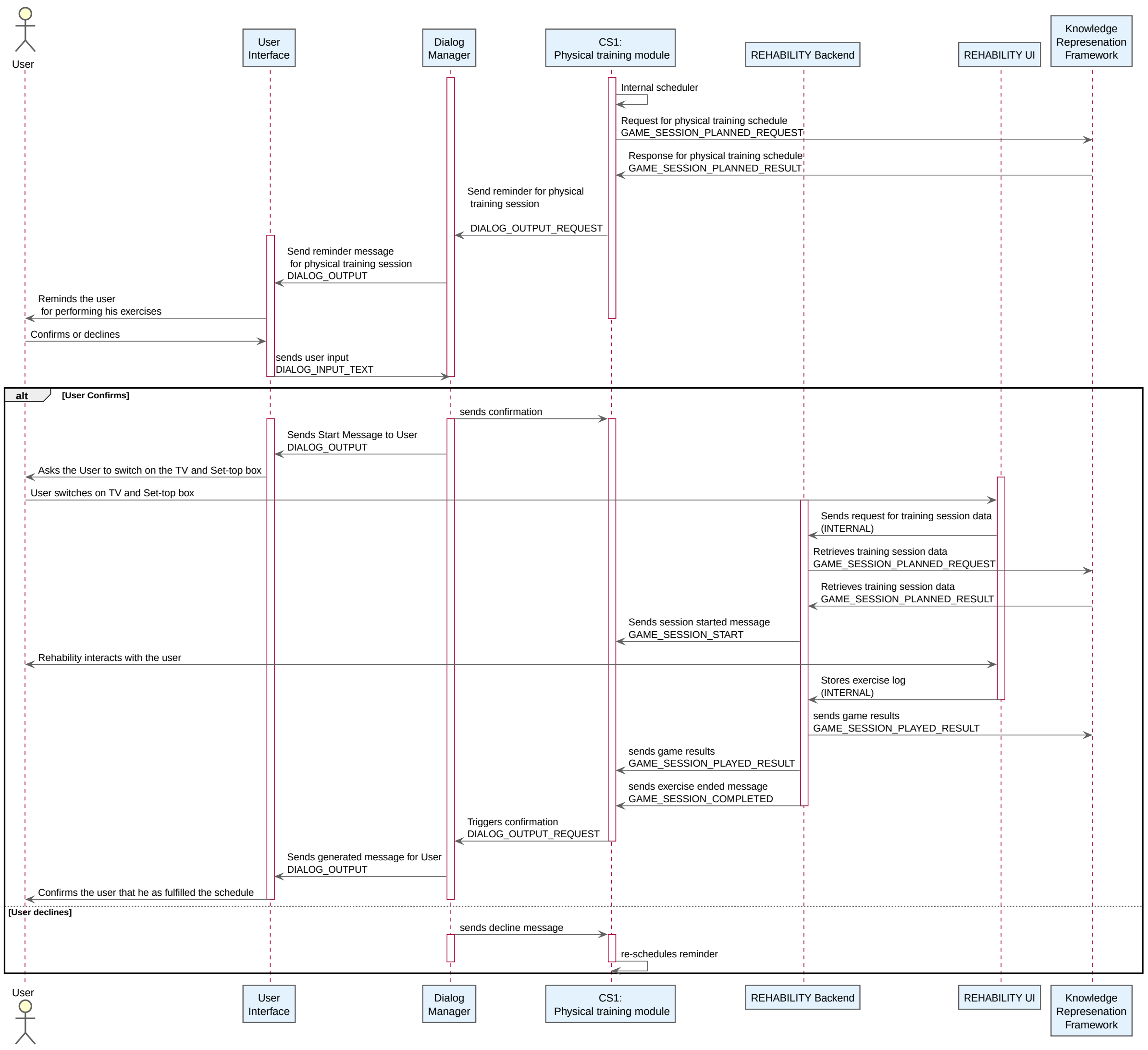

User
User Interface
Dialog Manager
CS1: Physical training module
REHABILITY Backend
REHABILITY UI
Knowledge Represenation Framework
Internal scheduler
Request for physical training schedule GAME_SESSION_PLANNED_REQUEST
Response for physical training schedule GAME_SESSION_PLANNED_RESULT
Send reminder for physical training session
DIALOG_OUTPUT_REQUEST
Send reminder message for physical training session DIALOG_OUTPUT
Reminds the user for performing his exercises
Confirms or declines
sends user input DIALOG_INPUT_TEXT
alt
[User Confirms]
sends confirmation
Sends Start Message to User DIALOG_OUTPUT
Asks the User to switch on the TV and Set-top box
User switches on TV and Set-top box
Sends request for training session data (INTERNAL)
Retrieves training session data GAME_SESSION_PLANNED_REQUEST
Retrieves training session data GAME_SESSION_PLANNED_RESULT
Sends session started message GAME_SESSION_START
Rehability interacts with the user
Stores exercise log (INTERNAL)
sends game results GAME_SESSION_PLAYED_RESULT
sends game results GAME_SESSION_PLAYED_RESULT
sends exercise ended message GAME_SESSION_COMPLETED
Triggers confirmation DIALOG_OUTPUT_REQUEST
Sends generated message for User DIALOG_OUTPUT
Confirms the user that he as fulfilled the schedule
[User declines]
sends decline message
re-schedules reminder

Figure 2. Data flows and interaction schema for CS-1.

2.3.3. SPEECH INTERACTION DEFINITION

In the REHABILITY software, speech output is used as well. This is indicated as REHABILITY coach in the following and is technically and logically independent of the virtual coach.

Interaction when user accepts the training

Virtual Coach: "You haven't performed your physical exercise for today yet. Do you want to do the exercises right now?"
User: "Yes, please start".
Virtual Coach: "Ok, please switch on your TV and the Set-top box to start the exercises"
REHABILITY coach: "Welcome back John!"
User: *(starts the game)*
REHABILITY coach: *(explains game instructions)* "Move the chosen shape to the correct colour touching as many red dots as possible."
User: *(plays)*
User: *(user sits while playing - not supposed to do that)*
REHABILITY coach: "Please stand up while playing"
User: *(game ends)*
REHABILITY coach: "NEW HIGHSCORE! *(applause)*" REHABILITY coach: "Session completed." Virtual coach: "Well done!"

Interaction when user declines the training

Virtual Coach: "Today physical training is on your agenda; shall I launch physical games for you?"
User: "No, later please"
Virtual Coach: *postpones the request and asks again later. ("later" is derived from the time the user usually performs the cognitive training)*

Interaction when the training has ended

Virtual Coach: "You have finished your training for today."
Virtual Coach: "You have performed very well today." | "Seemed you had some difficulties with your training today. Let's see if it will get better next time."

Interaction for outdoor activities

Virtual Coach: "You have scheduled an outdoor exercise for today. Your task is to walk fast for 30min with a target heart rate of 100 beats per minute. Please use your heart rate monitor when performing the exercise."

The exact exercise will be retrieved from the KRF including the duration and target heart rate.

2.3.4. REQUIREMENTS ON PERSONAL DATA

The following list of data elements are required to retrieve from the KRF:

- User id

- Type of exercise
- Parameters for the exercise
- Date and time when the exercise is planned

2.3.5. IMPLEMENTATION

The service consists of three major components:

1) The Physical Training coaching service, implemented in Java as OSGi bundle running inside the vCare Coaching Services Container and responsible for the communication with the avatar
2) the REHABILITY backend, the training and game repository and manager for the training itself
3) the REHABILITY UI running on a separate device (Android-based Set-Top Box) connected to the user's TV.

The diagram below depicts the internal logic of the service triggering the rehabilitation games (Figure 3).

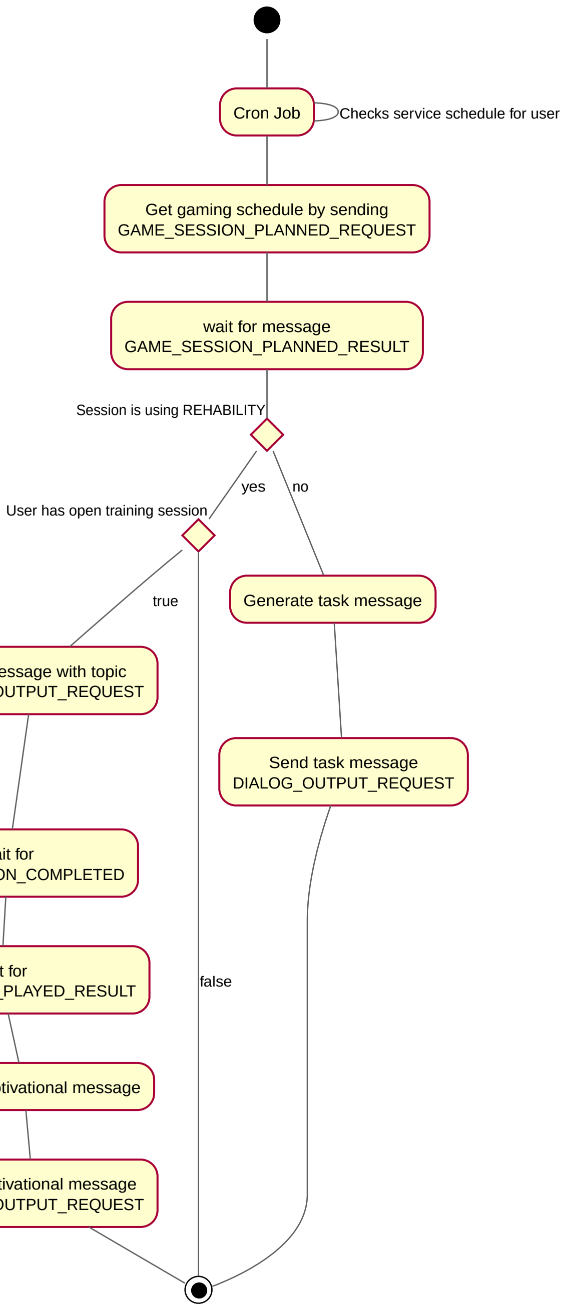

Cron Job
Checks service schedule for user
Get gaming schedule by sending GAME_SESSION_PLANNED_REQUEST
wait for message GAME_SESSION_PLANNED_RESULT
Session is using REHABILITY
yes
no
User has open training session
true
Generate task message
Send message with topic DIALOG_OUTPUT_REQUEST
Send task message DIALOG_OUTPUT_REQUEST
wait for GAME_SESSION_COMPLETED
wait for GAME_SESSION_PLAYED_RESULT
false
generate motivational message
Send motivational message DIALOG_OUTPUT_REQUEST

Figure 3. Activity diagram for CS-1.

2.3.6. INTERFACE DESCRIPTION

The whole communication with other modules is implemented via OSGi EventAdmin, the connected MQTT broker and its related JSON messages. The modules (the OSGi coaching service) consumes the topics

- GAME_SESSION_PLANNED_RESULT
- GAME_SESSION_COMPLETED
- GAME_SESSION_PLAYED_RESULT

and publishes on topics

- GAME_SESSION_PLANNED_REQUEST
- DIALOG_OUTPUT_REQUEST

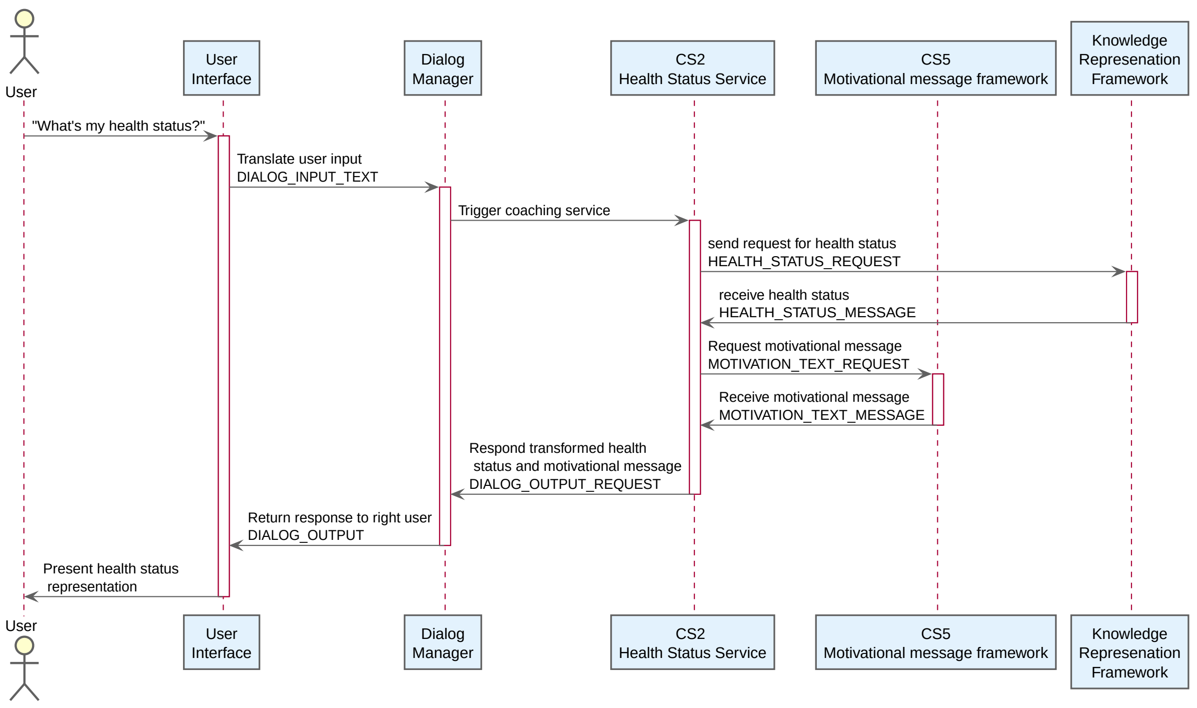
2.4. CS-2: HEALTH STATUS

2.4.1. GENERAL DESCRIPTION

The user shall be informed about his/her current health status and progress in the individual rehabilitation pathway. Aggregated health data are presented to the user under the form of numbers, graphs and speech output. Also, deviations from normal values are also shown. The deviations are determined in an underlying service within the knowledge layer.

2.4.2. INTERACTION DESIGN

The service is triggered by the user by asking the avatar for his or her health status. The request is recognized by the dialogue manager and forwarded to the coaching service. The interaction and the data flow with other modules are depicted in Figure 4 below.



User
User Interface
Dialog Manager
CS2 Health Status Service
CS5 Motivational message framework
Knowledge Representation Framework
"What's my health status?"
Translate user input DIALOG_INPUT_TEXT
Trigger coaching service
send request for health status HEALTH_STATUS_REQUEST
receive health status HEALTH_STATUS_MESSAGE
Request motivational message MOTIVATION_TEXT_REQUEST
Receive motivational message MOTIVATION_TEXT_MESSAGE
Respond transformed health status and motivational message DIALOG_OUTPUT_REQUEST
Return response to right user DIALOG_OUTPUT
Present health status representation

Figure 4. Data flows and interaction schema for CS-2.

2.4.3. SPEECH INTERACTION DEFINITION

User: "Show me my health status" | "Give me my health data" | "what are my last measurements?"
Virtual Coach: "Your blood pressure is normal today, your physical activity index increased by 3% during the last week".

2.4.4. REQUIREMENTS ON PERSONAL DATA

The following list of data elements are required to retrieve from the KRF:

- User id
- List of relevant health parameters to be shown to the user
- Information how these parameters have to be interpreted in three levels (very bad, bad, normal)
- The trend of the provided parameters in three levels (positive, negative, steady)

2.4.5. IMPLEMENTATION

The module is implemented in Java as an OSGi Bundle running inside the vCare Coaching Services Container. The diagram below depicts the internal logic of the service generating messages referring to the health status of the user (Figure 5).

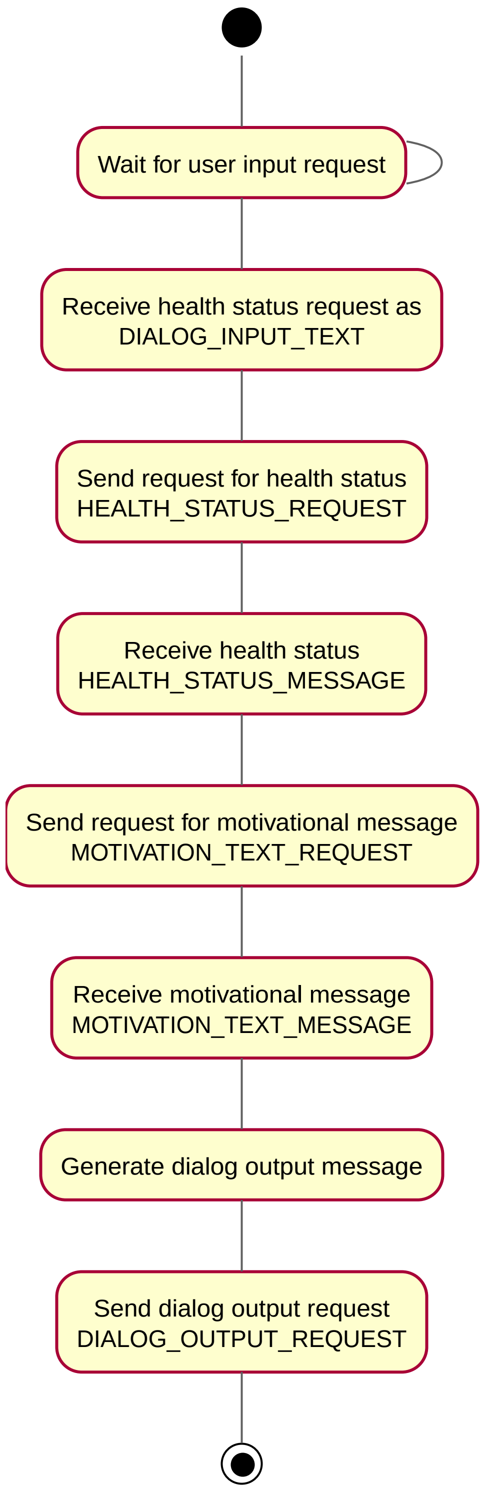

Wait for user input request
Receive health status request as DIALOG_INPUT_TEXT
Send request for health status HEALTH_STATUS_REQUEST
Receive health status HEALTH_STATUS_MESSAGE
Send request for motivational message MOTIVATION_TEXT_REQUEST
Receive motivational message MOTIVATION_TEXT_MESSAGE
Generate dialog output message
Send dialog output request DIALOG_OUTPUT_REQUEST

Figure 5. Activity diagram for CS-2.

2.4.6. INTERFACE DESCRIPTION

The whole communication with other modules is implemented via OSGi EventAdmin, the connected MQTT broker and its related JSON messages. The module consumes the topics

- DIALOG_INPUT_TEXT
- HEALTH_STATUS_MESSAGE
- MOTIVATIONAL_TEXT_MESSAGE

and publishes on topics

- HEALTH_STATUS_REQUEST
- DIALOG_OUTPUT_REQUEST
- MOTIVATIONAL_TEXT_REQUEST

2.5. CS-3: COGNITIVE TRAINING

2.5.1. GENERAL DESCRIPTION

The service is very similar to CS1 - *Physical training* and uses the same game repository. It uses the REHABILITY system which is based on motion tracking through 3D depth sensors, providing game-based personalized exercises with immediate feedback to the user and data collection for professionals. The serious games are triggered either by the user or by the avatar via a reminder. Cognitive exercise is then invoked automatically. A specific game plan is available after configuration through the caregiver-interface and can be retrieved through the event topics as defined below.

2.5.2. INTERACTION DESIGN

The service is triggered by the system according to the schedule of the patient's pathway. The interaction with the user and the message exchanges with other components are depicted in Figure 6. The service does not directly provide the training. Instead, it just reminds the user to turn on the REHABILITY system which is a closed solution providing the serious games running on a separate device.

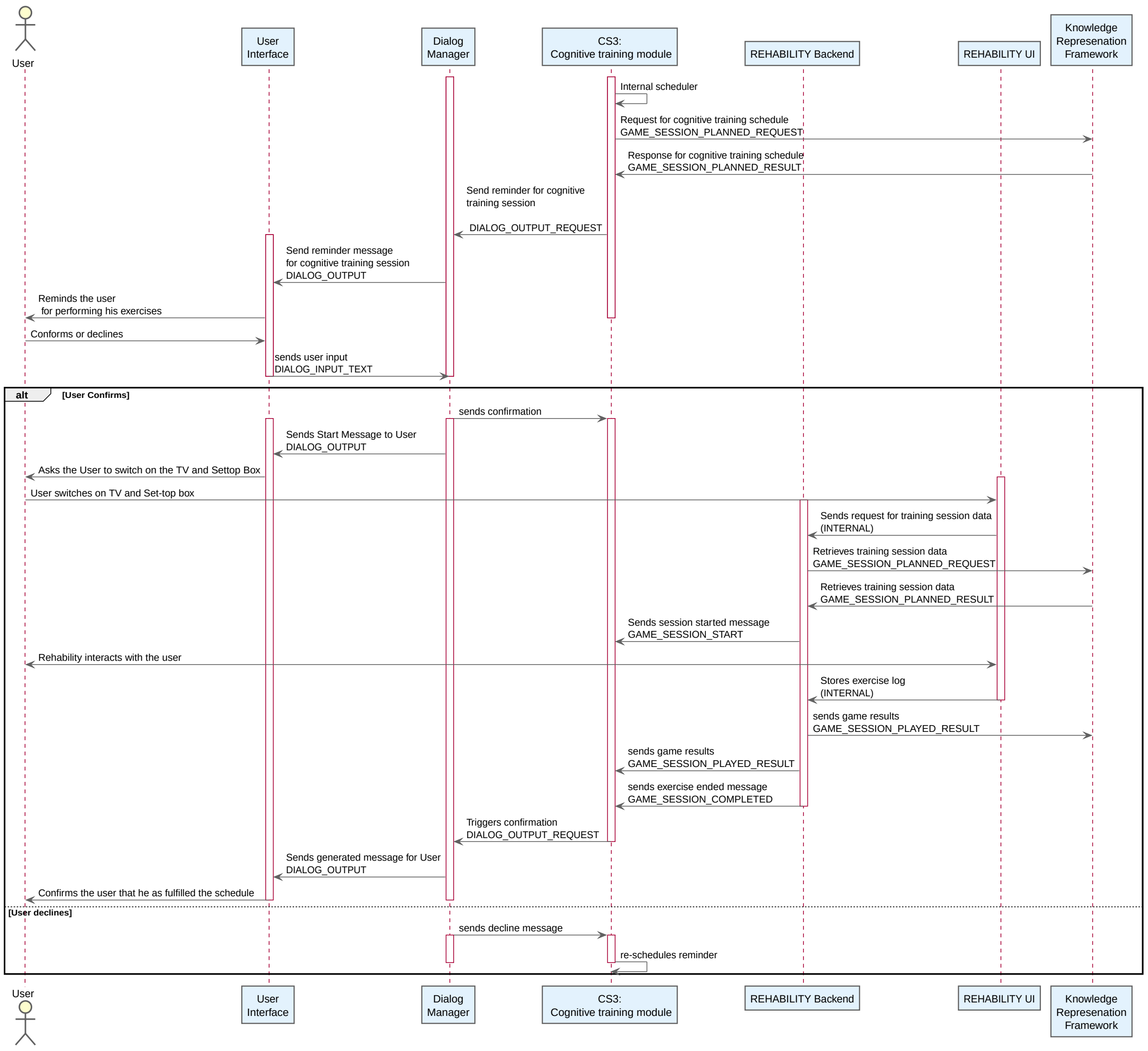

User
User Interface
Dialog Manager
CS3: Cognitive training module
REHABILITY Backend
REHABILITY UI
Knowledge Representation Framework
Internal scheduler
Request for cognitive training schedule GAME_SESSION_PLANNED_REQUEST
Response for cognitive training schedule GAME_SESSION_PLANNED_RESULT
Send reminder for cognitive training session
DIALOG_OUTPUT_REQUEST
Send reminder message for cognitive training session DIALOG_OUTPUT
Reminds the user for performing his exercises
Conforms or declines
sends user input DIALOG_INPUT_TEXT
alt
[User Confirms]
sends confirmation
Sends Start Message to User DIALOG_OUTPUT
Asks the User to switch on the TV and Settop Box
User switches on TV and Set-top box
Sends request for training session data (INTERNAL)
Retrieves training session data GAME_SESSION_PLANNED_REQUEST
Retrieves training session data GAME_SESSION_PLANNED_RESULT
Sends session started message GAME_SESSION_START
Rehability interacts with the user
Stores exercise log (INTERNAL)
sends game results GAME_SESSION_PLAYED_RESULT
sends game results GAME_SESSION_PLAYED_RESULT
sends exercise ended message GAME_SESSION_COMPLETED
Triggers confirmation DIALOG_OUTPUT_REQUEST
Sends generated message for User DIALOG_OUTPUT
Confirms the user that he as fulfilled the schedule
[User declines]
sends decline message
re-schedules reminder

Figure 6. Data flows and interaction schema for CS-3.

2.5.3. SPEECH INTERACTION DEFINITION

Interaction when user accepts to do the training

Virtual Coach: "You haven't performed your cognitive exercise today yet. Do you want to do the exercises right now?"
User: "Yes".
Virtual Coach: "Ok, please switch on your TV to start your exercises."

Interaction when user declines the training

Virtual Coach: "Today cognitive training is on your agenda; shall I launch cognitive games for you?"
User: "No, later please"
Virtual Coach: *postpones the request and asks again later. ("later" is derived from the time the user usually performs the cognitive training)*

Interaction when the training has ended

Virtual Coach: "You have finished your training for today."
Virtual Coach: "You have performed very well today." | "Seemed you had some difficulties with your training today. Let's see if it will get better next time."

2.5.4. REQUIREMENTS ON PERSONAL DATA

The following list of data elements are required to retrieve from the KRF:

OSGi service

- User id

REHABILITY backend

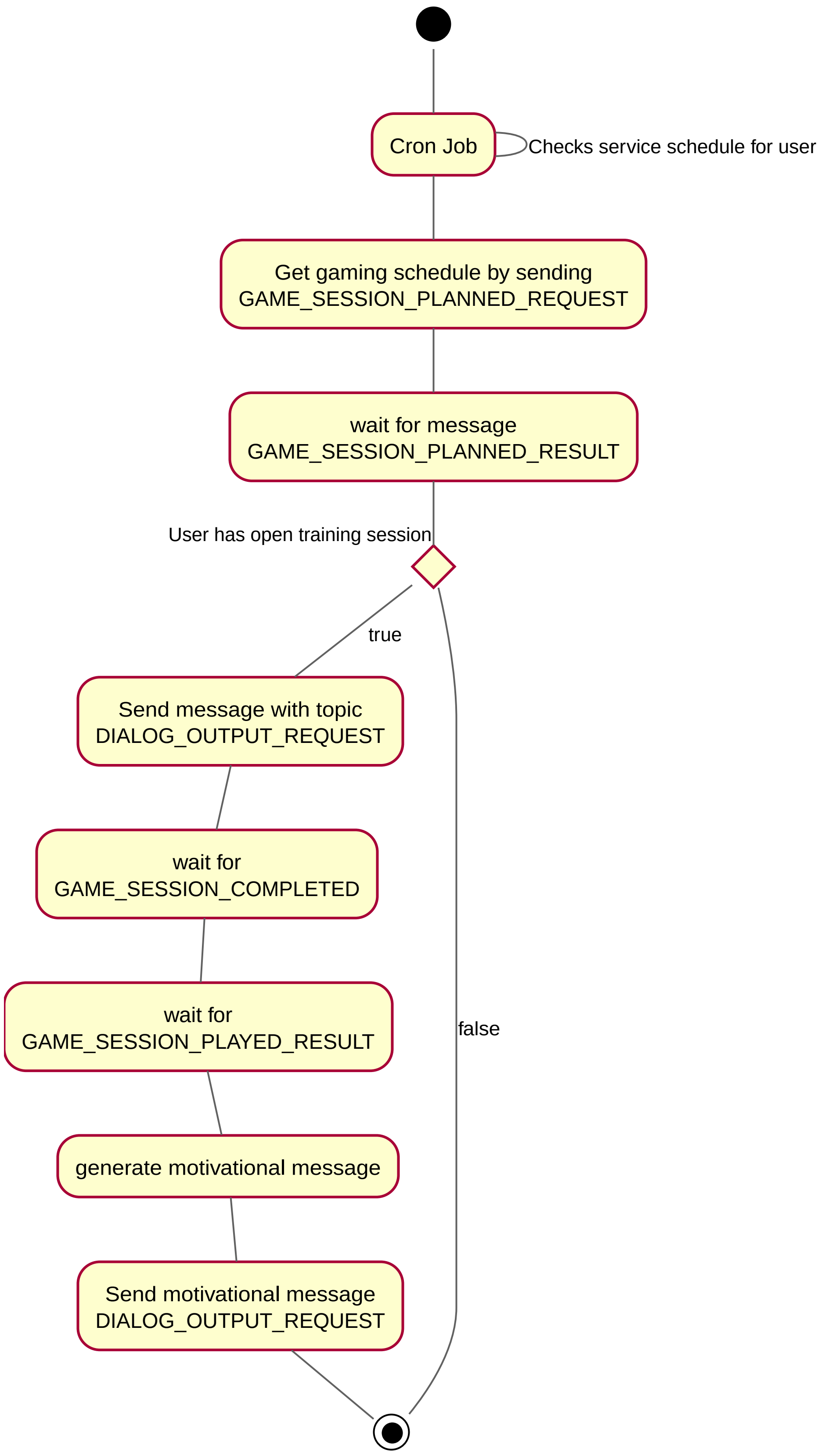
- User id
- List of exercises to be performed
- Parameters for the exercise

2.5.5. IMPLEMENTATION

The service makes use of three major components:

1) The Cognitive coaching service, implemented in Java as OSGi bundle running inside the vCare Coaching Services Container and responsible for the communication with the avatar
2) the REHABILITY backend, the training and game repository and manager for the training itself
3) the REHABILITY UI running on a separate device (Android-based Set-Top Box) connected to the user's TV.

The diagram below depicts the internal logic of the service triggering the rehabilitation games (Figure 7).



Cron Job
Checks service schedule for user
Get gaming schedule by sending GAME_SESSION_PLANNED_REQUEST
wait for message GAME_SESSION_PLANNED_RESULT
User has open training session
true
false
Send message with topic DIALOG_OUTPUT_REQUEST
wait for GAME_SESSION_COMPLETED
wait for GAME_SESSION_PLAYED_RESULT
generate motivational message
Send motivational message DIALOG_OUTPUT_REQUEST

Figure 7. Activity diagram for CS-3.

2.5.6. INTERFACE DESCRIPTION

The whole communication with other modules is implemented via OSGi EventAdmin, the connected MQTT broker and its related JSON messages. The modules (the OSGi coaching service) consumes the topics

- GAME_SESSION_PLANNED_RESULT
- GAME_SESSION_COMPLETED
- GAME_SESSION_PLAYED_RESULT

and publishes on topics

- GAME_SESSION_PLANNED_REQUEST
- DIALOG_OUTPUT_REQUEST

2.6. CS-4: E-LEARNING

2.6.1. GENERAL DESCRIPTION

To increase the patient's health literacy, e-learning components related to a healthy lifestyle and the user's rehabilitation pathway are provided to the user. The content is provided by videos explaining particular aspects related to the patient's disease and therapy, but also about a healthy lifestyle in general. At this stage, a number of content for videos for the different pathologies considered are defined and listed in Table 2. This list is not final and might be extended during the remaining project phase.

Table 2. Defined e-learning content to be provided for the different pathways/pathologies.

ID	Purpose	Pathology	Content description	Partner / owner
RF_FALLING_1	Health literacy	SD	Multi-medial lesson concerning the risk of falling at home	CCP
PT_UPPER_LIMBS	Health literacy	SD	Physical activity for the upper limbs	CCP
HL_PD	Health literacy	PD	Multi-medial lesson concerning health in PD	OSA
RF_FALLING_2	Health literacy	PD	Multi-medial lesson concerning the risk of falling at home	CCP
HL_HF	Health literacy	HF	Content related to cardiovascular disease prevention: heart failure description, risk factors control in heart failure, physical activities role in heart disease, alcohol consumption, role of medication, sexual activity in heart failure.	UMFCD

HL_IHD	Health literacy	IHD	Video lesson with heart-healthy diet instructions and with arguments of necessity for conducting aerobic and muscle-strengthening exercises	HEV
RF_NUTRITION_WEIGHT	Weight control	HF	Information about correct nutrition	UMFCD
RF_SMOKING_HF	Smoking cessation	HF	Providing information related to the importance of correcting smoking habits	UMFCD
RF_SMOKING_IHD	Smoking cessation	IHD	Video lesson with smoking cessation instructions	HEV
RF_NUTRITION_ALCOHOL	Alcohol reduction	IHD	Video lesson with alcohol reduction instructions	HEV

2.6.2. INTERACTION DESIGN

The service retrieves the schedule provided by the KRF from the patient’s pathway and informs the user. The KRF not only provides information of which content shall be presented to the patient, but also meta-information about it, e.g. introductions to the content, link to the video etc. The schedule and selection of videos for a patient is configured by the care professionals in the caregiver UI. The interaction with the patient as well as the data exchange with other modules is depicted in Figure 8.

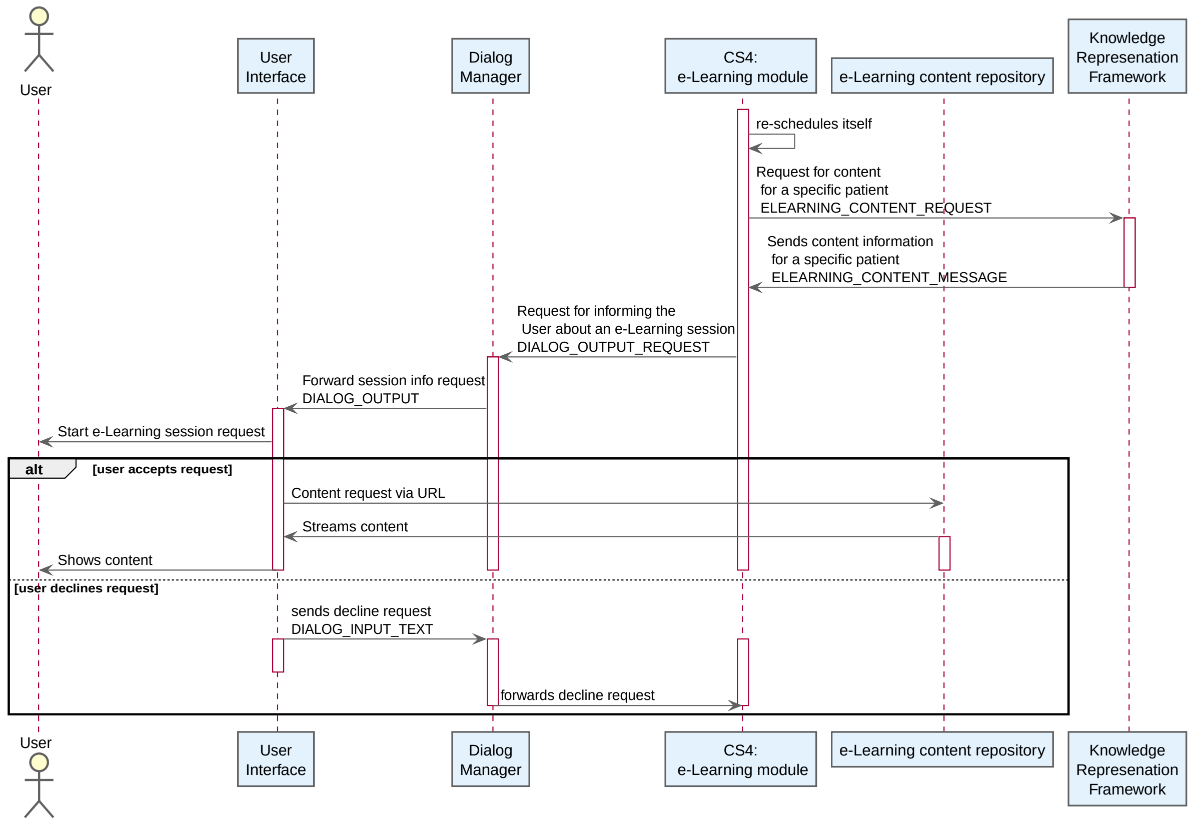

User
User Interface
Dialog Manager
CS4: e-Learning module
e-Learning content repository
Knowledge Representation Framework
re-schedules itself
Request for content for a specific patient ELEARNING_CONTENT_REQUEST
Sends content information for a specific patient ELEARNING_CONTENT_MESSAGE
Request for informing the User about an e-Learning session DIALOG_OUTPUT_REQUEST
Forward session info request DIALOG_OUTPUT
Start e-Learning session request
alt
[user accepts request]
Content request via URL
Streams content
Shows content
[user declines request]
sends decline request DIALOG_INPUT_TEXT
forwards decline request

Figure 8. Data flows and interaction schema for CS-4.

2.6.3. SPEECH INTERACTION DEFINITION

Virtual Coach: "I have an educational session for you today. Shall I start the video about <VIDEO_CONTENT> now?" User: "Yes" | "No"

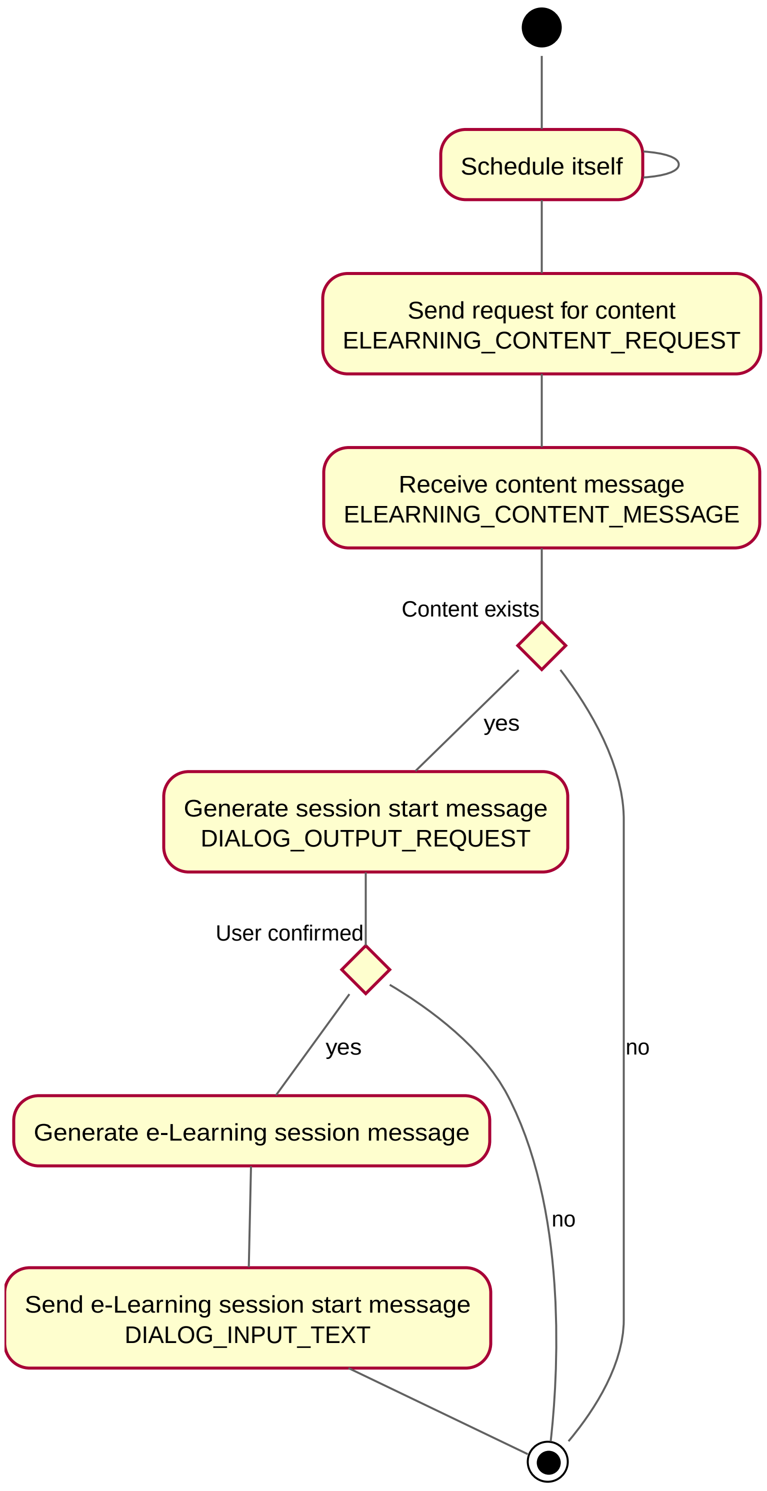
2.6.4. REQUIREMENTS ON PERSONAL DATA

The following list of data elements are required to retrieve from the KRF:

- User id
- List of the material to be presented to the user
- Date and time when the e-Learning session is planned

2.6.5. IMPLEMENTATION

The module is implemented in Java as an OSGi Bundle running inside the vCare Coaching Services Container. The diagram below depicts the internal logic of the service generating e-learning session requests for the user (Figure 9).



Schedule itself
Send request for content ELEARNING_CONTENT_REQUEST
Receive content message ELEARNING_CONTENT_MESSAGE
Content exists
yes
no
Generate session start message DIALOG_OUTPUT_REQUEST
User confirmed
yes
no
Generate e-Learning session message
Send e-Learning session start message DIALOG_INPUT_TEXT

Figure 9. Activity diagram for CS-4.

2.6.6. INTERFACE DESCRIPTION

The whole communication with other modules is implemented via OSGi EventAdmin, the connected MQTT broker and its related JSON messages. The module consumes the topics

- DIALOG_INPUT_TEXTS
- ELEARNING_CONTENT_MESSAGE
- DIALOG_INPUT_TEXT

and publishes on topics

- ELEARNING_CONTENT_REQUEST
- DIALOG_OUTPUT_REQUEST

2.7. CS-5: REHABILITATION COACH

2.7.1. GENERAL DESCRIPTION

This service aims at assisting the patient through his/her rehabilitation process by providing recommendations for a lifestyle which is supporting the rehabilitation plan. The detailed information to be provided is defined in the patient's personal pathway. Other than depicted in D5.1 figure 6, the rehabilitation coach plays a more central role: Instead of being a subdomain of the Information, Motivation, Behavior model (IMB) model and only be triggered dependent on the Patient Activation Measure (PAM, is described in more detail in D1.4 and D5.1.) score, it seemed more reasonable to trigger the rehabilitation coach dependent on the coaching services or the Knowledge Representation Framework (KRF). However, the PAM, as well as the IMB model, are now integrated in the rehabilitation coach directly. Dependent on the PAM score the rehabilitation coach acts differently. When having a low PAM score the rehabilitation coach responds with a message with more information on a topic as well as sends a request to the KRF to trigger the e-learning service. This represents the information dimension of the IMB. When having a high PAM score the rehabilitation coach sends a message including persuasive strategies and more information on how to achieve the goal. This represents the Motivation and Behavior dimensions of the IMB.

2.7.2. INTERACTION DESIGN

The service is triggered by the system, mainly by the KRF, but also by other coaching services. Based on historical data and the information in the pathway, the KRF triggers the rehabilitation coach whenever the pathway has changed by the reasoner. The Rehabilitation coach generates motivational messages based on the reasons which led to the change of the pathway. Hence, the reasons reflect the performance of the user according to the exercises defined in the pathway (lower part of Figure 10). The service also generates context-related motivational messages for other services upon request (upper part of Figure 10). The rehabilitation coach has a knowledge base which includes motivational messages for both the coaching services which can trigger the rehabilitation coach and the KRF. Messages are grouped into "positive", "negative" and "neutral" as well as into "information topic" and "information goal". The messages labelled with "information goal" also contain persuasive strategies. The rehabilitation coach chooses the message based on the information it receives from the coaching services/KRF as well as on the patient activation measure (PAM).

Examples for messages triggered by the KRF: A user with a low PAM score who did not meet his physical training goal receives a message with a negative sentiment, containing information about the topic (e.g. Start working out before its too late! Regular exercise will make you less vulnerable for diseases.). A user with a high PAM score who met his physical training goal receives a message with a negative sentiment, containing information on how to achieve the goal (e.g. Start working out before its too late! Make exercise your hobby! If you have fun while doing it, it will get easier to start.). A user met his physical activity goal he receives a message with a positive sentiment (e.g. Wow, you are doing very well! Hold on!).

Examples for messages triggered by the physical training service: A user who did well on the current training will receive a message with a positive sentiment(e.g. You have performed very well today.). A user who did not well on the current training will receive a message with a neutral sentiment(e.g. Seemed you had some difficulties with your training today.I am sure that you can do it better next time.).

In summary, each message in the knowledge base has information about its sentiment, PAM value and context.

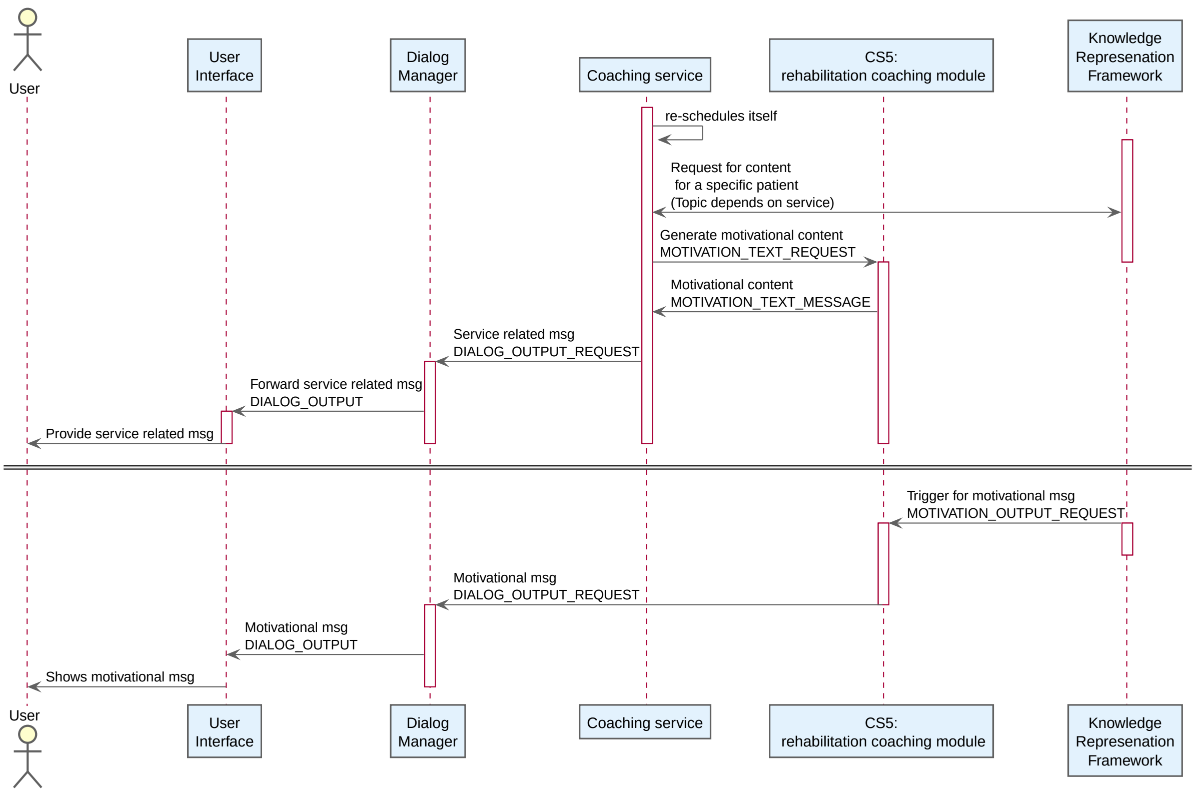

User
User Interface
Dialog Manager
Coaching service
CS5: rehabilitation coaching module
Knowledge Represenation Framework
re-schedules itself
Request for content for a specific patient (Topic depends on service)
Generate motivational content MOTIVATION_TEXT_REQUEST
Motivational content MOTIVATION_TEXT_MESSAGE
Service related msg DIALOG_OUTPUT_REQUEST
Forward service related msg DIALOG_OUTPUT
Provide service related msg
Trigger for motivational msg MOTIVATION_OUTPUT_REQUEST
Motivational msg DIALOG_OUTPUT_REQUEST
Motivational msg DIALOG_OUTPUT
Shows motivational msg

Figure 10. Data flows and interaction schema for CS-5.

2.7.3. SPEECH INTERACTION DEFINITION

Contextual recommendation

Virtual Coach states: "Physical exercise can lead to mental relaxation. You didnâ€™t do for a while. I would suggest going for a walk." User can respond: "Ok, thank you."

Health-related recommendation

Virtual Coach states: "Your blood pressure was a bit high during the last weeks. Be aware that a normal blood pressure is essential for a healthy lifestyle". User can respond: "Ok, thank you."

2.7.4. REQUIREMENTS ON PERSONAL DATA

The following list of data elements are required to retrieve from the KRF:

- User id
- Context information
- The sentiment (positive, negative, neutral)
- PAM level

The following list of data elements are required to retrieve from other coaching services:

- User id
- The sentiment (positive, negative, neutral)
- PAM level

2.7.5. IMPLEMENTATION

The module is implemented in Python and runs in parallel to the vCare Coaching Services Container. It is connected to the other components of the vCare system via the MQTT message protocol and connects itself to the MQTT broker. Dependent on the sentiment, the module will choose a primary intention for the message which can be either discouraging, encouraging or neutral. Dependent on the level derived from the PAM questionnaire, the module will choose a secondary intention which could be either feedback, information, suggestion or reinforcement. For every context, suitable messages are stored in the database of the module.

The diagram below depicts the internal logic of the service generating e-learning session requests for the user (Figure 11).

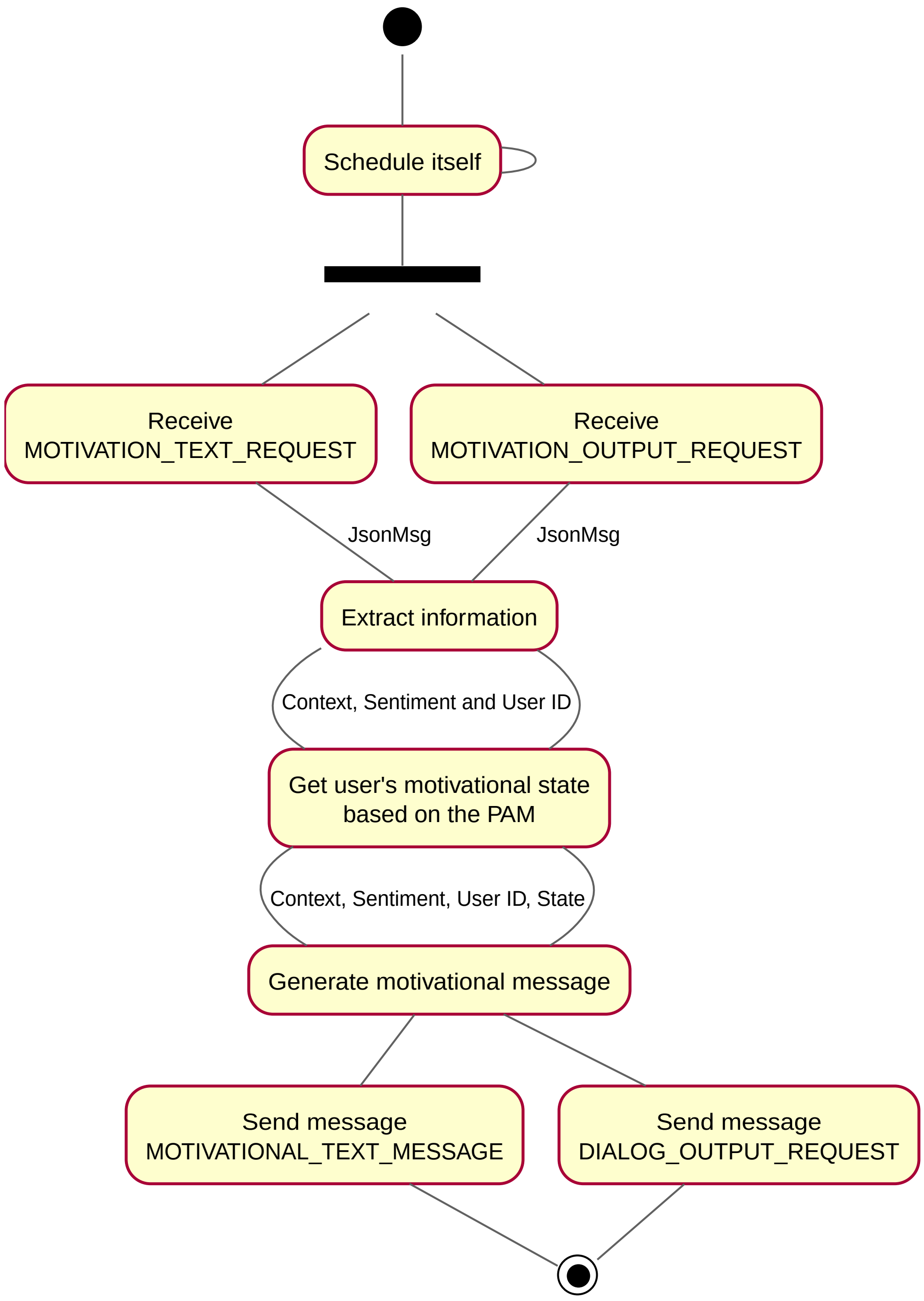

Schedule itself
Receive MOTIVATION_TEXT_REQUEST
Receive MOTIVATION_OUTPUT_REQUEST
JsonMsg
JsonMsg
Extract information
Context, Sentiment and User ID
Get user's motivational state based on the PAM
Context, Sentiment, User ID, State
Generate motivational message
Send message MOTIVATIONAL_TEXT_MESSAGE
Send message DIALOG_OUTPUT_REQUEST

Figure 11. Activity diagram for CS-5.

2.7.6. INTERFACE DESCRIPTION

The whole communication with other modules is implemented via OSGi EventAdmin, the connected MQTT broker and its related JSON messages. The module consumes the topics

- MOTIVATION_TEXT_REQUEST
- MOTIVATION_OUTPUT_REQUEST

and publishes on topics

- MOTIVATIONAL_TEXT_MESSAGE
- DIALOG_OUTPUT_REQUEST

2.8. CS-6: INTELLIGENT NOTIFICATIONS/SCHEDULER

2.8.1. GENERAL DESCRIPTION

This service provides a mechanism to notify the user about outstanding tasks or relevant information like changes within the pathway. Additionally, this service informs the patient about his daily schedule as foreseen by the rehabilitation plan, which can be configured through the caregiver user interface and is part of the KIOLA platform, developed and provided by AIT. Notifications can be triggered by any service or module within the vCare ecosystem via MQTT. The reminder and notification service ensures that the user retrieves the information and the user is able to postpone it if he or she wants to perform a task later or wants to get reminded again. Context information is used for efficient triggering.

2.8.2. INTERACTION DESIGN

The service can be triggered in three ways:

1. The service triggers itself automatically and regularly searches for upcoming events and tasks in the calendars
2. The service is triggered by the KRF which sends requests for reminders or notifications
3. The services can be triggered by the user himself by asking for upcoming events and tasks The interaction and the data flows with other modules are depicted in Figure 12 below.

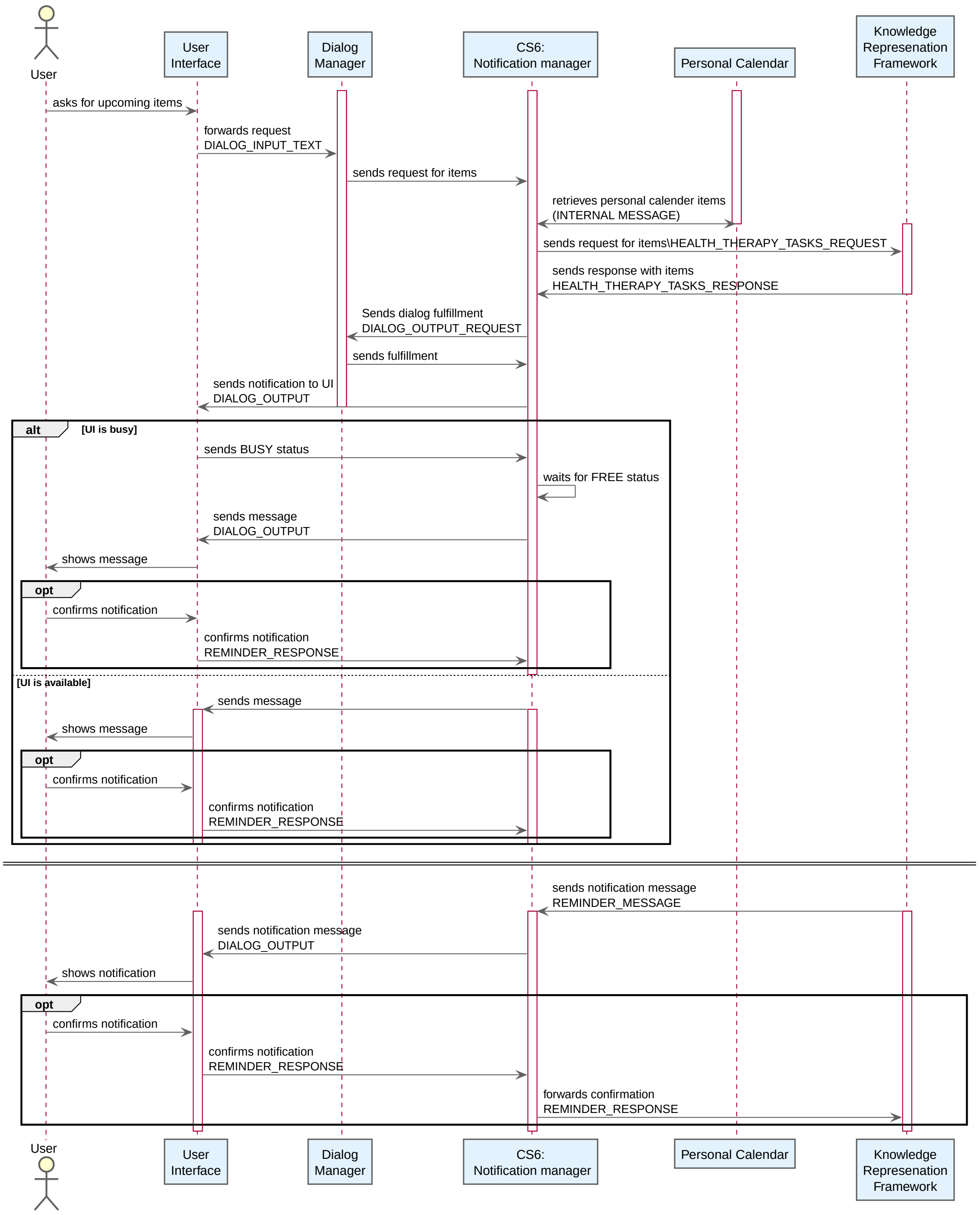

User
User Interface
Dialog Manager
CS6: Notification manager
Personal Calendar
Knowledge Represenation Framework
asks for upcoming items
forwards request DIALOG_INPUT_TEXT
sends request for items
retrieves personal calender items (INTERNAL MESSAGE)
sends request for items\HEALTH_THERAPY_TASKS_REQUEST
sends response with items HEALTH_THERAPY_TASKS_RESPONSE
Sends dialog fulfillment DIALOG_OUTPUT_REQUEST
sends fulfillment
sends notification to UI DIALOG_OUTPUT
alt
[UI is busy]
sends BUSY status
waits for FREE status
sends message DIALOG_OUTPUT
shows message
opt
confirms notification
confirms notification REMINDER_RESPONSE
[UI is available]
sends message
shows message
opt
confirms notification
confirms notification REMINDER_RESPONSE
sends notification message REMINDER_MESSAGE
sends notification message DIALOG_OUTPUT
shows notification
opt
confirms notification
confirms notification REMINDER_RESPONSE
forwards confirmation REMINDER_RESPONSE
User
User Interface
Dialog Manager
CS6: Notification manager
Personal Calendar
Knowledge Represenation Framework

Figure 12. Data flows and interaction schema for CS-6.

2.8.3. SPEECH INTERACTION DEFINITION

Interaction with no required response

Virtual Coach: "I have a reminder for you: today there is a physical exercise session scheduled."

Interaction with the required response

Virtual Coach: "I have a reminder for you: today there is a physical outdoor session scheduled. Did you perform it already?"
User: "Yes" | "No"

User triggered interaction

User: "What is on my schedule today?"
Virtual Coach: "Here is your schedule for today:
<item1_time> <item1_title>, <item2_time> <item2_title>…."

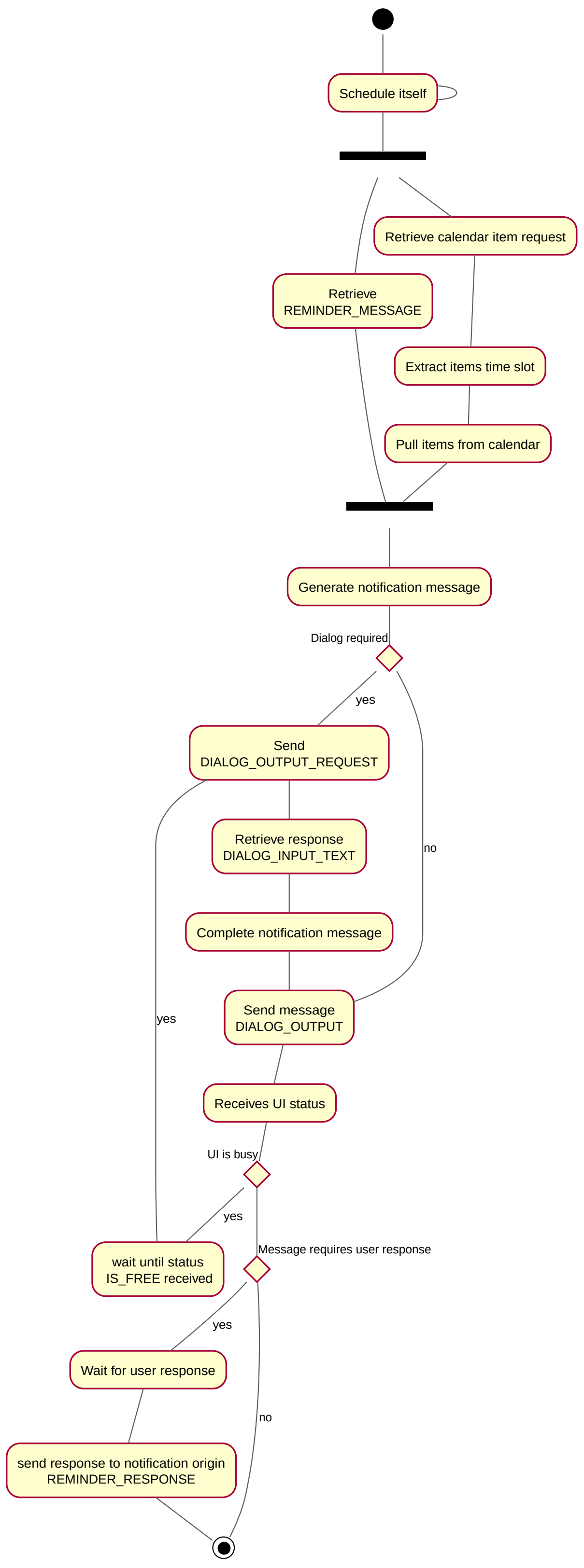
2.8.4. REQUIREMENTS ON PERSONAL DATA

The following list of data elements are required to retrieve from the KRF:

- User id
- Items the user shall be reminded on / informed about

2.8.5. IMPLEMENTATION

The module is implemented in Java as an OSGi Bundle running inside the vCare Coaching Services Container. The diagram below depicts the internal logic of the service generating e-learning session requests for the user (Figure 13). The notification manager ensures that every message is delivered to the user and only one message is provided at a time. This is crucial in particular when using speech output. The majority of the messages are generated by other modules, the manager itself only generates reminders for items of the personal calendar.



Schedule itself
Retrieve calendar item request
Retrieve REMINDER_MESSAGE
Extract items time slot
Pull items from calendar
Generate notification message
Dialog required
yes
no
Send DIALOG_OUTPUT_REQUEST
Retrieve response DIALOG_INPUT_TEXT
Complete notification message
Send message DIALOG_OUTPUT
yes
Receives UI status
UI is busy
yes
wait until status IS_FREE received
Message requires user response
yes
Wait for user response
no
send response to notification origin REMINDER_RESPONSE

Figure 13. Activity diagram for CS-6.

2.8.6. INTERFACE DESCRIPTION

The whole communication with other modules is implemented via OSGi EventAdmin, the connected MQTT broker and its related JSON messages. The module consumes the topics

- DIALOG_INPUT_TEXT
- REMINDER_MESSAGE
- HEALTH_THERAPY_TASKS_RESPONSE

and publishes on topics

- REMINDER_RESPONSE
- DIALOG_OUTPUT
- DIALOG_OUTPUT_REQUEST HEALTH_THERAPY_TASKS_REQUEST

2.9. CS-7: MEDICAL QUESTIONNAIRES

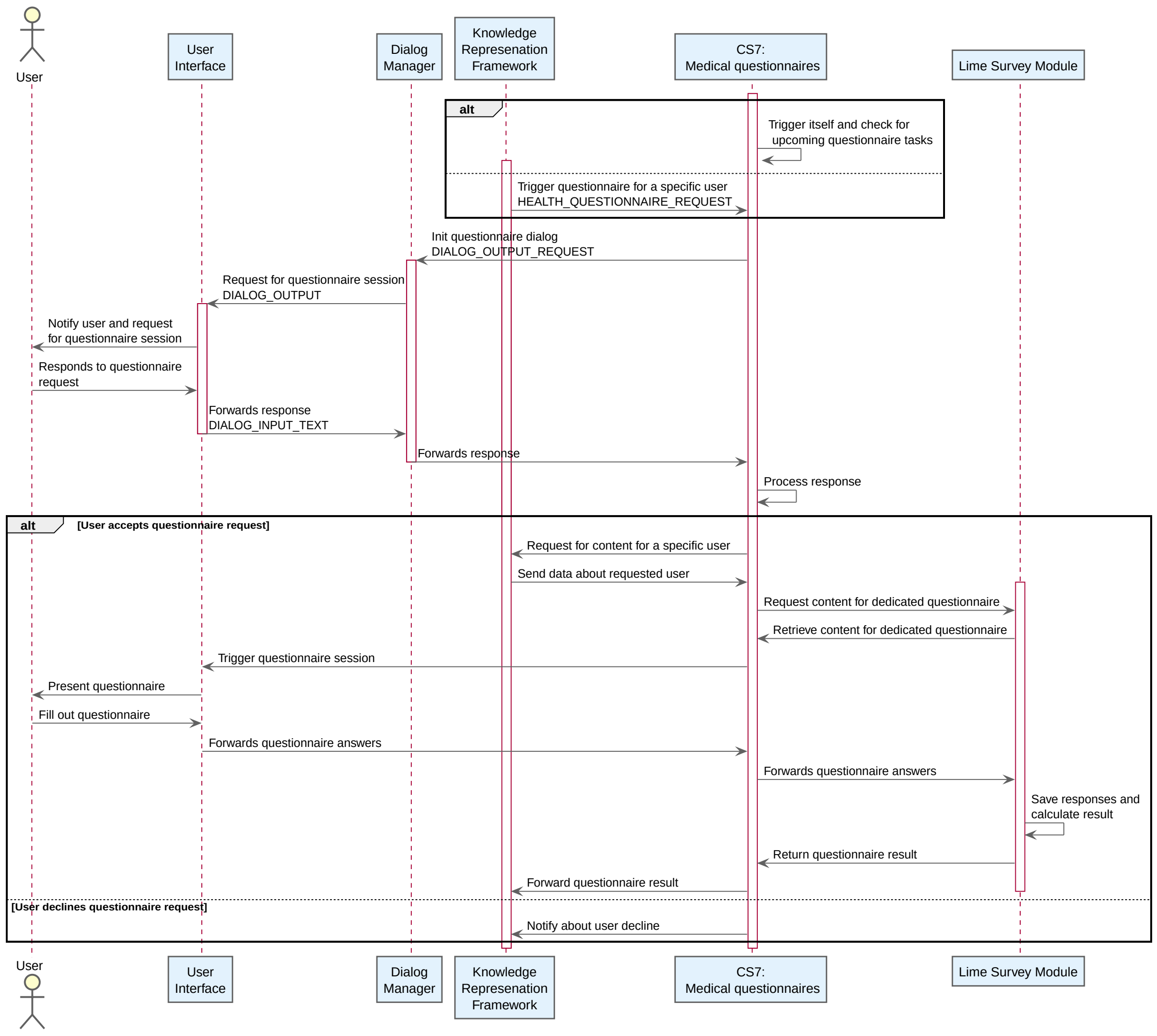
2.9.1. GENERAL DESCRIPTION

This service triggers a request to fill a standardized medical questionnaire at the user's home via a tablet interface. This is on top of the avatar and provided via touch input. One example for the questionnaire is the HADS [Snaith et al, 2003] for anxiety and depression scale. The answers to the questionnaire are stored and used to support the monitoring of the health status progression as well as the diagnosis of co-morbidities.

This service is system triggered, that means the system initiates the interaction and is scheduled according to the patient's rehabilitation plan.

2.9.2. INTERACTION DESIGN

The questionnaires are triggered by the KRF, by other modules which require questionnaires (e.g. CS8 - User Feeling) or by an internal scheduler of the service (see Figure 14).



User
User Interface
Dialog Manager
Knowledge Representation Framework
CS7: Medical questionnaires
Lime Survey Module
alt
Trigger itself and check for upcoming questionnaire tasks
Trigger questionnaire for a specific user HEALTH_QUESTIONNAIRE_REQUEST
Init questionnaire dialog DIALOG_OUTPUT_REQUEST
Request for questionnaire session DIALOG_OUTPUT
Notify user and request for questionnaire session
Responds to questionnaire request
Forwards response DIALOG_INPUT_TEXT
Forwards response
Process response
alt
[User accepts questionnaire request]
Request for content for a specific user
Send data about requested user
Request content for dedicated questionnaire
Retrieve content for dedicated questionnaire
Trigger questionnaire session
Present questionnaire
Fill out questionnaire
Forwards questionnaire answers
Forwards questionnaire answers
Save responses and calculate result
Return questionnaire result
Forward questionnaire result
[User declines questionnaire request]
Notify about user decline
User
User Interface
Dialog Manager
Knowledge Representation Framework
CS7: Medical questionnaires
Lime Survey Module

Figure 14. Data flows and interaction schema for CS-7.

2.9.3. SPEECH INTERACTION DESCRIPTION

Virtual Coach: "Today there is a questionnaire scheduled for you. Do you want to start with the questionnaire now?"
User: "Yes" | "No"
if no:
Virtual Coach: "Ok, I will come back to you later."
if yes:
Virtual Coach: "Ok, I will start with the questionnaire. Please use the touch screen of the tablet to provide your responses."

2.9.4. REQUIREMENTS ON PERSONAL DATA

The following list of data elements are required to retrieve from the KRF:

- User id
- List of questionnaires to be asked

2.9.5. IMPLEMENTATION

The service is implemented in Java as a OSGi bundle running inside the vCare Coaching Services Container. In Figure 15 the internal logic is depicted including the topics of sent and received MQTT messages. The Open Source LimeSurvey framework, which is connected to the coaching service via an RPC interface, is used for framing the questionnaires and analysing the results. Some information about LimeSurvey provided below (Section 2.9.7).

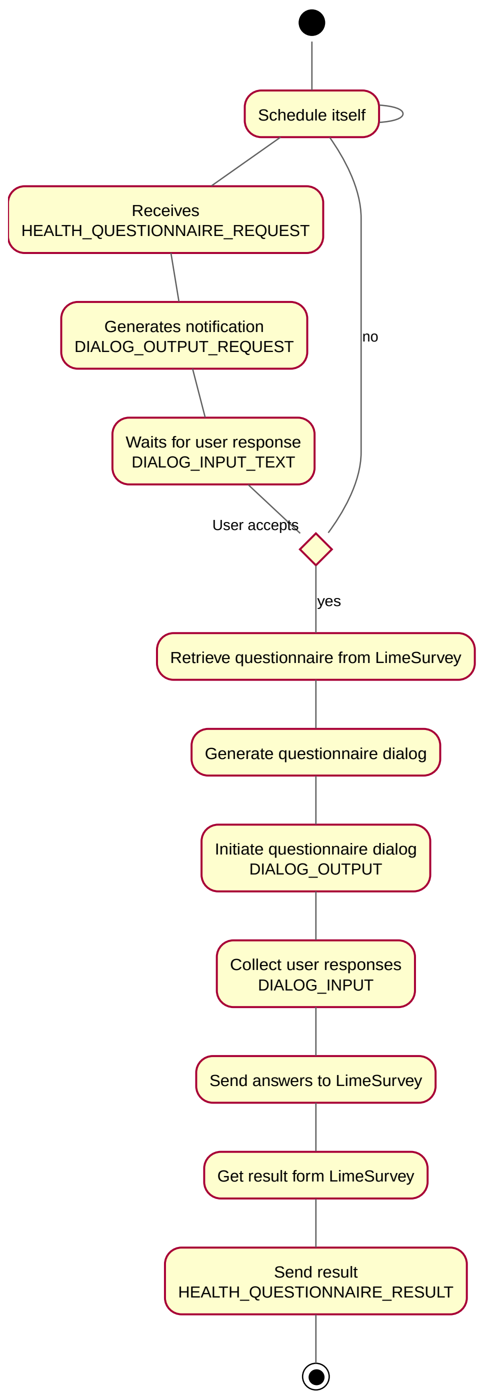

Schedule itself
Receives HEALTH_QUESTIONNAIRE_REQUEST
Generates notification DIALOG_OUTPUT_REQUEST
Waits for user response DIALOG_INPUT_TEXT
no
User accepts
yes
Retrieve questionnaire from LimeSurvey
Generate questionnaire dialog
Initiate questionnaire dialog DIALOG_OUTPUT
Collect user responses DIALOG_INPUT
Send answers to LimeSurvey
Get result form LimeSurvey
Send result HEALTH_QUESTIONNAIRE_RESULT

Figure 15. Activity diagram for CS-7.

2.9.6. INTERFACE DESCRIPTION

The whole communication with other modules is implemented via OSGi EventAdmin, the connected MQTT broker and its related JSON messages. The module consumes the topics

- HEALTH_QUESTIONNAIRE_REQUEST
- DIALOG_INPUT_TEXT
- DIALOG_INPUT

and publishes on topics

- HEALTH_QUESTIONNAIRE_RESULT
- DIALOG_OUTPUT_REQUEST
- DIALOG_OUTPUT

2.9.7. LIMESURVEY

LimeSurvey is a very popular Open Source survey tool which is provided as a cloud service but can also be self-hosted, which is the case for vCare. LimeSurvey is a framework which can host arbitrary custom surveys providing tools for analysing the responses as well as providing appropriate UIs for it. In vCare, we use LimeSurvey for hosting the questionnaires related to the pathways in a standardized manner and for analyzing the responses, but the questions as such are presented to the user via the vCare avatar-based UI. This service connects to the LimeSurvey framework via a JSON-RPC interface. The interface is documented online: https://api.limesurvey.org/classes/remotecontrol_handle.html, examples for using the interface in various programming languages exist.

2.10. CS-8: USER FEELING

2.10.1. GENERAL DESCRIPTION

The purpose of this service is to retrieve the user feeling by asking the user with a frequency as defined by the clinician, e.g. once a day, about his personal feeling. The user's response is classified and stored in the common database. The service uses the questionnaire service for asking the user and the evaluation of the results. In order to ensure accuracy, the user may only select from pre-defined responses according to standardized questionnaires measuring the subjective well-being.

2.10.2. INTERACTION DESIGN

The service triggers itself by following a predefined schedule, e.g. asking the user once a day in the morning about how he feels. The user will be asked verbally to give feedback on a simple questionnaire with pre-defined answers. The interaction and data flows are depicted below in Figure 16.

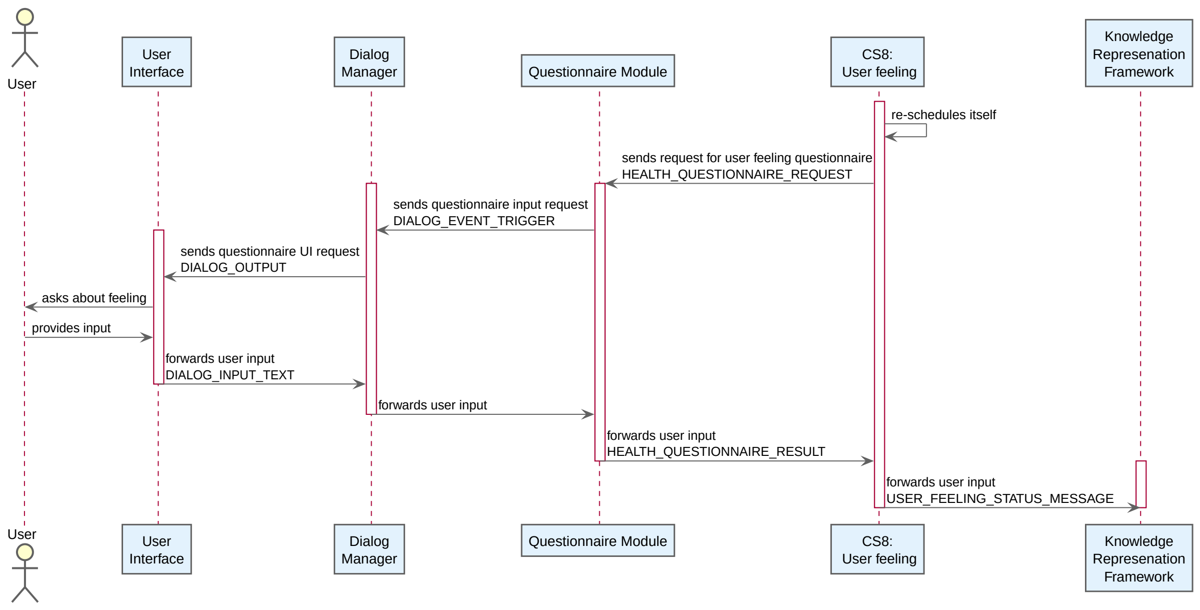

User
User Interface
Dialog Manager
Questionnaire Module
CS8: User feeling
Knowledge Represenation Framework
re-schedules itself
sends request for user feeling questionnaire HEALTH_QUESTIONNAIRE_REQUEST
sends questionnaire input request DIALOG_EVENT_TRIGGER
sends questionnaire UI request DIALOG_OUTPUT
asks about feeling
provides input
forwards user input DIALOG_INPUT_TEXT
forwards user input
forwards user input HEALTH_QUESTIONNAIRE_RESULT
forwards user input USER_FEELING_STATUS_MESSAGE

Figure 16. Data flows and interaction schema for CS-8.

2.10.3. SPEECH INTERACTION DEFINITION

Virtual Coach asks: "How do you feel today?" | "How are you today?"
User can respond: "Good" / "Dizzy" / "Weak" / "Bad" / "Not good" / "Tired"

2.10.4. REQUIREMENTS ON PERSONAL DATA

The following list of data elements are required to retrieve from the KRF:

- None

2.10.5. IMPLEMENTATION

The service is implemented in Java as a OSGi bundle to run inside the vCare Coaching Services Container. Internally, the service follows the structure depicted in Figure 17.

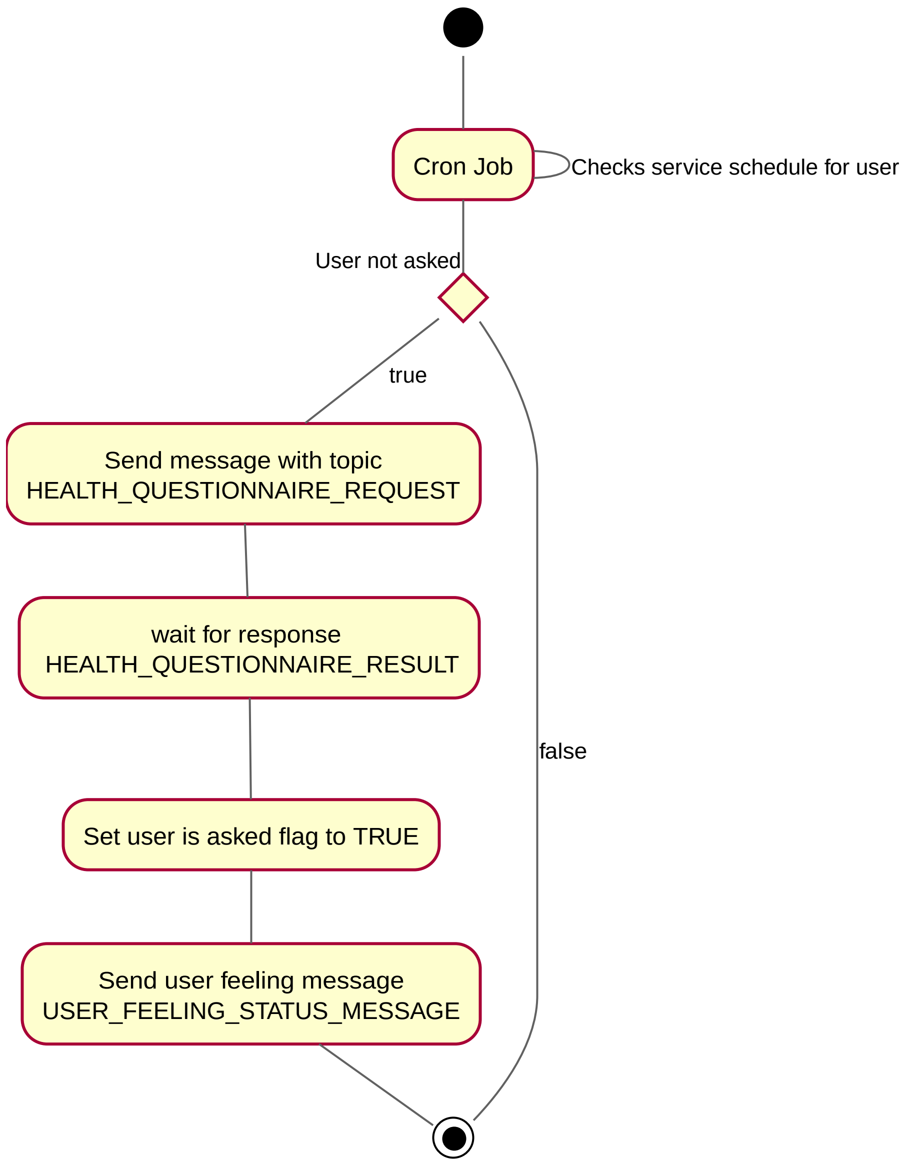

Cron Job
Checks service schedule for user
User not asked
true
false
Send message with topic HEALTH_QUESTIONNAIRE_REQUEST
wait for response HEALTH_QUESTIONNAIRE_RESULT
Set user is asked flag to TRUE
Send user feeling message USER_FEELING_STATUS_MESSAGE

Figure 17. Activity diagram for CS-8.

2.10.6. INTERFACE DESCRIPTION

The whole communication with other modules is implemented via OSGi EventAdmin, the connected MQTT broker and its related JSON messages. The module consumes the topics

- HEALTH_QUESTIONNAIRE_RESULT

and publishes on topics

- USER_FEELING_STATUS_MESSAGE
- HEALTH_QUESTIONNAIRE_REQUEST

2.11. CS-9: SPEECH AND SWALLOWING THERAPY

2.11.1. GENERAL DESCRIPTION

This service aims to provide digital support for therapy of dysphonia, dysarthria and oropharyngeal dysphagia via the avatar. The virtual coach avatar guides the patient through breathing exercises following a pre-defined therapy plan. Avatar gestures support the exercises. This service is system triggered by itself automatically according to the therapy plan provided by the KRF.

2.11.2. INTERACTION DESIGN

The service is triggered by an internal scheduler which pulls relevant information about the user's therapy plan related to the speech and swallowing therapies. The specific content of the exercises provided to the patient is not patient-specific, only the schedule for performing the exercises is retrieved from the Knowledge Representation Framework (KRF). The interaction and data flows are depicted in Figure 18 below.

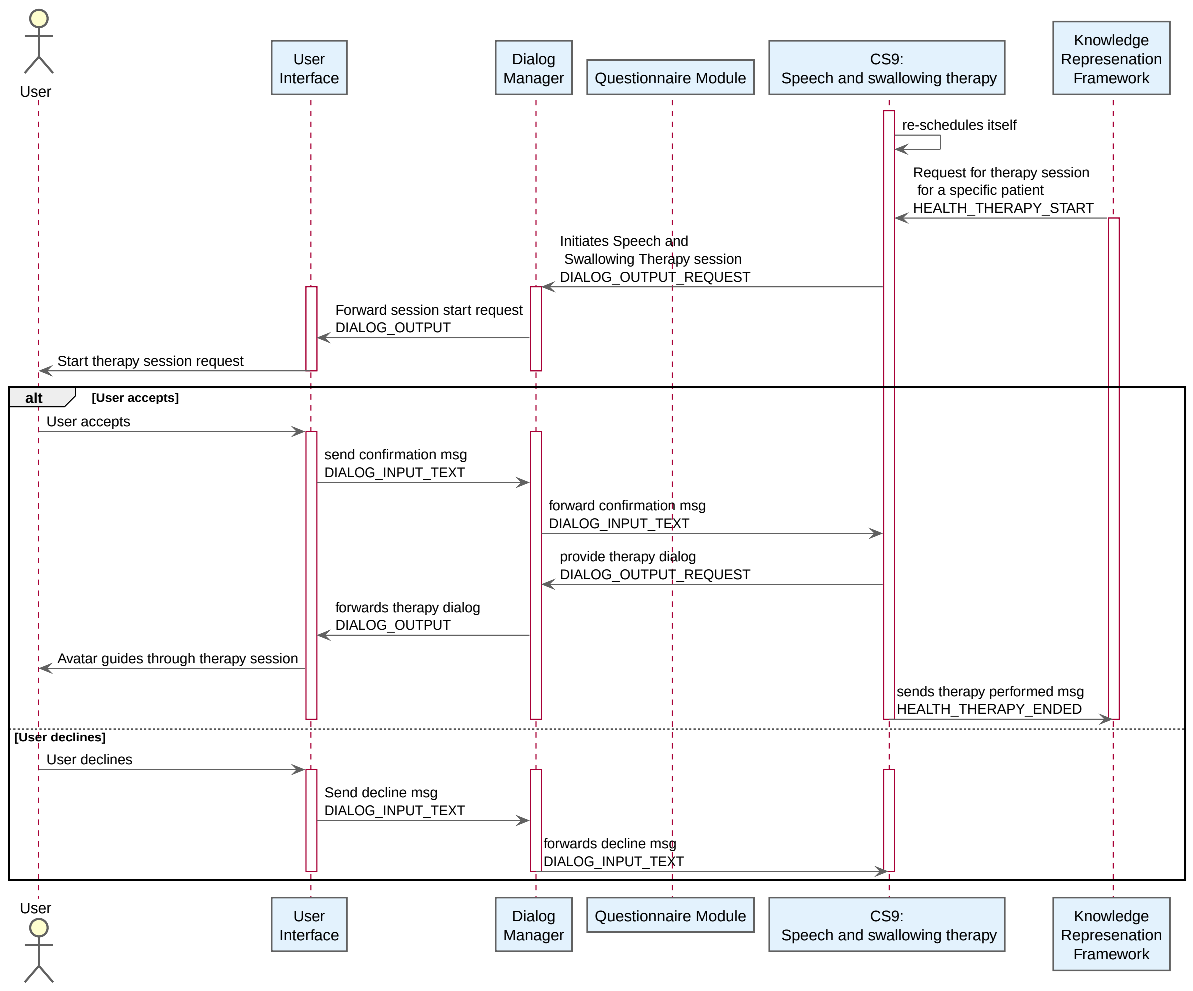

User
User Interface
Dialog Manager
Questionnaire Module
CS9: Speech and swallowing therapy
Knowledge Represenation Framework
re-schedules itself
Request for therapy session for a specific patient HEALTH_THERAPY_START
Initiates Speech and Swallowing Therapy session DIALOG_OUTPUT_REQUEST
Forward session start request DIALOG_OUTPUT
Start therapy session request
alt
[User accepts]
User accepts
send confirmation msg DIALOG_INPUT_TEXT
forward confirmation msg DIALOG_INPUT_TEXT
provide therapy dialog DIALOG_OUTPUT_REQUEST
forwards therapy dialog DIALOG_OUTPUT
Avatar guides through therapy session
sends therapy performed msg HEALTH_THERAPY_ENDED
[User declines]
User declines
Send decline msg DIALOG_INPUT_TEXT
forwards decline msg DIALOG_INPUT_TEXT

Figure 18. Data flows and interaction schema for CS-9.

2.11.3. SPEECH INTERACTION DEFINITION

Speech interaction with the user to decline the therapy

Virtual Coach asks: "Today it is time for your speech therapy. Shall I start the exercises now?"
User can respond: "No"
Virtual Coach responses: "Ok, I will come back to you later."

Speech interaction with the user to accept the therapy

Virtual Coach asks: "Today it is time for your speech therapy. Shall I start the exercises now?"
User can respond: "Yes, please | Yes"
Virtual Coach responses: "Ok, please stand-up in upright position in front of me. We start in 5 Seconds."
Virtual Coach states: "We will start with a 5-cycle inspiration exercise with a 5 second pause. Breath-in - (5 secs pause) - breath out - (5 secs pause) - breath in - (5 secs pause) - breath out - (5 secs pause) - breath in - (5 secs pause) - breath out - (5 secs pause) - breath in - (5 secs pause) - breath out - breath in - breath out."
Virtual Coach states: "Perfect, we continue with a slow expiration exercise. Breath in - (4 secs pause) - breath out slowly; 4, 3, 2, 1. Great."
Virtual Coach states: "Perfect, we continue with a slow expiration exercise. Breath in - (4 secs pause) - breath out slowly; 4, 3, 2, 1. Great. Let’s continue."
Virtual Coach states: "Breath in - (3 secs pause) - breath out slowly - 3 - 2 -1. Perfect!"
Virtual Coach states: "Now breath by speaking the letter 's'. Breath in through your nose and hold the air for 3 seconds. Then breath out slowly and feel how your belly gets in- and deflated. Let’s start. Breath in - 3 - 2 - 1, hold for 3 - 2 -1 seconds, now slowly breath out - 3 - 2 -1."
Virtual Coach states: "Ok, one more time, but now spell 'a'. Breath in through your nose and hold the air for 3 seconds. Then breath out slowly and feel how your belly gets in- and deflated. Let’s start. Breath in - 3 - 2 - 1, hold for 3 - 2 -1 seconds, now slowly breath out - 3 - 2 - 1."
Virtual Coach states: "That’s it for today, how did you feel?"
User can respond: "Good | hard | bad | fine".
The Virtual Coach responds accordingly, e.g. "Good to hear that | Will be better next time".

2.11.4. REQUIREMENTS ON PERSONAL DATA

The following list of data elements are required to retrieve from the KRF:

- User id
- Time and date when the speech and swallowing therapy is scheduled for this user

2.11.5. IMPLEMENTATION

The service is implemented in Java as OSGi bundle to run inside the vCare Coaching Services Container. Internally, the service follows the structure depicted in Figure 19.

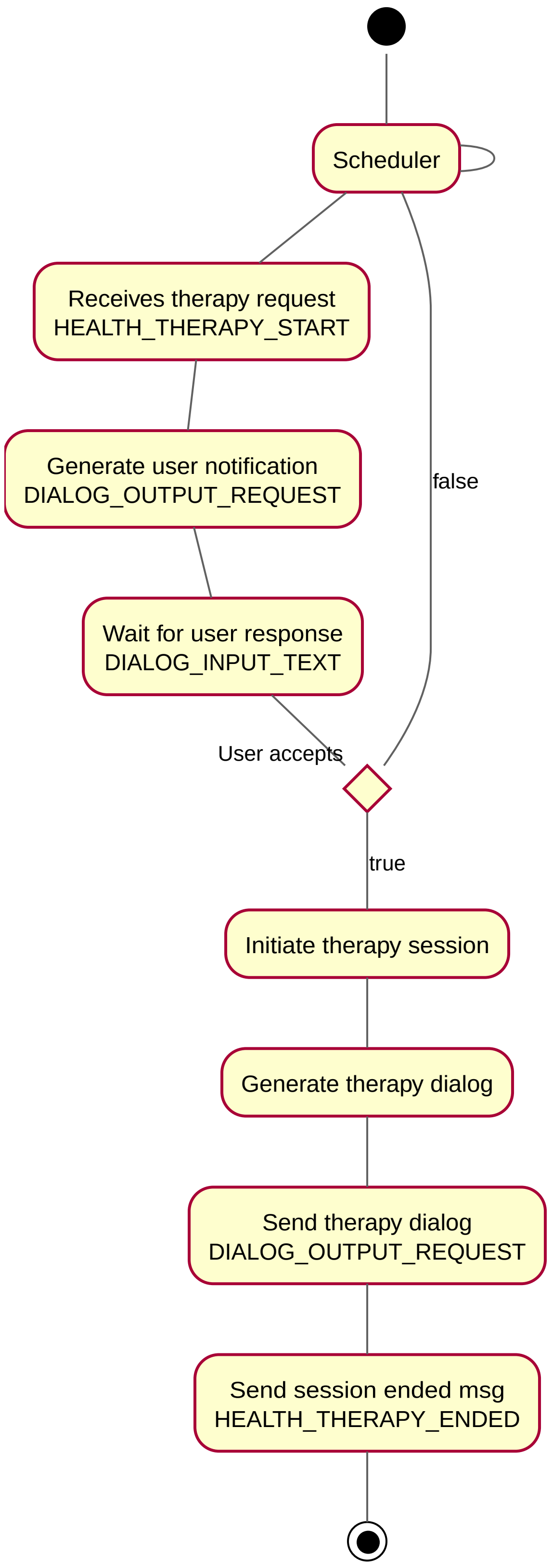


Figure 19. Activity diagram for CS-9

2.11.6. INTERFACE DESCRIPTION

The whole communication with other modules is implemented via OSGi EventAdmin, the connected MQTT broker and its related JSON messages. The module consumes the topics

- HEALTH_THERAPY_START
- DIALOG_INPUT_TEXT

and publishes on topics

- DIALOG_OUTPUT_REQUEST
- HEALTH_THERAPY_ENDED

3. SUPPORTING SERVICES

Supporting services provide context information feeding coaching services or modules of the UI layer supporting the intelligent user interaction. Additionally, they can be used to retrieve non-pathway related information for supporting the activities of daily life, following specific interest of the user or just for fun. Supporting services have in common that they retrieve information from third-party services over the web. They are depicted in Table 3. In this section, the backend components of the supporting services - including the user interactions with them - are described. How this is implemented at the UI side in detail will be described in D3.3 *Recipient and Caregiver Interface Description*.

Table 3. Supporting services overview

#	Name	Description	Requirement	Partner	Implementation	Deployment	Section
SS-1	Weather (Backend Service)	Provides weather information for a given location and given time.	R3-10	AIT	Java	Local device /cloud	3.1
SS-2	Agenda (Backend Service)	Provides access to the patient to a personal calendar and reminders related to non-pathway related events, e.g. family events etc. Even though the agenda is integrated into the UI, this service forms the backend holding and providing the data.	R3-1, R3-7	AIT	Java	cloud	3.2
SS-3	Standby (Backend Service)	Puts the VC in a standby/passive mode not to disturb the patient.	R3-11	AIT	Java	cloud	3.3

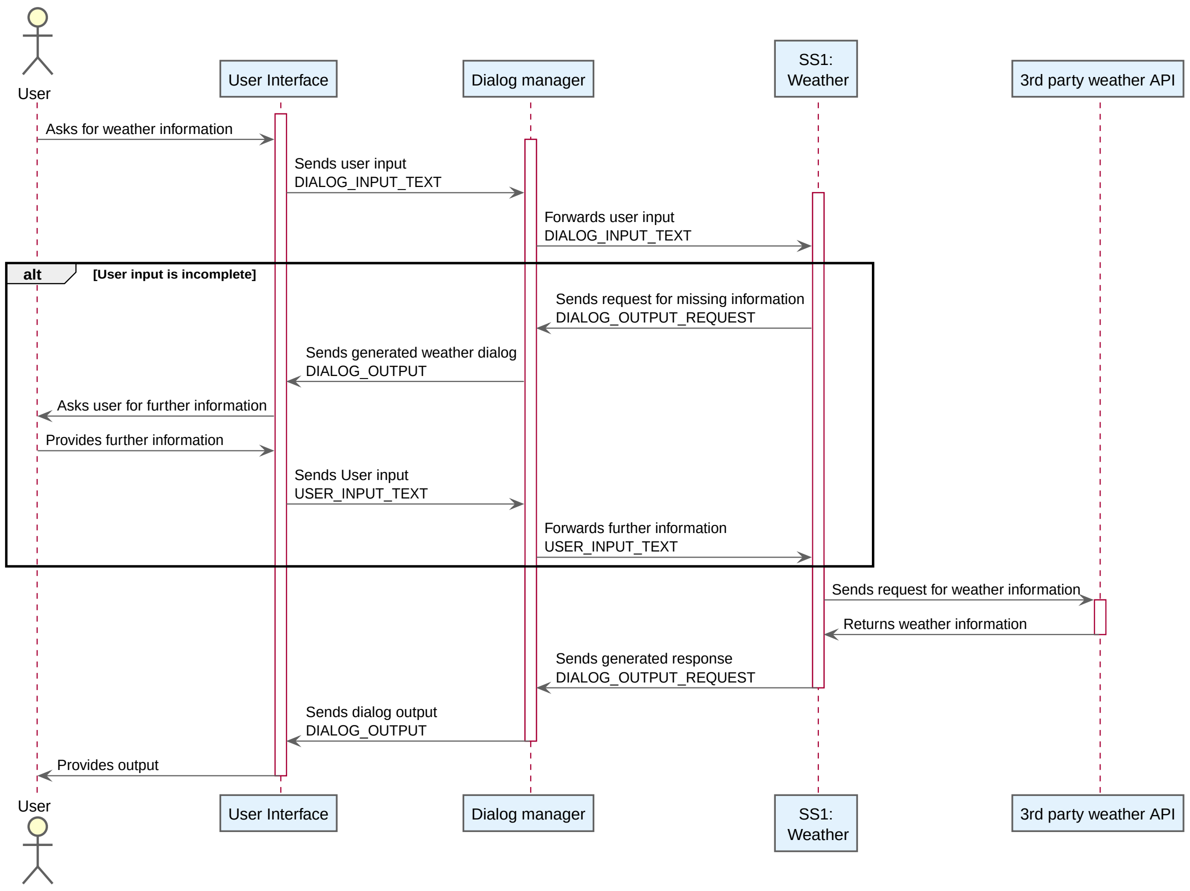
3.1. SS-1 - WEATHER

3.1.1. GENERAL DESCRIPTION

The User can ask the system for the weather forecast of the current day and/or the next day for a specific location. If no location is provided, the current location as defined in the user profile is used. For retrieving the data, a free and open third-party solution is used: http://www.openweathermap.org. The service provides a REST API providing data in JSON, XML or HTML format. The weather service converts the weather data in human-understandable text in the language used as a eference for speech output.

3.1.2. INTERACTION DESIGN

The module is independent of the vCare knowledge layer and has a direct interaction with the user. The user can request weather information via speech input. The overall interaction diagram including the data exchange with other modules is depicted in Figure 20.



User
User Interface
Dialog manager
SS1: Weather
3rd party weather API
Asks for weather information
Sends user input DIALOG_INPUT_TEXT
Forwards user input DIALOG_INPUT_TEXT
alt [User input is incomplete]
Sends request for missing information DIALOG_OUTPUT_REQUEST
Sends generated weather dialog DIALOG_OUTPUT
Asks user for further information
Provides further information
Sends User input USER_INPUT_TEXT
Forwards further information USER_INPUT_TEXT
Sends request for weather information
Returns weather information
Sends generated response DIALOG_OUTPUT_REQUEST
Sends dialog output DIALOG_OUTPUT
Provides output

Figure 20. Data flows and interaction schema for SS-1.

3.1.3. SPEECH INTERACTION DEFINITION

Automated fulfilment

User: "How is the weather?" Virtual Coach response (takes the location of the user's home and today as time): "The weather today in Vienna will be sunny. The temperature ranges from 20 to 25 degrees."

Fulfilment with asking the user

User: "How will be the weather?"
Virtual Coach response: "For which date do you want to get the weather information?"
User: "Tomorrow!"
Virtual Coach response (takes the location of the user's home): "The weather tomorrow in Vienna will be sunny. The temperature ranges from 20 to 25 degrees."

Other location

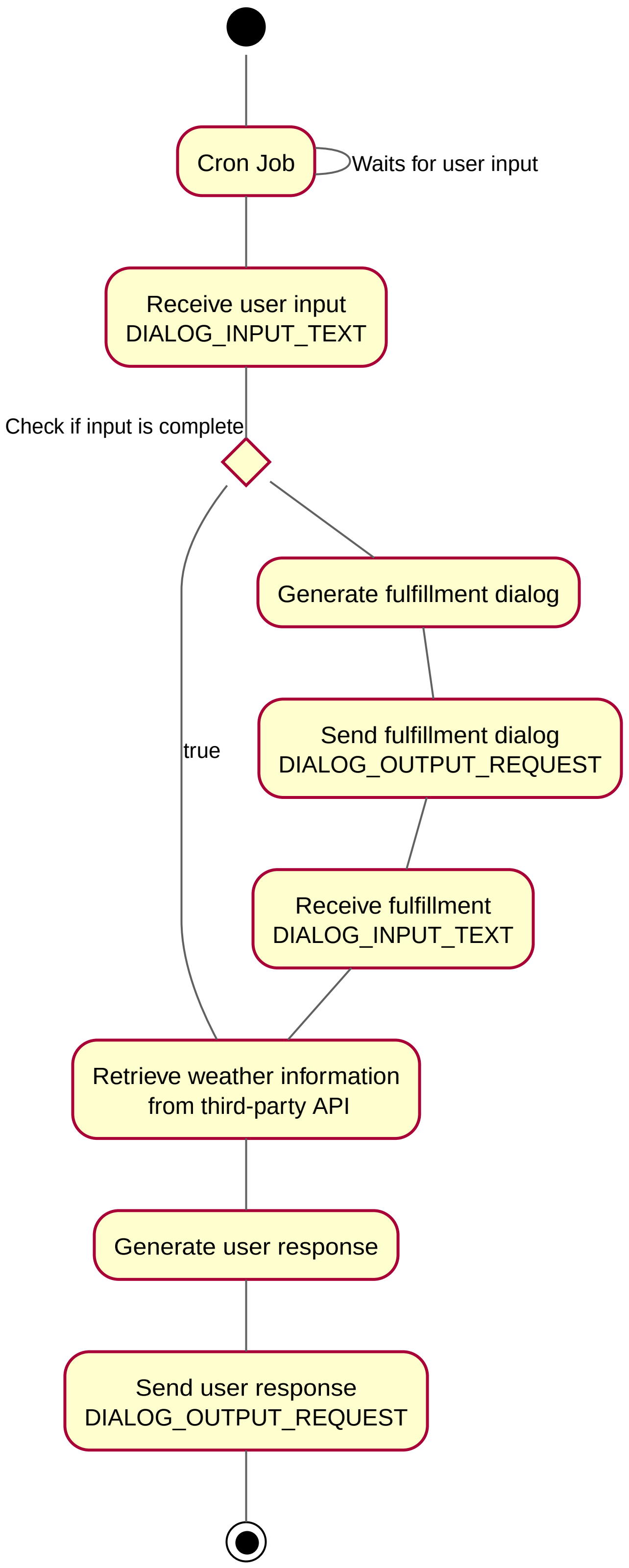
User: "How will be the weather in Prague next week?"
Virtual Coach response: "The weather next week in Prague will be sunny. The temperature ranges from 15 to 25 degrees."

Requirements on personal data

- User location

3.1.4. IMPLEMENTATION

The service is implemented in Java as a OSGi bundle to run inside the vCare Java Karaf Framework. Internally, the service follows the structure depicted in Figure 21.



Cron Job
Waits for user input
Receive user input DIALOG_INPUT_TEXT
Check if input is complete
true
Generate fulfillment dialog
Send fulfillment dialog DIALOG_OUTPUT_REQUEST
Receive fulfillment DIALOG_INPUT_TEXT
Retrieve weather information from third-party API
Generate user response
Send user response DIALOG_OUTPUT_REQUEST

Figure 21. Activity diagram for SS-1.

3.1.5. INTERFACE DESCRIPTION

The whole communication with other modules is implemented via OSGi EventAdmin, the connected MQTT broker and its related JSON messages. The modules (the OSGi coaching service) consumes the topics

- DIALOG_INPUT_TEXT

and publishes on topics

- DIALOG_OUTPUT_REQUEST

3.2. SS-2 - AGENDA

3.2.1. GENERAL DESCRIPTION

This service provides information and reminders about the personal, non-pathway related agenda of the patient, like having an appointment with a friend/family or at a hairdresser. Furthermore, the calendar synchronizes with the calendar in the backend setup by the care professionals which records the therapy schedule and appointments at the doctor/hospital.

3.2.2. INTERACTION DESIGN

The user can retrieve from and add new custom events to the calendar using speech input. The interaction with the user as well as the data exchange with other modules is depicted in Figure 22. The interaction schema for retrieving and setting new calendar items is exactly the same, only the dialog differs.

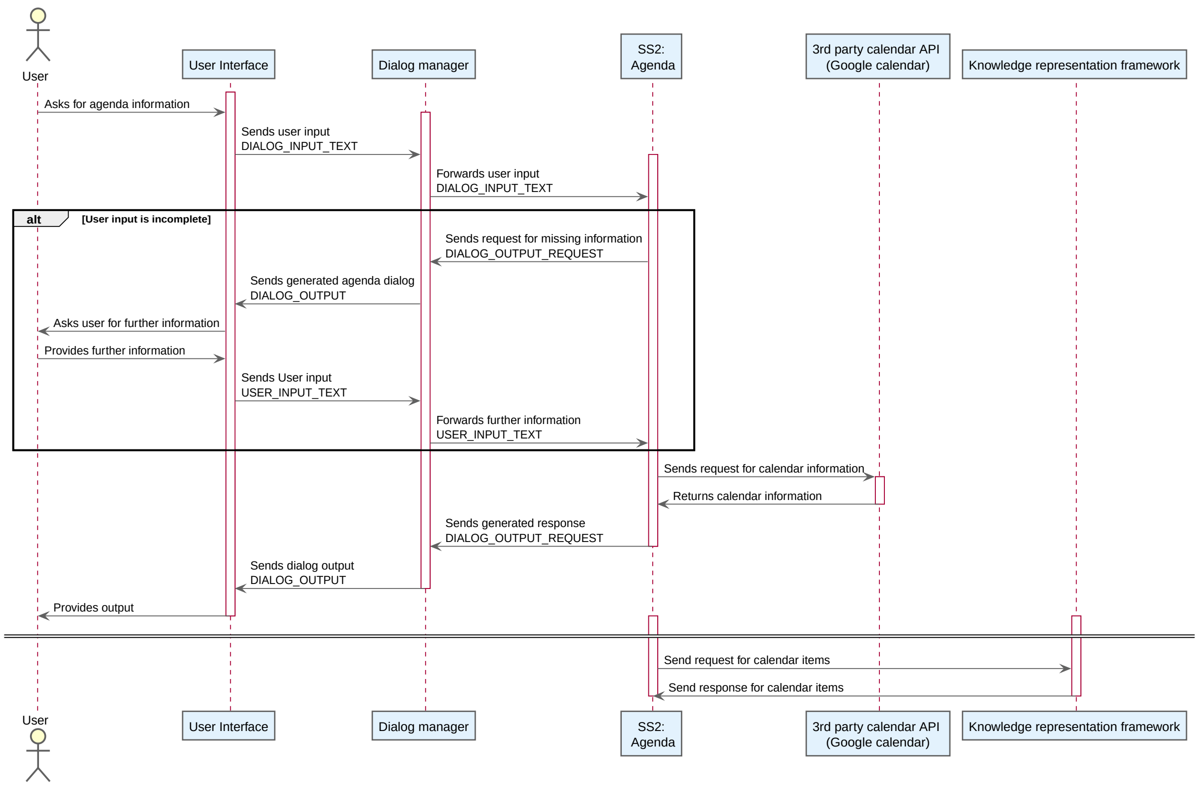

User
User Interface
Dialog manager
SS2: Agenda
3rd party calendar API (Google calendar)
Knowledge representation framework
Asks for agenda information
Sends user input DIALOG_INPUT_TEXT
Forwards user input DIALOG_INPUT_TEXT
alt
[User input is incomplete]
Sends request for missing information DIALOG_OUTPUT_REQUEST
Sends generated agenda dialog DIALOG_OUTPUT
Asks user for further information
Provides further information
Sends User input USER_INPUT_TEXT
Forwards further information USER_INPUT_TEXT
Sends request for calendar information
Returns calendar information
Sends generated response DIALOG_OUTPUT_REQUEST
Sends dialog output DIALOG_OUTPUT
Provides output
Send request for calendar items
Send response for calendar items

Figure 22. Data flows and interaction schema for SS-2.

3.2.3. SPEECH INTERACTION DEFINITION

Automated fulfillment

User: "What is on my agenda?" Virtual Coach response *(assumes that the user wants to know the agenda for today)*: "Today on your agenda is:
9 o'clock: physical exercises session
11 o'clock: appointment with Frank at Mary's cafe.
2 o'clock: appointment at the hairdresser".

Other time

User: "What is on my agenda tomorrow | in the afternoon | next Monday?" Virtual Coach response:"Tomorrow | In the afternoon | Next Monday on your agenda is:
9 o'clock: physical exercises session
11 o'clock: appointment with Frank at Mary's cafe.
2 o'clock: appointment at the hairdresser".

User adds custom items

User: "Please add a new entry on my agenda".
Virtual Coach : "Ok, please tell me what is it about"
User: "Hairdresser"
Virtual Coach: "Ok, when is it scheduled?"
User: "Tomorrow, 9 o'clock".

Virtual Coach: "Ok, do you want to add a location?"

Option 1:
User: "No".
Virtual Coach: "Ok, your new event has been stored."

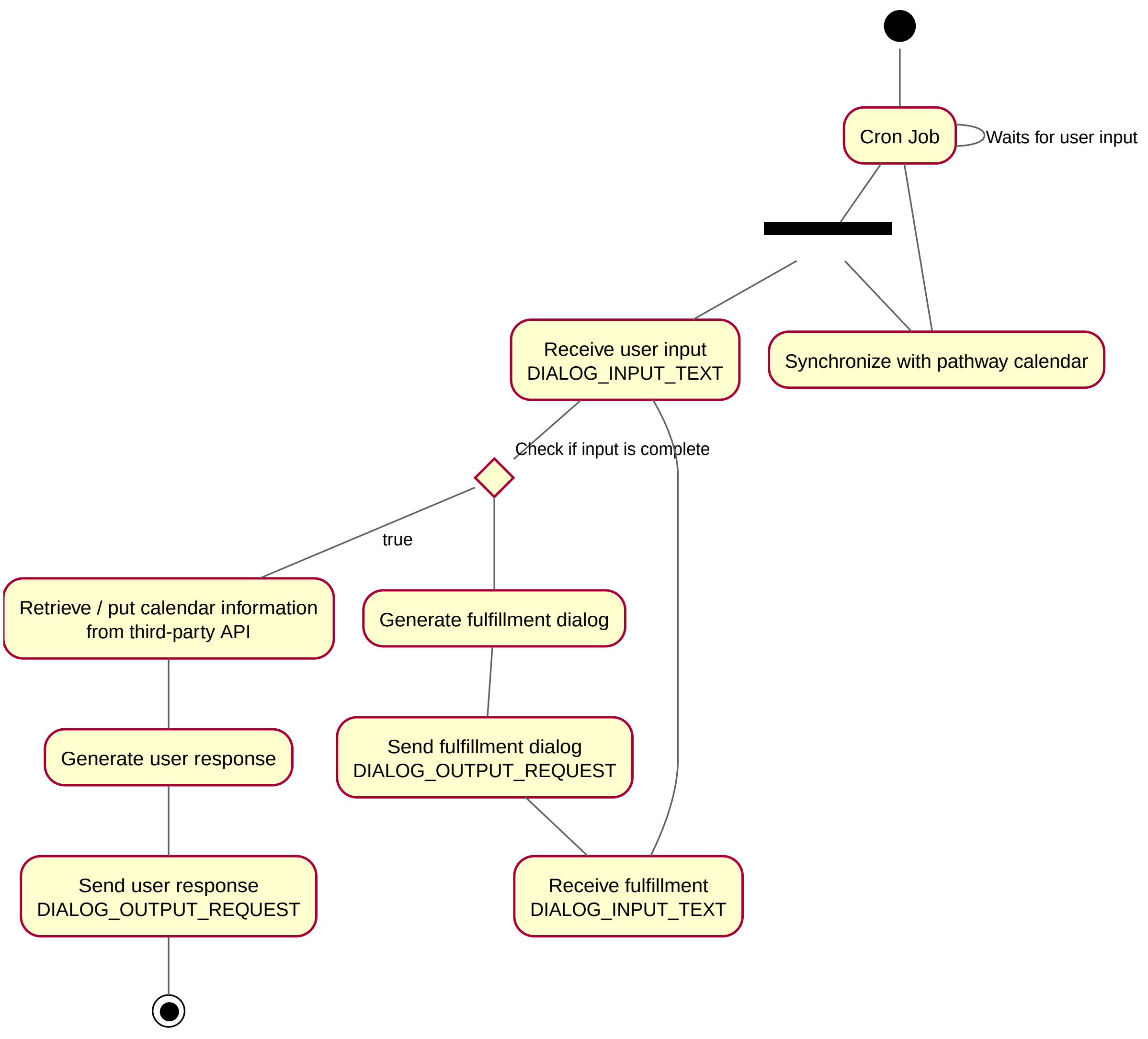
Option 2:
User: "Yes".
Virtual Coach: "Ok, please tell me the location."
User: "Mary's café"
Virtual Coach: "Ok, your new calendar item has been stored."

3.2.4. REQUIREMENTS ON PERSONAL DATA

- Personal calendar
- Pathway calendar

3.2.5. IMPLEMENTATION

The service is implemented in Java as OSGi bundle to run inside the vCare Java Karaf Framework. Internally, the service follows the structure depicted in Figure 23.



Cron Job
Waits for user input
Receive user input DIALOG_INPUT_TEXT
Synchronize with pathway calendar
Check if input is complete
true
Retrieve / put calendar information from third-party API
Generate fulfillment dialog
Generate user response
Send fulfillment dialog DIALOG_OUTPUT_REQUEST
Send user response DIALOG_OUTPUT_REQUEST
Receive fulfillment DIALOG_INPUT_TEXT

Figure 23. Activity diagram for SS-2.

3.2.6. INTERFACE DESCRIPTION

The whole communication with other modules is implemented via OSGi EventAdmin, the connected MQTT broker and its related JSON messages. The modules (the OSGi coaching service) consumes the topics

- DIALOG_INPUT_TEXT

and publishes on topics

- DIALOG_OUTPUT_REQUEST

3.3. SS-3 - STANDBY

3.3.1. GENERAL DESCRIPTION

The service enables the user to put the system in standby mode by voice input and/or button click (touch). In standby, the voice detection is switched off and there is no speech output provided by the system. The UI remains on and may provide silent messages while in stand-by mode. Hence, while in this mode, notifications are shown only on the display in a non-obtrusive way. The avatar indicates that the system is not active / listening, e.g. by not showing the avatar itself on the UI.

3.3.2. INTERACTION DESIGN

The user can switch the avatar based UI to silent mode via a speech command or by clicking a button in the UI. In silent mode, no speech input will be recognized, speech output is deactivated as well. The silent mode has a timeout to anticipate a situation such as when the user forgets to switch off the silent mode again. The timeout has a default value (1 hour), but can be modified by the user. The interaction with the user and the data exchange with other modules is depicted in Figure 24. For reactivating the avatar, the user can press a button or use a specific command which is processed locally on the device.

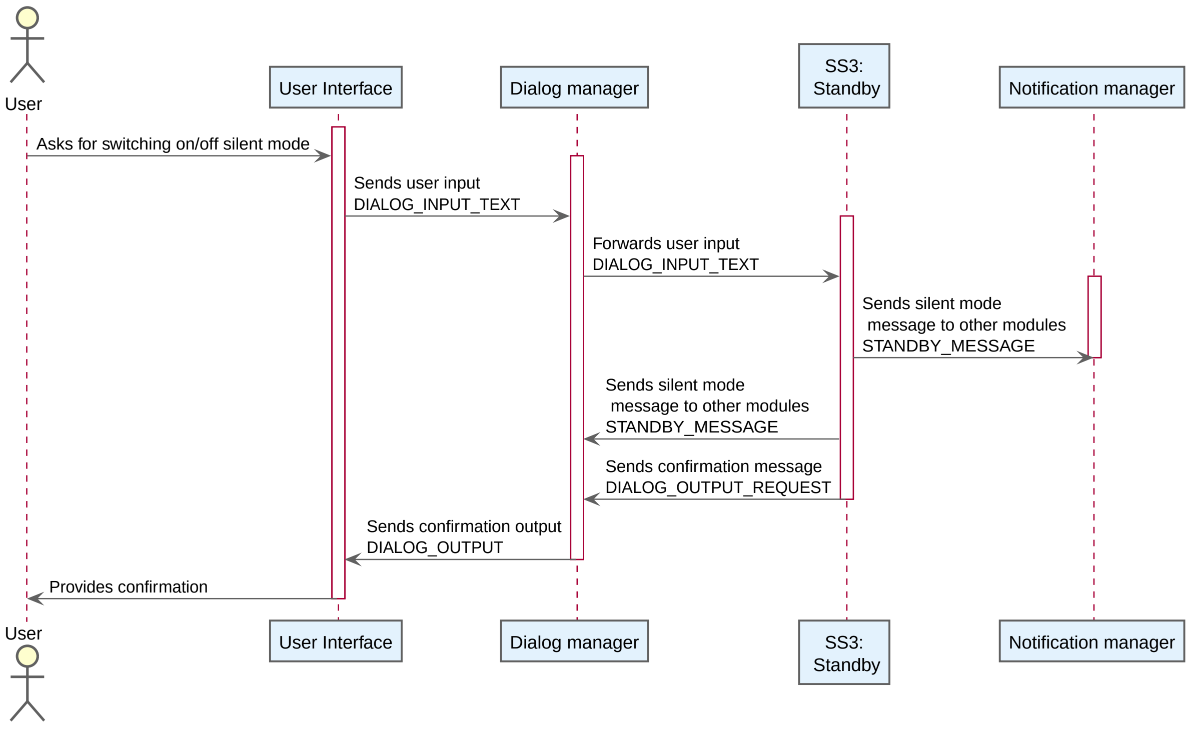

User
User Interface
Dialog manager
SS3: Standby
Notification manager
Asks for switching on/off silent mode
Sends user input DIALOG_INPUT_TEXT
Forwards user input DIALOG_INPUT_TEXT
Sends silent mode message to other modules STANDBY_MESSAGE
Sends silent mode message to other modules STANDBY_MESSAGE
Sends confirmation message DIALOG_OUTPUT_REQUEST
Sends confirmation output DIALOG_OUTPUT
Provides confirmation
User
User Interface
Dialog manager
SS3: Standby
Notification manager

Figure 24. Data flows and interaction schema for SS-3.

3.3.3. SPEECH INTERACTION DEFINITION

User: "Please switch to standby."
Virtual Coach response: "I’m now in standby mode. You can wake me up by pressing the reactivation button or saying "Wake up" to me."

3.3.4. REQUIREMENTS ON PERSONAL DATA

- None

3.3.5. IMPLEMENTATION

The service is implemented in Java as OSGi bundle to run inside the vCare Java Karaf Framework. Internally, the service follows the structure depicted in Figure 25.

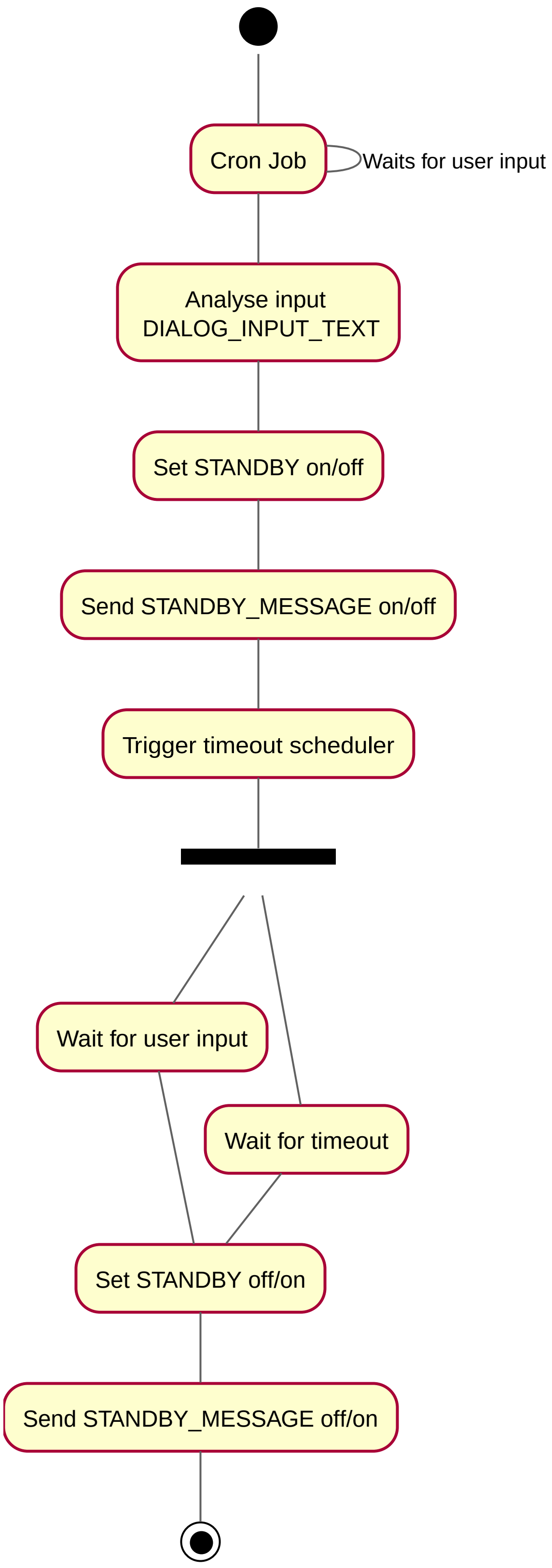

Cron Job
Waits for user input
Analyse input DIALOG_INPUT_TEXT
Set STANDBY on/off
Send STANDBY_MESSAGE on/off
Trigger timeout scheduler
Wait for user input
Wait for timeout
Set STANDBY off/on
Send STANDBY_MESSAGE off/on

Figure 25. Activity diagram for SS-3.

3.3.6. INTERFACE DESCRIPTION

The whole communication with other modules is implemented via OSGi EventAdmin, the connected MQTT broker and its related JSON messages. The modules (the OSGi coaching service) consumes the topics

- DIALOG_INPUT_TEXT

and publishes on topics

- STANDBY_MESSAGE

4. AUTHENTICATION AND AUTHORIZATION

Within vCare identification and authentication are enforced by using Keycloak in combination with the KIOLA platform. KIOLA is part of the security components layer of the overall vCare system architecture, which uses Keycloak as a single-sign-on solution for all services invoked in vCare. Furthermore, KIOLA provides an identity management component and a role and user management system for all services ensuring that only authorized stakeholders can access the data of the patients.

For security reasons tokens, provided by Keycloak, following the JWT specification [Jones et al, 2015] are used for authentication and authorization between the various vCare services and components.

Since the vCare architecture is a distributed one with components in different layers which needs to communicate with each other in a secure way, vCare requires an identity management component, which is provided by the KIOLA [Hanke et al, 2017] platform. A RESTful interface is provided for the various components of the vCare platform. The interface enables clients to register local data identifiers through a HTTP POST request. KIOLA is then able to link all known identifiers through the Keycloak user account. Moreover, KIOLA provides an interface to map and retrieve identifiers through a HTTP GET request as required. Permissions in KIOLA are organized around so-called user circles, illustrated in Figure 26. User circles define a number of actors (e.g., general practitioners, clinical experts, coordinators) and an audience (patients). According to GDPR, only health care professionals in direct care relationship with the patient can have access to his data. Otherwise a specific consent is required. Typical examples of user circles include: hospitals, medical office or even entire regions (e.g., a country, a city or a federal state):

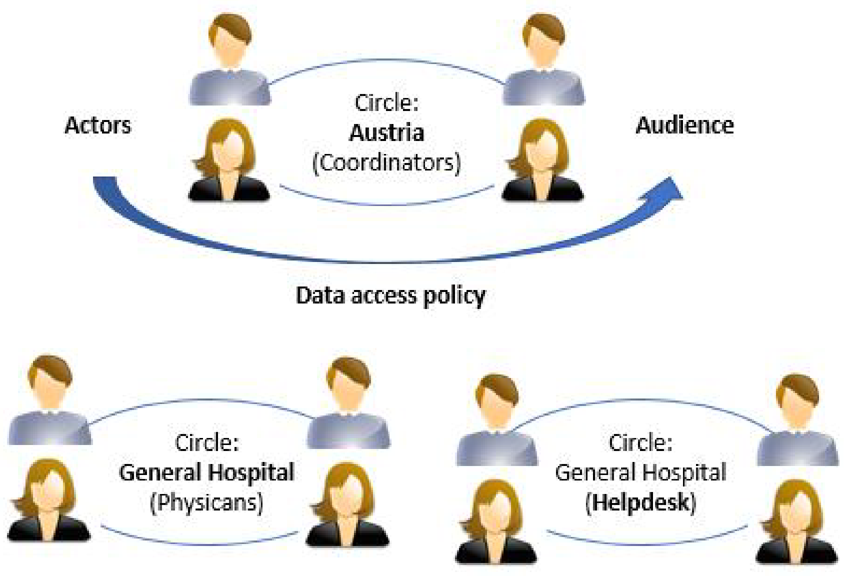


Figure 26. Circle permission system: Circles are defined by actors (users who can access data), audience (typically patients whose data is accessed), the role (e.g. coordinator) which defines the data access policy (what data can be defined).

User registration can be performed only by coordinators and includes three steps:

- providing personal data
- patient assignment to a circle (thus, to a clinic/medical office)
- review of the data provided

4.1. AUTHENTICATION OF THE END USER DEVICE

The patient logs in with his device with pre-configured credentials. Even though any OAUTH2 [Hardt, 2012] provided could be used, in the specific context of the vCare trials, username and password, stored in KeyCloak, will be used. Since the User Interface (UI) is exclusively connected to the MQTT broker, it needs authentication only for it. The workflow of this procedure is depicted in Figure Figure 27. The workflow in the diagram is applicable to any module which wants to send and consume messages to and from the MQTT broker. The application authenticates itself with a client Id and a Client secret.

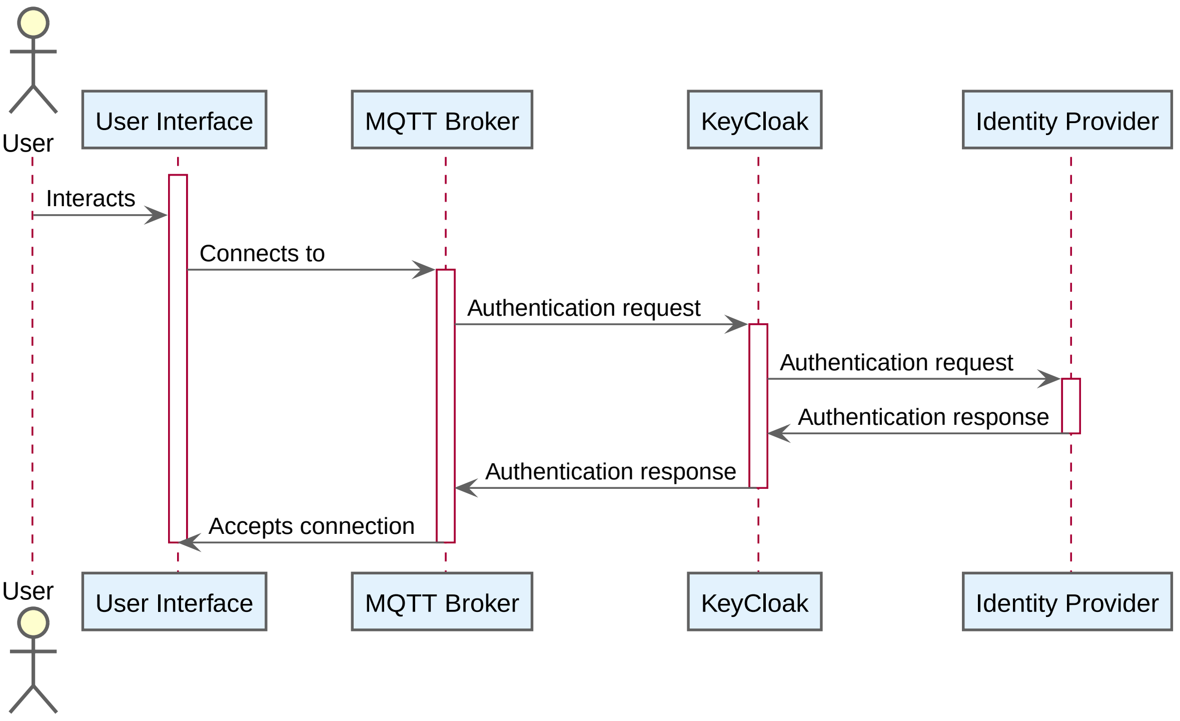


Figure 27. Diagram of the UI authentication process

4.2. AUTHENTICATION OF CARE PROFESSIONALS

Care professionals interact with the vCare system only via the KIOLA platform which is used as the care giver user interface. Hence, the authentication of care professionals will be performed within the KIOLA using the internal authentication features. Care professionals will get a username and a password for logging in and appropriate roles will be assigned to them as described in the fist section of this chapter. Patients will also get an internal id used in the KIOLA platform which will be mapped to the KeyCloak id. This can be performed automatically using the identity management features of the KIOLA platform.

4.3. AUTHENTICATION OF BACKEND SERVICES

Backend services mainly communicate over the MQTT broker and will use a so called M2M authorization procedure. The procedure foresees that an application which wants to connect to an API sends a token request to the authorization broker, KeyCloak in our case. The procedure is depicted below in Figure 28.

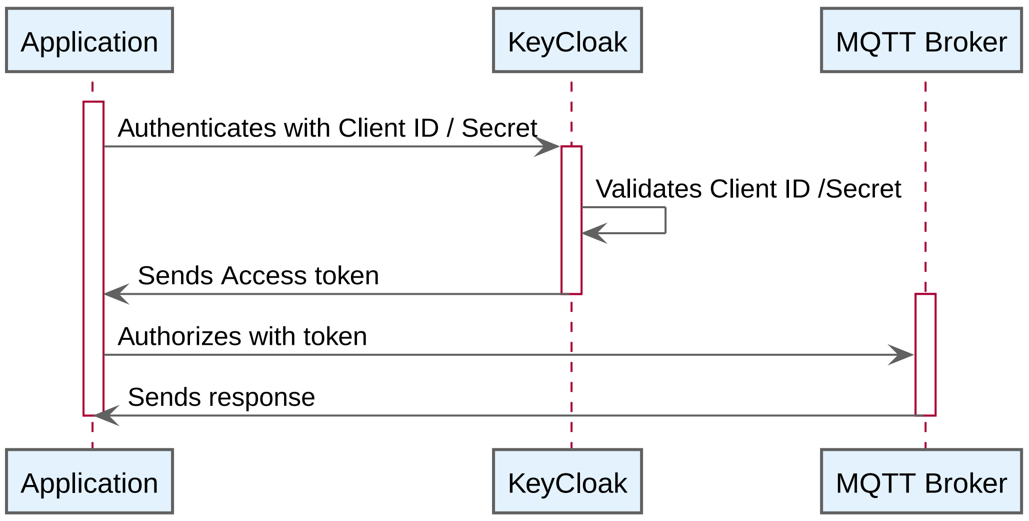


Figure 28. Diagram of the M2M authentication process

4.4. AUTHENTICATION OF 3RD PARTY SERVICE CC2U

To support the implementation of the vCaaS paradigm (vCare-as-a-Service), the CC2U (CloudCare2U, [Kyriazakos et al, 2018], [iSprint, 2017]) IoT platform, provided by the partner iSprint will be used as a demo case. CloudCare2U is a cloud-based platform delivering Personalized Health Care Services through advanced interaction with patients with chronic diseases and frailty conditions.

With vCaaS, vCare functionalities may on one hand be enriched by integrating services provided by third parties, on the other hand, vCare can by connected other, already established platforms to extend their functionality. In the following, the authentication mechanisms for these two scenarios are described.

4.4.1. BACKGROUND

The CC2U Login service is the entry point for all the CC2U users to the CC2U platform. Every user who is entitled to access the platform is assigned a username and password. Before accessing an CC2U service, users must authenticate by providing their credentials towards the Login service. Then, the Login service checks the validity of the credentials (existence of the user and matching with a hashed password). After this preliminary verification, the service retrieves the user profile from the CC2U Profiling Server and extracts from it the user role and the list of the applications that such user is authorized to access. This setting (definition of the set of applications per user) is from the CC2U portal by an administrator when inserting a new user (or modifying an existing one) into the system. From the list of applications, the Login service retrieves the list of all the needed backend services. Finally, the Login service builds a "token", a string containing a set of information which the end user and the set of services he/she can access. The token is compliant with the JSON Web Token (JWT, [Jones et al, 2015]) specification used in vCare. The token contains information such as the username, the expiration date (the token has limited lifetime), the set of services accessible, and the user role. After creation, the token is encoded and signed using the content of the token itself and a secret key, which is known only by the login service and by the backend services which builds up the CC2U platform. The token is then sent back to the client over HTTP/HTTPS. In all following requests towards every CC2U service endpoint, the client must provide in an HTTP header the token obtained after login. The service providers will extract the token from the request and check it for validation before serving the request. The signing mechanism grants that the token, if modified by a client (to get more permissions than stated by the claims), will be considered invalid by the service providers. In fact, the change will be immediately detected, due to the mismatch between the signature in the token and the signature built for verification by the server by applying the secret key to the token contents received by the client. Since the token has an expiration time, the Login service provides to clients an endpoint which allows them to renew an existing token before its expiration. In summary, the Login service provides two REST endpoints for:

- Authenticating a client, taking as input username and password, and returning a token in case of successful authentication
- Renewing a valid token, taking as input a valid token and returning a new token with updated contents and a new expiration date

Thanks to the renewal mechanism, a client can maintain the token live for the time required by requesting a renewal before the expiration time. Moreover, since the token is dynamically generated based on the user profile, if a modification in the profile happens, at the next renewal the client can detect the change and react accordingly, e.g. by modifying an application GUI to reflect the current set of applications that a user can access.

4.4.2. EXAMPLE AUTHENTICATION AND AUTHORIZATION FLOW OF CC2U WTH VCARE USING THE VCAAS API

CC2U consuming services from vCare

Since the CC2U services use the same JWT token that is present in Keycloak, only one needs to match CC2U services contained in the token with the vCare services. Instead of the CC2U login service handling the authentication, it will forward the CC2U client ID together with the secret towards the vCaaS which will take care of the token generation via KeyCloak as well as the periodic renewal of the token. The schema hereunder summarizes the workflow (Figure 29):.

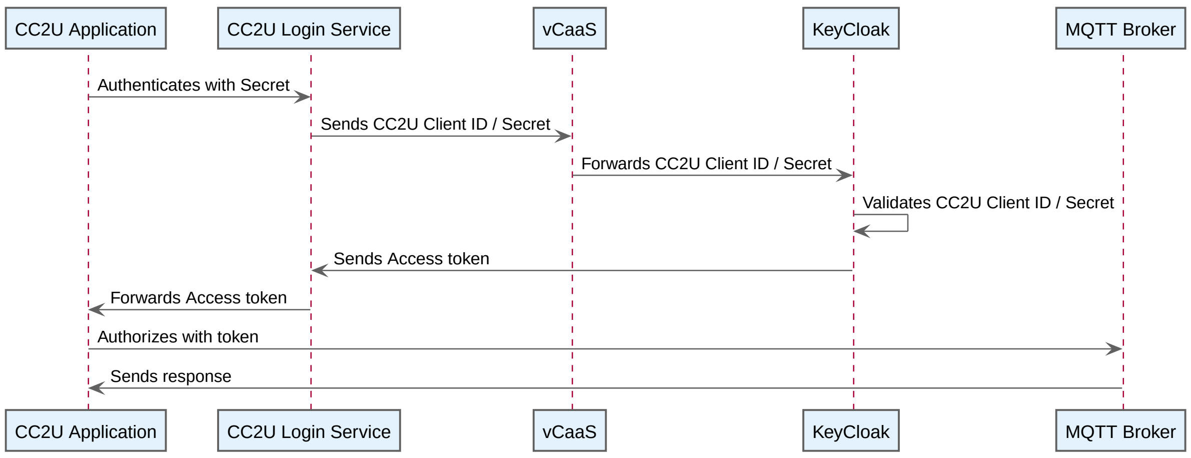


Figure 29. Authentication of CC2U Application when consuming a service from vCare

On the contrary, when vCare services want to consume services of the CC2U platform, the CC2U Login Service plays the role of the authentication broker. How this works logically is depicted in Figure 30.

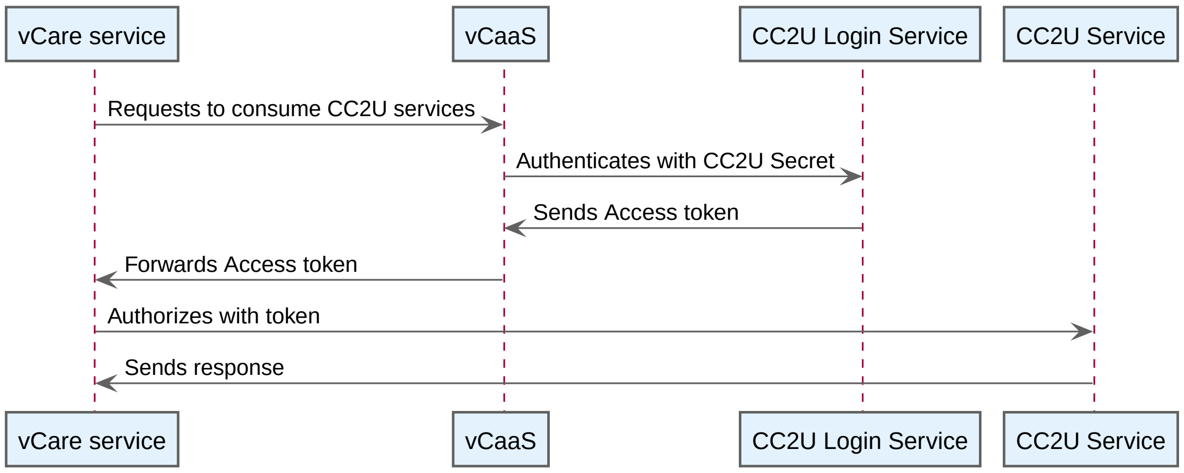


Figure 30. Authentication of vCare service when consuming a service from CC2U

5. INTER-MODULE COMMUNICATION

Communication between modules is established by structured messages using the JSON format. For data to be exchanged between services which are running inside the OSGi services container, the OSGi event bus is used. This is described in the following Section 5.1. For the communication with modules outside of the services container, the MQTT transport mechanism is used over a centralized MQTT broker. The data format remains the same as for the internal case. For doing so, the OSGi services container provides a MQTT connector for bidirectionally forwarding messages. The messages per se are defined in Section 5.2. On the KRF side, mappings for translating the pathway-related ontological messages in RDF format to the MQTT messaging format will be implemented. Since no further pathway-related decision making is performed within the coaching services layer, the semantic information from the ontology is not needed herein.

5.1. INTER-SERVICE COMMUNICATION

For the vCare Coaching Services and the underlying OSGi framework one important concept is the concept of sending and receiving (a)synchronous events between the modules for communication purposes.

In the vCare Coaching Services this concept is also used outside the actual OSGi framework for communication with mobile devices.

For the OSGi framework internal use the functionality for inter module communication is specified by the OSGi specification [OSGi Alliance, 2013] in chapter "113 Event Admin Service Specification". The so called EventAdmin is the OGSi service which manages the distribution of events among the components which are registered for receiving certain events. The EventAdmin itself is supplied by the OSGi framework and provides all the necessary functionalities for sending/receiving events.

The communication works via topics where everyone who wants to receive events and implements the EventHandler interface can register for receiving multiple topics. After a successful registration these topics then will be sent to the EventHandler by calling the "handleEvent()" function.

For further information on how the implementation works, please have a look at https://felix.apache.org/documentation/subprojects/apache-felix-event-admin.html.

The vCare Coaching Services framework uses this concept also for sending events over the borders of the OSGi framework and translates the messages sent around in the framework using other carriers like MQTT. The concept of Message Queuing Telemetry Transport (MQTT), which is an open OASIS [Oasis, 2020] and ISO standard (ISO/IEC PRF 20922, [ISO, 2016]), offers a lightweight, publish-subscribe network protocol that transports messages between devices. As the MQTT concept is pretty close to the OSGi Event Admin Service Specification in terms of topics and events, in combination with JSON (JavaScript Object Notation) it enables the communication with devices over TCP/IP networks.

The vCare Coaching Services framework provides a so called MQTT-Connector module which translates the OSGi internal events to MQTT events and the events coming through MQTT back to OSGi Event Admin compliant events. All this happens in such a way that it is fully transparent for the user. As it’s not expected that all internal communication is available beyond of the OSGi framework this connector can be configured to include/exclude certain topics for both directions.

The tables on the following pages describe the event types and the event properties defined for the vCare Coaching Services.

The separate event types are clustered in the categories:

- DIALOG
- GAMING
- HEALTH
- REMINDER
- SUPPORT
- MESSAGE
- LOCATION
- POSITION

Each event has a related topic which specifies the type of an event. All topics defined in vCare start with at/ac/ait/hbs/dialog/ or eu/vcare/event/ followed by a category identifier. For a specific event one may have one or more properties defined. A property consists basically of a key and a corresponding value. All the specified properties for a certain topic are also included in the descriptions in the following sections.

For better understanding and readability also a JSON representation is included. In the OSGi framework itself the events sent via the EventAdmin are Java objects of the type org.osgi.service.event.Event. When using connectors like the WebSocket or MQTT the transferred content pretty looks like the JSON used.

5.2. EVENT TOPICS DEFINITION

5.3. EVENT CATEGORY "DIALOG"

Event topics linked to this category are all related to dialog management.

This category contains the following topics as described in Table 4:

Table 4. Overview of the message topics discussed in this section.

DIALOG_CLIENT_MODE_MESSAGE	A message sent to the dialog management from the output device
DIALOG_CLIENT_MODE_REQUEST	A message sent to the dialog management from the output device

DIALOG_CLIENT_PING	A message sent to the dialog management
DIALOG_CLIENT_PONG	A message sent from the dialog management
DIALOG_CLIENT_STATUS_MESSAGE	A message sent to the dialog management from the output device
DIALOG_EVENT_TRIGGER	A message to trigger dialog management event
DIALOG_INPUT_TEXT	A message to trigger dialog management
DIALOG_OUTPUT	A message sent from the dialog management to the output device
DIALOG_OUTPUT_REQUEST	A message sent to the dialog management to be sent to the output device

5.3.1. DIALOG_CLIENT_MODE_MESSAGE

This is a message that is sent to the dialog management from the output device containing information about the client's output mode

Table 5. Details of DIALOG_CLIENT_MODE_MESSAGE

Name	DIALOG_CLIENT_MODE_MESSAGE
Topic	at/ac/ait/hbs/dialog/dialog/client/mode/message
Use	A message sent to the dialog management from the output device
References	-

Sequence diagram

The following diagram (Figure 31) shows how the "DIALOG_CLIENT_MODE_MESSAGE", the dialog manager and underlying services play together:

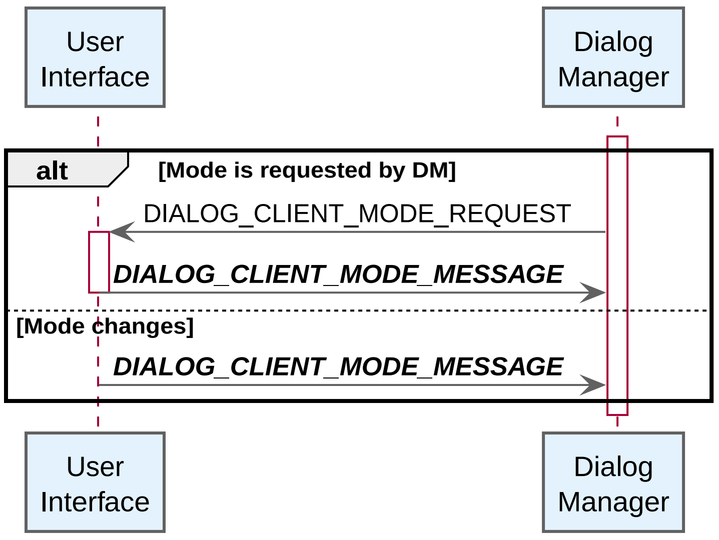

User Interface
Dialog Manager
alt
[Mode is requested by DM]
DIALOG_CLIENT_MODE_REQUEST
DIALOG_CLIENT_MODE_MESSAGE
[Mode changes]
DIALOG_CLIENT_MODE_MESSAGE
User Interface
Dialog Manager

Figure 31. Sequence diagram of the DIALOG_CLIENT_MODE_MESSAGE topic use.

Table 6. Properties of DIALOG_CLIENT_MODE_MESSAGE

Property	Class	Description
CLIENT_ID	String	A string specifying the unique identifier of a client that sends/should receive a message
OUTPUT_MODE	ClientOutputMode	A enum describing the current mode of the client in terms of the output modality
SOURCE_ID [1]	String	A string specifying the component that sent out the message
OFFSET_TO_UTC	Integer	A long representing the offset to UTC in seconds
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC

[1] property is optional

5.3.2. DIALOG_CLIENT_MODE_REQUEST

This is a message that is sent to the dialog management from the output device containing information about the client's output mode

Table 7. Details of DIALOG_CLIENT_MODE_REQUEST

Name	DIALOG_CLIENT_MODE_REQUEST
Topic	at/ac/ait/hbs/dialog/dialog/client/mode/request
Use	A message sent to the dialog management from the output device
References	-

Sequence diagram

The following diagram (Figure 32) shows how the "DIALOG_CLIENT_MODE_REQUEST", the dialog manager and underlying services play together:

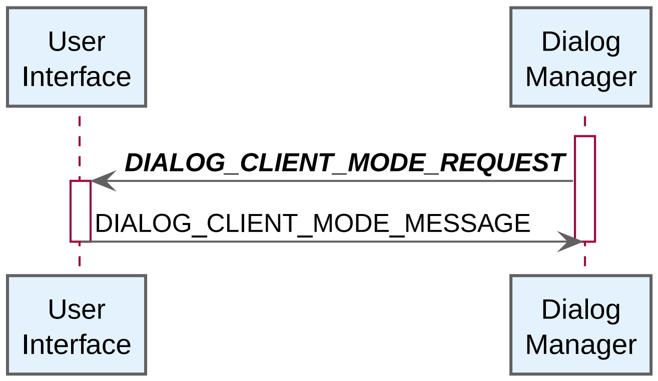


Figure 32. Sequence diagram of the DIALOG_CLIENT_MODE_REQUEST topic use.

Table 8. Properties of DIALOG_CLIENT_MODE_REQUEST

Property	Class	Description
CLIENT_ID	String	A string specifying the unique identifier of a client that sends/should receive a message
SOURCE_ID [1]	String	A string specifying the component that sent out the message
OFFSET_TO_UTC	Integer	A long representing the offset to UTC in seconds
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC

[1] property is optional

5.3.3. DIALOG_CLIENT_PING

This is a message sent to the dialog management to check if the dialog management is alive

Table 9. Details of DIALOG_CLIENT_PING

Name	DIALOG_CLIENT_PING
Topic	at/ac/ait/hbs/dialog/dialog/client/ping
Use	A message sent to the dialog management
References	-

Sequence diagram

The following diagram (Figure 33) shows how the "DIALOG_CLIENT_PING" and the dialog manager play together:

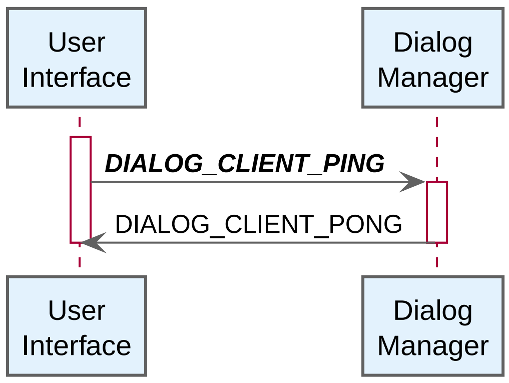


Figure 33. Sequence diagram of the DIALOG_CLIENT_PING topic use.

Table 10. Properties of DIALOG_CLIENT_PING

Property	**Class**	**Description**
CLIENT_ID	String	A string specifying the unique identifier of a client that sends/should receive a message
SOURCE_ID [1]	String	A string specifying the component that sent out the message
OFFSET_TO_UTC	Integer	A long representing the offset to UTC in seconds
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC

[1] property is optional

DIALOG_CLIENT_PING example

```
{
  "topic": "at/ac/ait/hbs/dialog/dialog/client/ping",
  "properties": {
    "OFFSET_TO_UTC": 0,
    "mqttUser": "123456",
    "SOURCE_ID": "at.ac.ait.hbs.dialog.demo.client.package",
    "CLIENT_ID": "098697ce-cfba-4e8a-98ba-75bbc74c1c51",
    "mqttClientId": "da9d7c8d-555b-4bd1-ce5d-5d61dfd0eb91",
    "TIMESTAMP": 1584060514773
  }
}
```

5.3.4. DIALOG_CLIENT_PONG

This is a message sent from the dialog management to acknowledge that the dialog management is alive

Table 11. Details of DIALOG_CLIENT_PONG

Name	DIALOG_CLIENT_PONG
Topic	at/ac/ait/hbs/dialog/dialog/client/pong
Use	A message sent from the dialog management
References	-

Sequence diagram

The following diagram (Figure 34) shows how the "DIALOG_CLIENT_PONG" and the dialog manager play together:

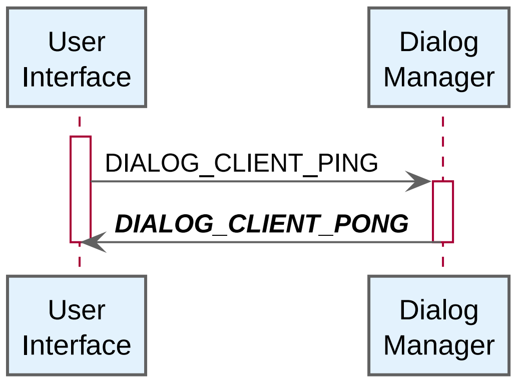


Figure 34. Sequence diagram of the DIALOG_CLIENT_PONG topic use.

Table 12. Properties of DIALOG_CLIENT_PONG

Property	Class	Description
CLIENT_ID	String	A string specifying the unique identifier of a client that sends/should receive a message
SOURCE_ID [1]	String	A string specifying the component that sent out the message
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC

[1] property is optional

DIALOG_CLIENT_PONG example

```
{
  "topic": "at/ac/ait/hbs/dialog/dialog/client/pong",
  "properties": {
    "mqttUser": "123456",
    "SOURCE_ID": "at.ac.ait.hbs.dialog.demo.server.package",
    "CLIENT_ID": "098697ce-cfba-4e8a-98ba-75bbc74c1c51",
    "mqttClientId": "da9d7c8d-555b-4bd1-ce5d-5d61dfd0eb91",
    "TIMESTAMP": 1584060514789
  }
}
```

5.3.5. DIALOG_CLIENT_STATUS_MESSAGE

This is a message that is sent to the dialog management from the output device containing information about the client's status

Table 13. Details of DIALOG_CLIENT_STATUS_MESSAGE

Name	DIALOG_CLIENT_STATUS_MESSAGE

Topic	at/ac/ait/hbs/dialog/dialog/client/status
Use	A message sent to the dialog management from the output device
References	-

Sequence diagram

The following diagram (Figure 35) shows how the "DIALOG_CLIENT_STATUS_MESSAGE", the dialog manager and underlying services play together:

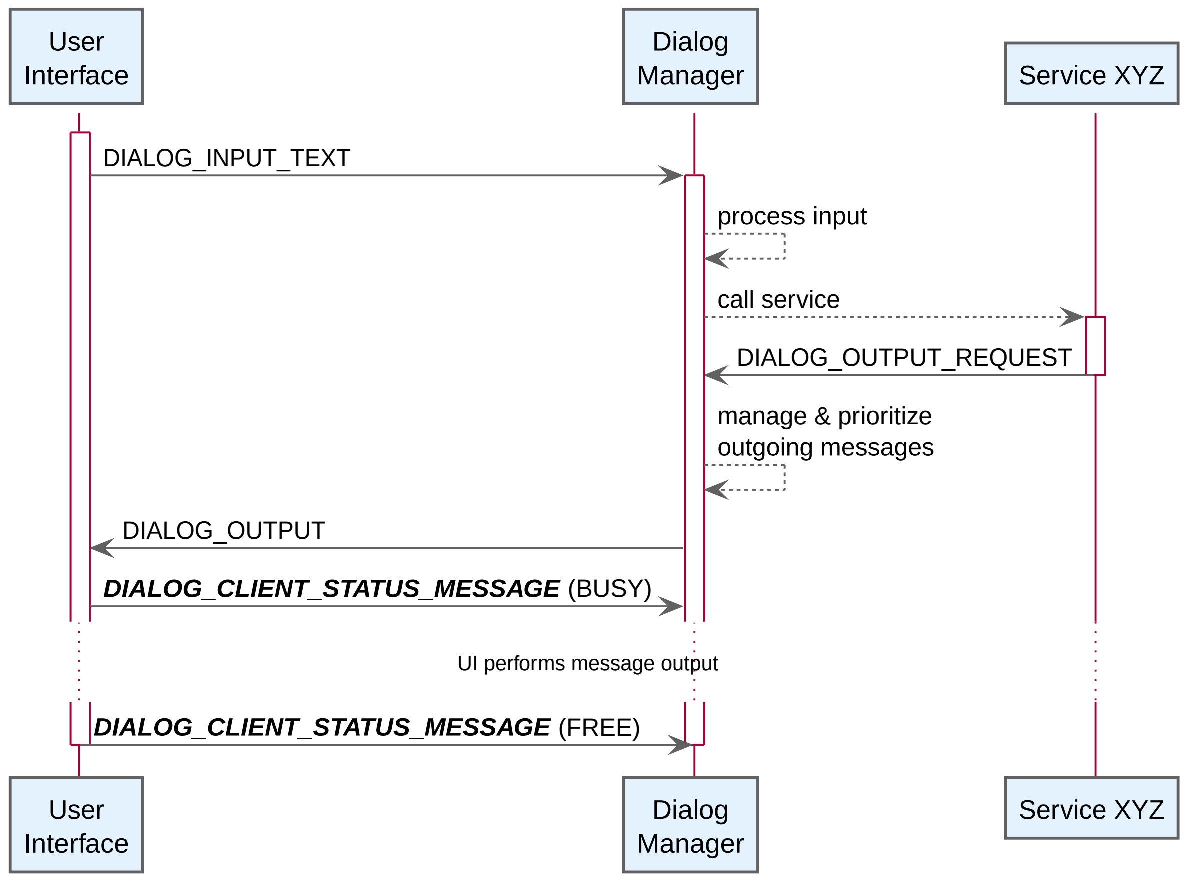


Figure 35. Sequence diagram of the DIALOG_CLIENT_STATUS_MESSAGE topic use.

Table 14. Properties of DIALOG_CLIENT_STATUS_MESSAGE

Property	**Class**	**Description**
CLIENT_ID	String	A string specifying the unique identifier of a client that sends/should receive a message
OUTPUT_STATUS	ClientOutputStatus	A enum describing the current status of the client in terms of the output modality
SOURCE_ID [1]	String	A string specifying the component that sent out the message

Property	Class	Description
OFFSET_TO_UTC	Integer	A long representing the offset to UTC in seconds
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC

[1] property is optional

DIALOG_CLIENT_STATUS_MESSAGE example

```
{
  "topic": "at/ac/ait/hbs/dialog/dialog/client/status",
  "properties": {
    "OFFSET_TO_UTC": 0,
    "mqttUser": "123456",
    "SOURCE_ID": "at.ac.ait.hbs.dialog.demo.client.package",
    "CLIENT_ID": "098697ce-cfba-4e8a-98ba-75bbc74c1c51",
    "OUTPUT_STATUS": "busy",
    "mqttClientId": "da9d7c8d-555b-4bd1-ce5d-5d61dfd0eb91",
    "TIMESTAMP": 1584060514790
  }
}
```

5.3.6. DIALOG_EVENT_TRIGGER

This is a message to trigger dialog management event with it’s event name. It contains the input for the dialog management model in written text form

Table 15. Details of DIALOG_EVENT_TRIGGER

Name	DIALOG_EVENT_TRIGGER
Topic	at/ac/ait/hbs/dialog/dialog/event/trigger
Use	A message to trigger dialog management event
References	-

Sequence diagram

The following diagram (Figure 36) shows how the "DIALOG_EVENT_TRIGGER", the dialog manager and underlying services play together:

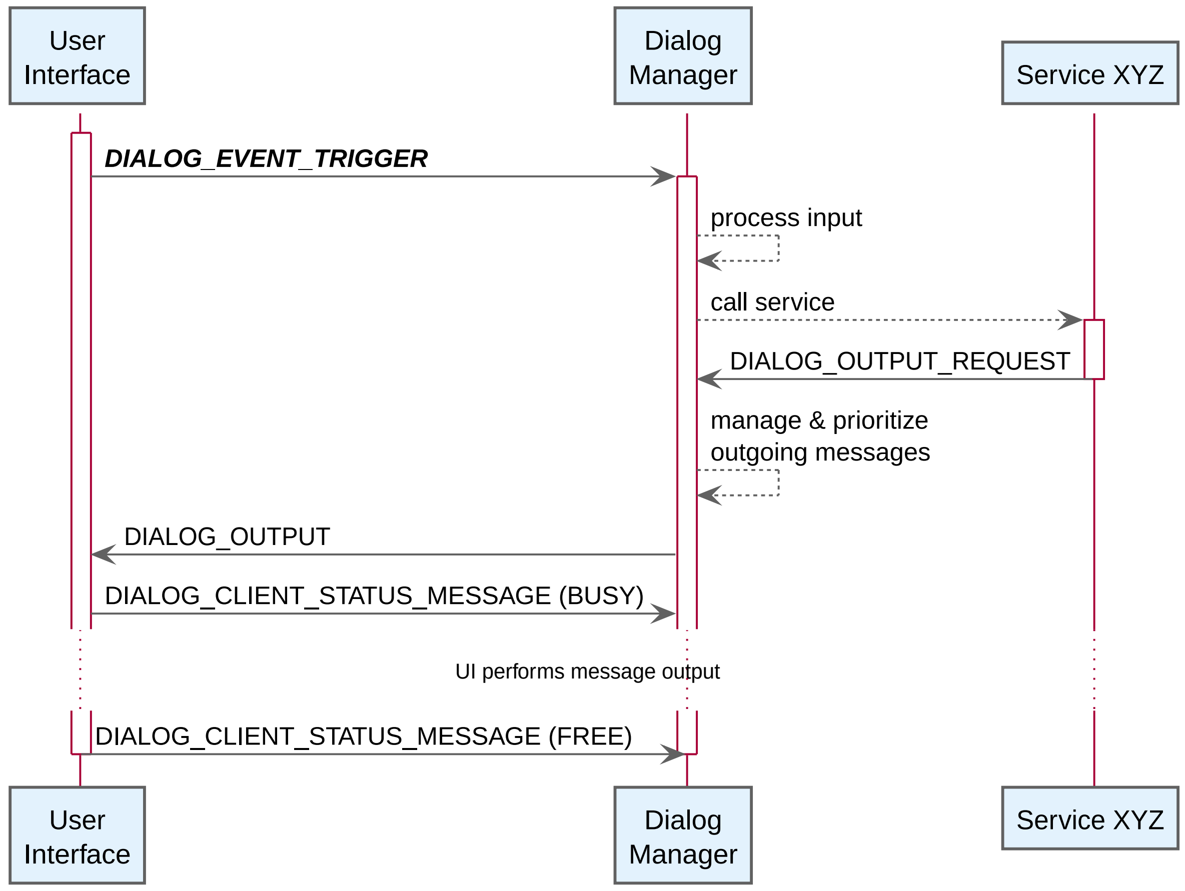


Figure 36. Sequence diagram of the DIALOG_EVENT_TRIGGER topic use.

Table 16. Properties of DIALOG_EVENT_TRIGGER

Property	**Class**	**Description**
CLIENT_ID	String	A string specifying the unique identifier of a client that sends/should receive a message
SESSION_ID [1]	String	A string specifying the unique identifier of a session the message belongs to
EVENT_NAME	String	A string representing the event name
LANGUAGE_CODE	LanguageCode	A enum describing the used language
PREFERENCE_EXTRAS [1,2]	Map	A map holding additional preferences
SOURCE_ID [1]	String	A string specifying the component that sent out the message
OFFSET_TO_UTC	Integer	A long representing the offset to UTC in seconds
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC

[1] property is optional
[2] value of the property can be null

DIALOG_EVENT_TRIGGER example

```
{
  "topic": "at/ac/ait/hbs/dialog/dialog/event/trigger",
  "properties": {
    "OFFSET_TO_UTC": 0,
    "LANGUAGE_CODE": "en",
    "mqttUser": "123456",
    "SOURCE_ID": "at.ac.ait.hbs.dialog.demo.client.package",
    "CLIENT_ID": "098697ce-cfba-4e8a-98ba-75bbc74c1c51",
    "mqttClientId": "da9d7c8d-555b-4bd1-ce5d-5d61dfd0eb91",
    "EVENT_NAME": "WEATHER_EVENT",
    "TIMESTAMP": 1584060514794
  }
}
```

5.3.7. DIALOG_INPUT_TEXT

This is a message that triggers the dialog management. It contains the input for the dialog management model in written text form

Table 17. Details of DIALOG_INPUT_TEXT

Name	DIALOG_INPUT_TEXT
Topic	at/ac/ait/hbs/dialog/dialog/input/text
Use	A message to trigger dialog management
References	-

Sequence diagram

The following diagram (Figure 37) shows how the "DIALOG_INPUT_TEXT", the dialog manager and underlying services play together:

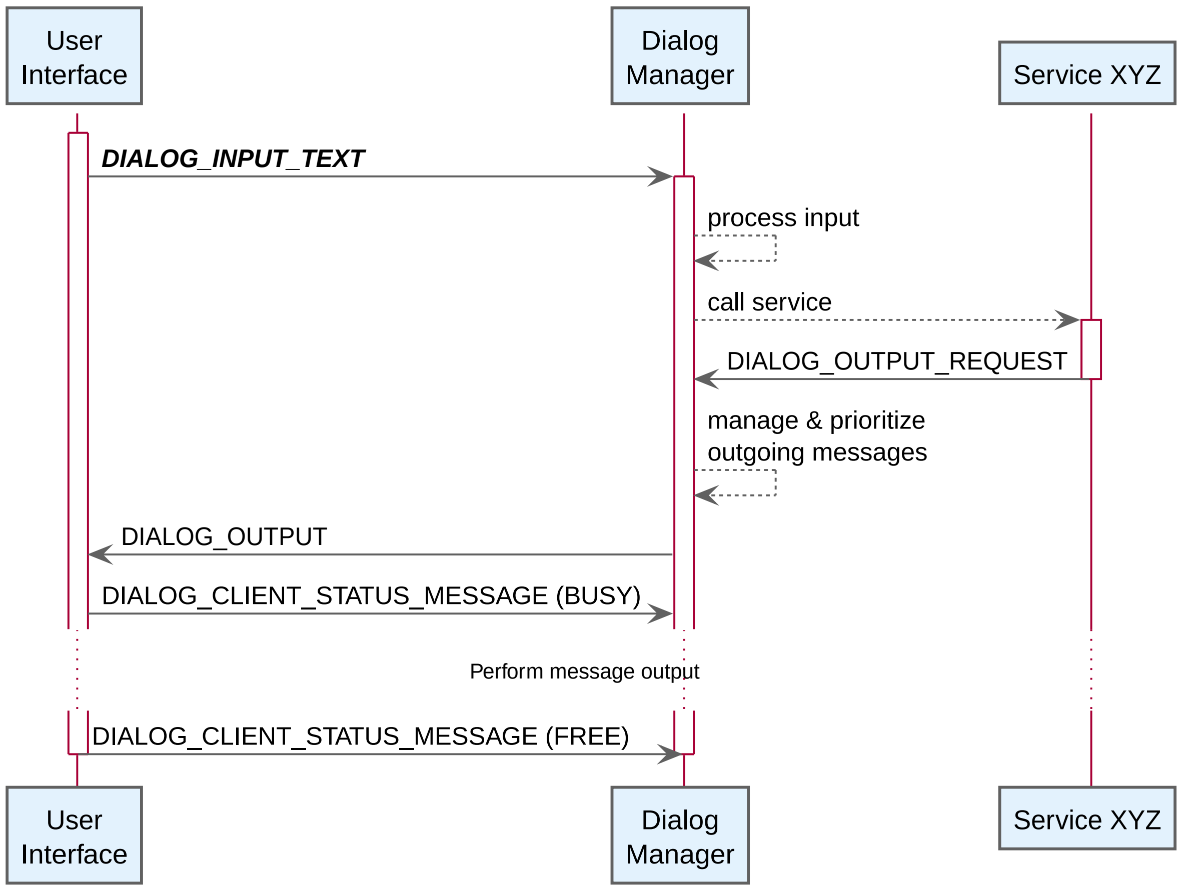


Figure 37. Sequence diagram of the DIALOG_INPUT_TEXT topic use.

Table 18. Properties of DIALOG_INPUT_TEXT

Property	Class	Description
CLIENT_ID	String	A string specifying the unique identifier of a client that sends/should receive a message
SESSION_ID [1]	String	A string specifying the unique identifier of a session the message belongs to
INPUT_TEXT	String	A string representing the textual input
LANGUAGE_CODE	LanguageCode	A enum describing the used language
PREFERENCE_EXTRAS [1,2]	Map	A map holding additional preferences
SOURCE_ID [1]	String	A string specifying the component that sent out the message
OFFSET_TO_UTC	Integer	A long representing the offset to UTC in seconds
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC

[1] property is optional
[2] value of the property can be null

DIALOG_INPUT_TEXT example

```
{
  "topic": "at/ac/ait/hbs/dialog/dialog/input/text",
  "properties": {
    "OFFSET_TO_UTC": 0,
    "INPUT_TEXT": "What is the weather today?",
    "LANGUAGE_CODE": "en",
    "mqttUser": "123456",
    "SOURCE_ID": "at.ac.ait.hbs.dialog.demo.client.package",
    "CLIENT_ID": "098697ce-cfba-4e8a-98ba-75bbc74c1c51",
    "mqttClientId": "da9d7c8d-555b-4bd1-ce5d-5d61dfd0eb91",
    "TIMESTAMP": 1584060514795
  }
}
```

5.3.8. DIALOG_OUTPUT

This is a message that is sent from dialog management to the output device with values for various modalities

Table 19. Details of DIALOG_OUTPUT

Name	DIALOG_OUTPUT
Topic	at/ac/ait/hbs/dialog/dialog/output
Use	A message sent from the dialog management to the output device
References	-

Sequence diagram

The following diagram (Figure 38) shows how the "DIALOG_OUTPUT", the dialog manager and underlying services play together:

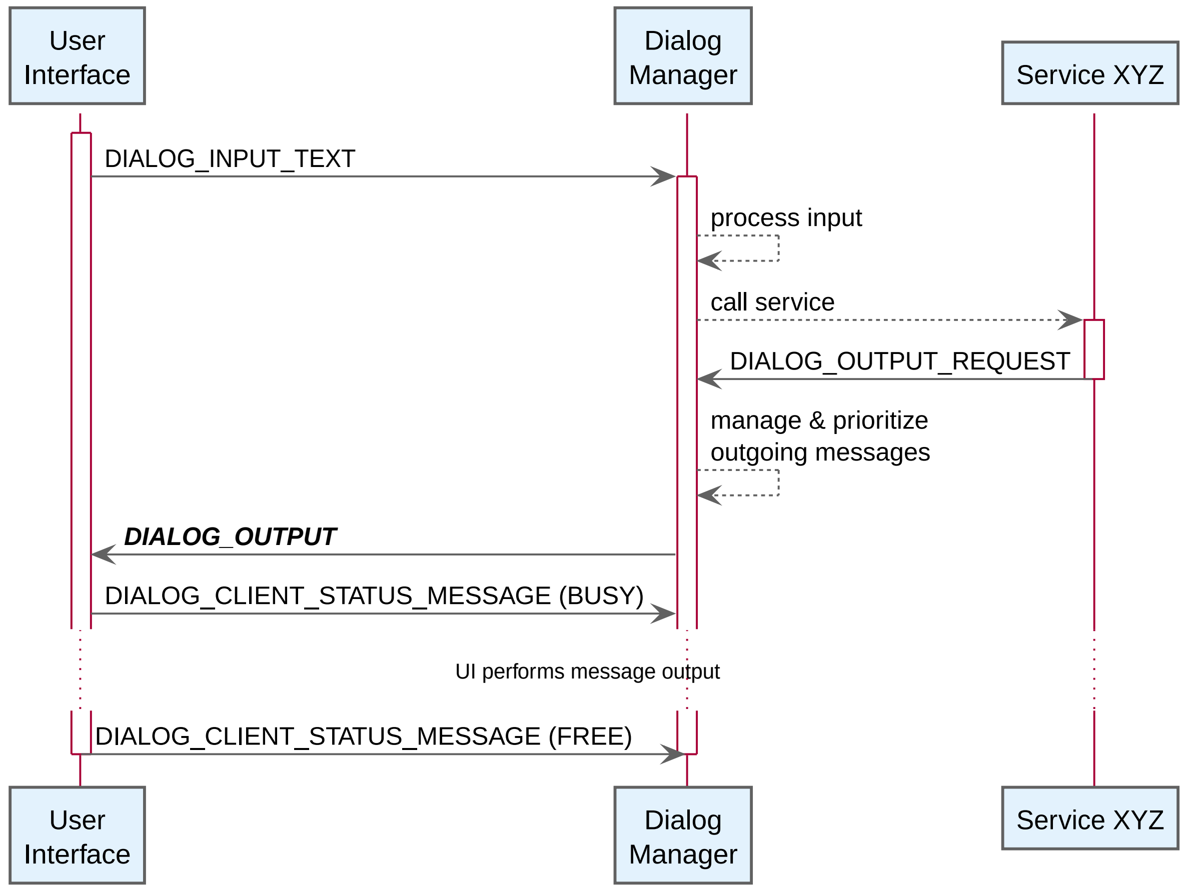


Figure 38. Sequence diagram of the DIALOG_OUTPUT topic use.

Table 20. Properties of DIALOG_OUTPUT

Property	**Class**	**Description**
CLIENT_ID	String	A string specifying the unique identifier of a client that sends/should receive a message
SESSION_ID [1]	String	A string specifying the unique identifier of a session the message belongs to
LANGUAGE_CODE	LanguageCode	A enum describing the used language
PRIORITY	Priority	A enum describing the message priority
OUTPUT_DISPLAY_TEXT [1]	String	A string representing the output as written text
OUTPUT_TEXT_TO_SPEECH	String	A string representing the output for the use with speech output
SOURCE_ID [1]	String	A string specifying the component that sent out the message
VALID_FOR [1,2]	Integer	A long representing the time in seconds how long the message should be valid

Property	Class	Description
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC

[1] property is optional
[2] value of the property can be null

DIALOG_OUTPUT example

```
{
  "topic": "at/ac/ait/hbs/dialog/dialog/output",
  "properties": {
    "LANGUAGE_CODE": "en",
    "mqttUser": "123456",
    "SOURCE_ID": "at.ac.ait.hbs.dialog.demo.server.package",
    "CLIENT_ID": "098697ce-cfba-4e8a-98ba-75bbc74c1c51",
    "mqttClientId": "da9d7c8d-555b-4bd1-ce5d-5d61dfd0eb91",
    "TIMESTAMP": 1584060514798,
    "OUTPUT_DISPLAY_TEXT": "The weather today is sunny.",
    "VALID_FOR": 1800,
    "PRIORITY": 1,
    "OUTPUT_TEXT_TO_SPEECH": "The weather today is sunny."
  }
}
```

5.3.9. DIALOG_OUTPUT_REQUEST

This is a message that is sent the dialog management with output for various modalities to be sent to the output device with values for various modalities

Table 21. Details of DIALOG_OUTPUT_REQUEST

Name	DIALOG_OUTPUT_REQUEST
Topic	at/ac/ait/hbs/dialog/dialog/output/request
Use	A message sent to the dialog management to be sent to the output device
References	-

Sequence diagram

The following (Figure 39) diagram shows how the "DIALOG_OUTPUT_REQUEST", the dialog manager and underlying services play together:

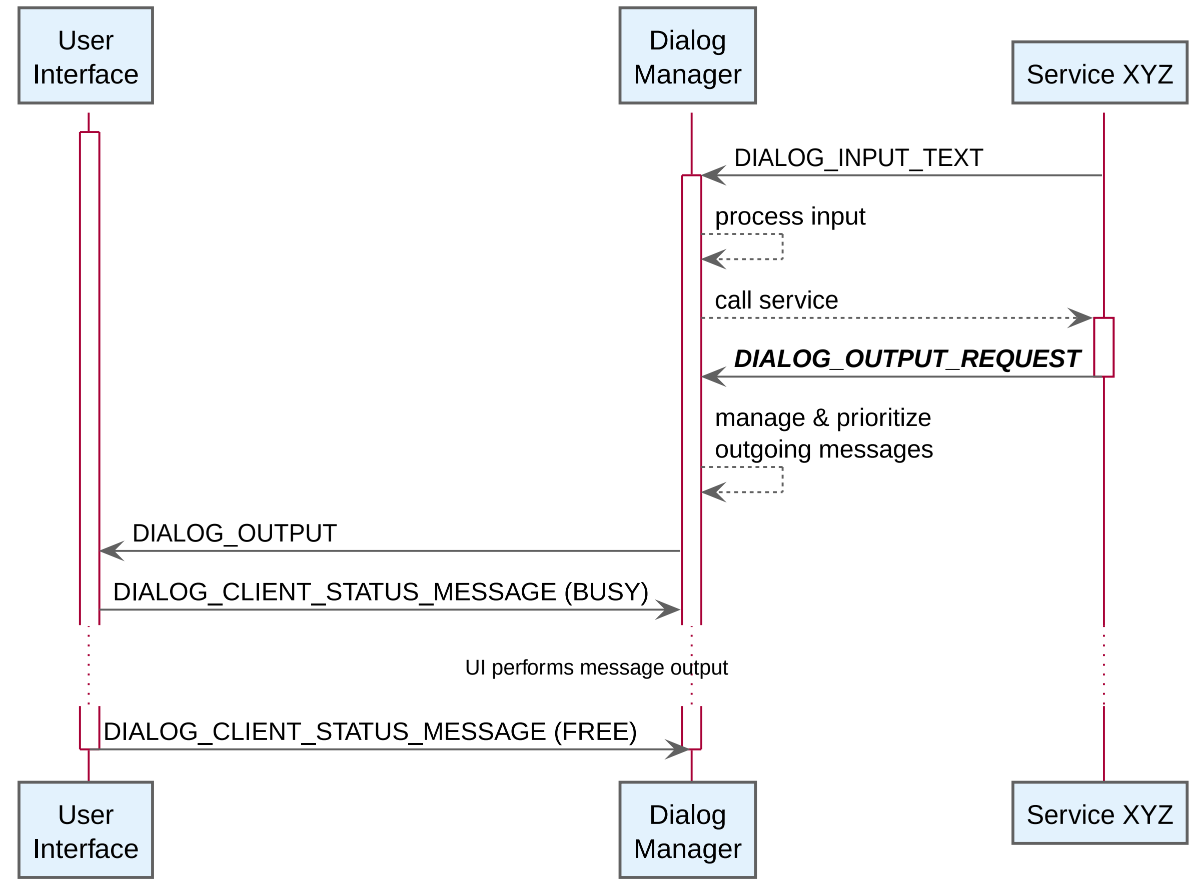


Figure 39. Sequence diagram of the DIALOG_OUTPUT_REQUEST topic use.

Table 22. Properties of DIALOG_OUTPUT_REQUEST

Property	**Class**	**Description**
CLIENT_ID	String	A string specifying the unique identifier of a client that sends/should receive a message
SESSION_ID [1]	String	A string specifying the unique identifier of a session the message belongs to
LANGUAGE_CODE	LanguageCode	A enum describing the used language
PRIORITY	Priority	A enum describing the message priority
OUTPUT_DISPLAY_TEXT [1]	String	A string representing the output as written text
OUTPUT_TEXT_TO_SPEECH	String	A string representing the output for the use with speech output
SOURCE_ID [1]	String	A string specifying the component that sent out the message
VALID_FOR [1,2]	Integer	A long representing the time in seconds how long the message should be valid

Property	Class	Description
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC

[1] property is optional
[2] value of the property can be null

DIALOG_OUTPUT_REQUEST example

```
{
  "topic": "at/ac/ait/hbs/dialog/dialog/output/request",
  "properties": {
    "LANGUAGE_CODE": "en",
    "mqttUser": "123456",
    "SOURCE_ID": "at.ac.ait.hbs.dialog.demo.server.package",
    "CLIENT_ID": "098697ce-cfba-4e8a-98ba-75bbc74c1c51",
    "mqttClientId": "da9d7c8d-555b-4bd1-ce5d-5d61dfd0eb91",
    "TIMESTAMP": 1584060514800,
    "OUTPUT_DISPLAY_TEXT": "The weather today is sunny.",
    "VALID_FOR": 1800,
    "PRIORITY": 1,
    "OUTPUT_TEXT_TO_SPEECH": "The weather today is sunny."
  }
}
```

5.4. EVENT CATEGORY "GAMING"

Event topics linked to this category are all related to gaming.

This category contains the topics as described in Table 23:

Table 23. Overview of the message topics discussed in this section.

GAME_RESULT	A message sent from the gaming module
GAME_SESSION_COMPLETED	A message sent from the gaming module
GAME_SESSION_PLANNED_REQUEST	A message sent to the UI professional module
GAME_SESSION_PLANNED_RESULT	A message sent from the UI professional module
GAME_SESSION_PLAYED_REQUEST	A message sent to the gaming module
GAME_SESSION_PLAYED_RESULT	A message sent from the gaming module

GAME_SESSION_START	A message sent to the gaming module
GAME_SESSION_STARTED	A message sent from the gaming module

5.4.1. GAME_RESULT

This is a message that is sent from the gaming module to indicate a specific game was finished

Table 24. Details of GAME_RESULT

Name	GAME_RESULT
Topic	eu/vcare/event/gaming/game/result
Use	A message sent from the gaming module
References	CS-1, CS-3

Table 25. Properties of GAME_RESULT

Property	**Class**	**Description**
CONTENT [2]	JsonElement	A JSON object describing the content
GAME_ID [1]	String	A string describing the game id
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
SESSION_TYPE	GameSessionType	A enum describing the game session type
SESSION_ID [1]	String	A string describing the game session id
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[1] property is optional
[2] value of the property can be null

GAME_RESULT example

```
{
  "topic": "eu/vcare/event/gaming/game/result",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472444,
    "USER_ID": "123456",
    "SESSION_TYPE": "cognitive",
    "GAME_ID": "135",
    "CONTENT": {
      "pg_id": "Plan game id as obtained from the loading of the plan",
      "session_id": "Actual session (Plan instance) ID",
      "status": "Can be \u0027complete\u0027, \u0027fail\u0027 or \u0027userexit
\u0027 to identify wheter the user complete succesfully the game, fails or just exit
without finish",
      "score": "The score obtained during the game",
      "results": {
        "description": "A custom JSON object with everything the developer want to track
about the game"
      },
      "tracking": {
        "description": "Optional, a JSON object containing the skeleton data tracked at
sampling rate"
      }
    }
  }
}
```

The tracking field shall contain:

- A list of points in space at the given sampling rate e.g. "right-hand": [[x, y, z], …]

  for specific joints like right-hand, left-hand, torso, right ankle or left-ankle there is also the possibility to have for each frame a json specifying other properties like

  ```
  {
    p: [x, y, z], # same as before
    s: 100, # the score obtained in that frame
    g: 1, # for left-hand and right hand identifies the grab gesture (c for the click)
  }
  ```

  Other available (mandatory) joints, as extracted from the medical needs' description, are left-shoulder, head, left-hand, right-knee, right-ankle, right-elbow, collar, right-hip, left-ankle, leftelbow, torso, right-shoulder, waist, left-knee, right-wrist, left-wrist, neck and left-hip

- root: The root node identifies the starting position of the user in device coordinates e.g. [26, -1220, 2402]
- f: The node f describes the sampling frequency e.g. 20

5.4.2. GAME_SESSION_COMPLETED

This is a message that is sent from the gaming module to indicate a specific session of games has been completed

Table 26. Details of GAME_SESSION_COMPLETED

Name	GAME_SESSION_COMPLETED
Topic	eu/vcare/event/gaming/session/completed
Use	A message sent from the gaming module
References	CS-1, CS-3

Table 27. Properties of GAME_SESSION_COMPLETED

Property	Class	Description
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
SESSION_TYPE	GameSessionType	A enum describing the game session type
SESSION_ID [1]	String	A string describing the game session id
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[1] property is optional
[2] value of the property can be null

GAME_SESSION_COMPLETED example

```
{
  "topic": "eu/vcare/event/gaming/session/completed",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472482,
    "USER_ID": "123456",
    "SESSION_TYPE": "cognitive",
    "CONTENT": {
      "exit_code": "Can be \"closed\", \"user_aborted\" or \"not_complete\" to identify how
the session has been closed"
    },
    "SESSION_ID": "567"
  }
}
```

5.4.3. GAME_SESSION_PLANNED_REQUEST

This is a message that is sent to the UI professional module to get a list of planned sessions related to a specific user

Table 28. Details of GAME_SESSION_PLANNED_REQUEST

Name	GAME_SESSION_PLANNED_REQUEST
Topic	eu/vcare/event/gaming/session/planned/request
Use	A message sent to the UI professional module
References	CS-1, CS-3

Table 29. Properties of GAME_SESSION_PLANNED_REQUEST

Property	**Class**	**Description**
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
SESSION_TYPE	GameSessionType	A enum describing the game session type
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[2] value of the property can be null

GAME_SESSION_PLANNED_REQUEST example

```
{
  "topic": "eu/vcare/event/gaming/session/planned/request",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472492,
    "USER_ID": "123456",
    "SESSION_TYPE": "cognitive",
    "CONTENT": {
      "filter_timestamp_start": 1551321273119,
      "filter_timestamp_end": 1551369273511
    }
  }
}
```

5.4.4. GAME_SESSION_PLANNED_RESULT

This is a message that is sent from the UI professional module as a response to a previous request related to a specific user

Table 30. Details of GAME_SESSION_PLANNED_RESULT

Name	GAME_SESSION_PLANNED_RESULT
Topic	eu/vcare/event/gaming/session/planned/result
Use	A message sent from the UI professional module
References	CS-1, CS-3

Table 31. Properties of GAME_SESSION_PLANNED_RESULT

Property	**Class**	**Description**
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
SESSION_TYPE	GameSessionType	A enum describing the game session type
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[2] value of the property can be null

GAME_SESSION_PLANNED_RESULT example

```
{
  "topic": "eu/vcare/event/gaming/session/planned/result",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472503,
    "USER_ID": "123456",
    "SESSION_TYPE": "cognitive",
    "CONTENT": {
      "sessions_planned": [
        {
          "session_id": "The session ID",
          "total": "Total count of games in the session",
          "games": [
            {
              "Plan_Game_ID": "Plan game id that identifies the game in the template plan",
              "Game_ID": "ID of the game type",
              "Name": "Title of the game"
            }
          ],
          "schedule": "JSON object describing the specified plan for this session"
        }
      ]
    }
  }
}
```

5.4.5. GAME_SESSION_PLAYED_REQUEST

This is a message that is sent to the gaming module to get a list of played sessions related to a specific user

Table 32. Details of GAME_SESSION_PLAYED_REQUEST

Name	GAME_SESSION_PLAYED_REQUEST
Topic	eu/vcare/event/gaming/session/played/request
Use	A message sent to the gaming module
References	CS-1, CS-3

Table 33. Properties of GAME_SESSION_PLAYED_REQUEST

Property	Class	Description
CONTENT [2]	JsonElement	A JSON object describing the content

Property	Class	Description
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
SESSION_TYPE	GameSessionType	A enum describing the game session type
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[2] value of the property can be null

GAME_SESSION_PLAYED_REQUEST example

```
{
  "topic": "eu/vcare/event/gaming/session/played/request",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472510,
    "USER_ID": "123456",
    "SESSION_TYPE": "cognitive",
    "CONTENT": {
      "filter_timestamp_start": 1551321273119,
      "filter_timestamp_end": 1551369273511
    }
  }
}
```

5.4.6. GAME_SESSION_PLAYED_RESULT

This is a message that is sent from the gaming module as a response to a previous request related to a specific user

Table 34. Details of GAME_SESSION_PLAYED_RESULT

Name	GAME_SESSION_PLAYED_RESULT
Topic	eu/vcare/event/gaming/session/played/result
Use	A message sent from the gaming module
References	CS-1, CS-3

Table 35. Properties of GAME_SESSION_PLAYED_RESULT

Property	Class	Description
CONTENT [2]	JsonElement	A JSON object describing the content

Property	Class	Description
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
SESSION_TYPE	GameSessionType	A enum describing the game session type
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[2] value of the property can be null

GAME_SESSION_PLAYED_RESULT example

```
{
  "topic": "eu/vcare/event/gaming/session/played/result",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472526,
    "USER_ID": "123456",
    "SESSION_TYPE": "cognitive",
    "CONTENT": {
      "sessions_played": [
        {
          "session_id": "The session ID",
          "total": "Total count of games in the session",
          "results": [
            {
              "Plan_Game_ID": "Plan game id that identifies the game in the template plan",
              "Status": "Can be toplay, complete or userexit identifies the final status of the
game",
              "Game_Start": "Start timestamp of the game (when patient click play button
after the briefing)",
              "Game_End": "End timestamp just when all the time elapsed",
              "Score": "Final score of the play",
              "Results": "JSON object containing the whole results of the play its different
for every game",
              "Game_ID": "ID of the game type",
              "Name": "Title of the game"
            }
          ]
        }
      ]
    }
  }
}
```

5.4.7. GAME_SESSION_START

This is a message that is sent to the gaming module to start a specific session of games

Table 36. Details of GAME_SESSION_START

Name	GAME_SESSION_START
Topic	eu/vcare/event/gaming/session/start
Use	A message sent to the gaming module

References	CS-1, CS-3

Table 37. Properties of GAME_SESSION_START

Property	Class	Description
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
SESSION_TYPE	GameSessionType	A enum describing the game session type
SESSION_ID [1]	String	A string describing the game session id
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[1] property is optional

GAME_SESSION_START example

```
{
  "topic": "eu/vcare/event/gaming/session/start",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472531,
    "USER_ID": "123456",
    "SESSION_TYPE": "cognitive",
    "SESSION_ID": "567"
  }
}
```

5.4.8. GAME_SESSION_STARTED

This is a message that is sent from the gaming module to indicate a specific session of games was started

Table 38. Details of GAME_SESSION_STARTED

Name	GAME_SESSION_STARTED
Topic	eu/vcare/event/gaming/session/started
Use	A message sent from the gaming module
References	CS-1, CS-3

Table 39. Properties of GAME_SESSION_STARTED

Property	Class	Description
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
SESSION_TYPE	GameSessionType	A enum describing the game session type
SESSION_ID [1]	String	A string describing the game session id
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[1] property is optional

GAME_SESSION_STARTED example

```
{
  "topic": "eu/vcare/event/gaming/session/started",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472532,
    "USER_ID": "123456",
    "SESSION_TYPE": "cognitive",
    "SESSION_ID": "567"
  }
}
```

5.5. EVENT CATEGORY "HEALTH"

Event topics linked to this category are all related to health services.

This category contains the topics as described in Table 40:

Table 40. Overview of the message topics discussed in this section.

HEALTH_QUESTIONNAIRE_REQUEST	A message sent to the coaching services
HEALTH_QUESTIONNAIRE_RESULT	A message sent from the coaching services
HEALTH_STATUS_MESSAGE	A message sent to the coaching services
HEALTH_STATUS_REQUEST	A message sent from the coaching services
HEALTH_THERAPY_ENDED	A message sent from the therapy module
HEALTH_THERAPY_START	A message sent to the therapy module

HEALTH_THERAPY_STARTED	A message sent from the therapy module
HEALTH_THERAPY_TASKS_REQUEST	A message sent to the KRF
HEALTH_THERAPY_TASKS_RESPONSE	A message sent from the KRF

5.5.1. HEALTH_QUESTIONNAIRE_REQUEST

This is a message that is sent to the coaching services to perform a questionnaire for a specific user

Table 41. Details of HEALTH_QUESTIONNAIRE_REQUEST

Name	HEALTH_QUESTIONNAIRE_REQUEST
Topic	eu/vcare/event/health/questionnaire/request
Use	A message sent to the coaching services
References	CS-7, CS-8

Table 42. Properties of HEALTH_QUESTIONNAIRE_REQUEST

Property	Class	Description
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[2] value of the property can be null

HEALTH_QUESTIONNAIRE_REQUEST example

```
{
  "topic": "eu/vcare/event/health/questionnaire/request",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472545,
    "USER_ID": "123456",
    "CONTENT": {
      "turns": [
        {
          "id": "Start",
          "content": "This the first question of a demo questionnaire",
          "info": "Some text can also be added as additional information",
          "infoTts": "Some information should only be provided by the text-to-speech
engine.",
          "question": "Why should I answer this questionnaire?",
          "questionTts": "Additional to the question \"Why should I answer this
questionnaire?\" it\u0027s possible to enrich it with TTS pronunciation tags",
          "type": "SINGLE",
          "choices": [
            {
              "content": "Just for fun! Give me all possible questions.",
              "reference": "Question1"
            },
            {
              "content": "I have to do it. I will only answer a few questions...",
              "reference": "Question2"
            },
            {
              "content": "That\u0027s the mandatory third option",
              "reference": "Question3"
            }
          ]
        },
        {
          "id": "Question1",
          "content": "Here you have to pick 3 out of 5...",
          "type": "MULTI",
          "choices": [
            {
              "content": "Answer number one"
            },
            {
              "content": "Answer number two"
            },
            {
              "content": "Answer number three"
            },
            {
              "content": "Answer number four"
            },
            {
              "content": "Answer number five"
            }
          ],
          "minAnswers": 3,
          "reference": "Question2",
          "score": false
        },
        {
          "id": "Question2",
          "question": "Please pick one answer",
          "type": "SINGLE",
          "choices": [
            {
              "content": "Answer number one"
            },
            {
              "content": "Answer number two"
            },
            {
              "content": "Answer number three"
            },
            {
              "content": "Answer number four"
            }
          ],
          "reference": "Question3",
          "score": true
        },
        {
          "id": "Question3",
          "question": "On a scale of 1 to 10, how do you feel today? (1 \u003d very bad, 5
\u003d as usual, 10 \u003d very good)",
          "type": "INT_SLIDER",
          "score": true,
          "slider": {
            "min": 1,
            "max": 10
          },
          "reference": "Question4"
        },
        {
          "id": "Question4",
          "precondition": "@{Question3} \u003c 5\n",
          "question": "Why do you feel bad today? Remember that you selected
@{Question3} in the previous question. Why didn\u0027t you select @[@{Question3} +
3]",
          "type": "STRING",
          "reference": "Question5"
        },
        {
          "id": "Question5",
          "precondition": "@{Question3} \u003e 5\n",
          "question": "Why do you feel good today? Remember that you selected
@{Question3} in the previous question. Why didn\u0027t you select @[@{Question3} -
4]",
          "type": "STRING",
          "reference": "End"
        },
        {
          "id": "End",
          "question": "We are done!",
          "type": "INFO",
          "score": false
        }
      ]
    }
  }
}
```

5.5.2. HEALTH_QUESTIONNAIRE_RESULT

This is a message that is sent from the coaching services with the result of a questionnaire for a specific user

Table 43. Details of HEALTH_QUESTIONNAIRE_RESULT

Name	HEALTH_QUESTIONNAIRE_RESULT
Topic	eu/vcare/event/health/questionnaire/result
Use	A message sent from the coaching services
References	CS-7, CS-8

Table 44. Properties of HEALTH_QUESTIONNAIRE_RESULT

Property	**Class**	**Description**
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[2] value of the property can be null

HEALTH_QUESTIONNAIRE_RESULT example

```
{
  "topic": "eu/vcare/event/health/questionnaire/result",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472567,
    "USER_ID": "123456",
    "CONTENT": {
      "turns": [
        {
          "id": "Start",
          "type": "SINGLE",
          "answer": 2
        },
        {
          "id": "Question1",
          "type": "MULTI",
          "answers": "1, 3, 5"
        },
        {
          "id": "Question2",
          "type": "SINGLE",
          "answer": 4
        },
        {
          "id": "Question3",
          "type": "INT_SLIDER",
          "answer": 2
        },
        {
          "id": "Question4",
          "type": "STRING",
          "answer": "I\u0027ve no clue."
        }
      ]
    }
  }
}
```

5.5.3. HEALTH_STATUS_MESSAGE

This is a message that is sent to the coaching services with the health status for a specific user

Table 45. Details of HEALTH_STATUS_MESSAGE

Name	HEALTH_STATUS_MESSAGE
Topic	eu/vcare/event/health/status/message
Use	A message sent to the coaching services
References	CS-2, DS-5, DS-6, DS-7, DS-9

Sequence diagram

The following diagram (Figure 40) shows the intended use of the "HEALTH_STATUS_MESSAGE".

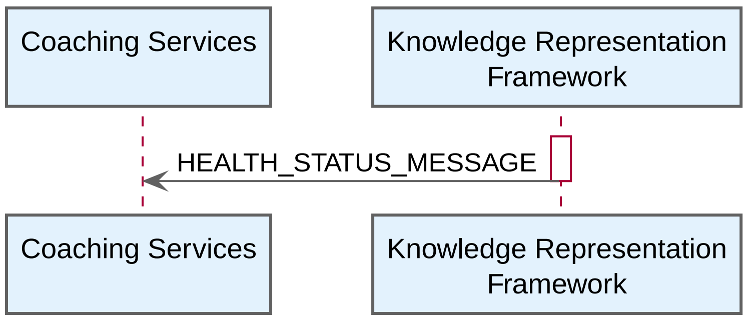

Coaching Services
Knowledge Representation Framework
HEALTH_STATUS_MESSAGE
Coaching Services
Knowledge Representation Framework

Figure 40. Sequence diagram of the HEALTH_STATUS_MESSAGE topic use.

Table 46. Properties of HEALTH_STATUS_MESSAGE

Property	**Class**	**Description**
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[2] value of the property can be null

HEALTH_STATUS_MESSAGE example

```
{
  "topic": "eu/vcare/event/health/status/message",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472574,
    "USER_ID": "123456",
    "CONTENT": [
      {
        "Type": "BMI",
        "Value": "22",
        "Unit": "kg/m^2",
        "Reward": 1
      }
    ]
  }
}
```

5.5.4. HEALTH_STATUS_REQUEST

This is a message that is sent from the coaching services to request the health status for a specific user

Table 47. Details of HEALTH_STATUS_REQUEST

Name	HEALTH_STATUS_REQUEST
Topic	eu/vcare/event/health/status/request
Use	A message sent from the coaching services
References	CS-2, DS-5, DS-6, DS-7, DS-9

Sequence diagram

The following diagram (Figure 41) shows how the "HEALTH_STATUS_REQUEST" and "HEALTH_STATUS_MESSAGE" play together.

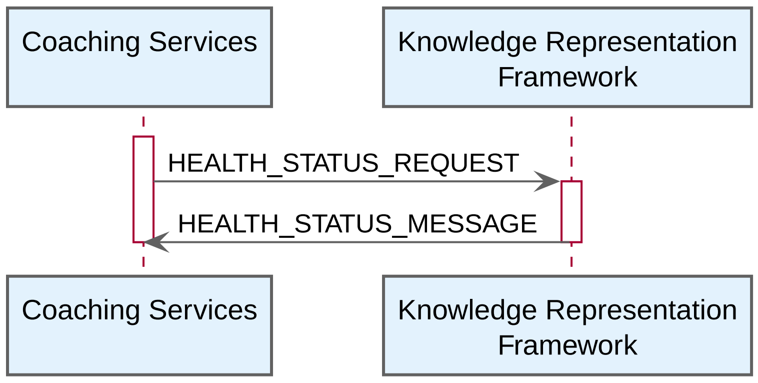


Figure 41. Sequence diagram of the HEALTH_STATUS_REQUEST topic use and interplay.

Table 48. Properties of HEALTH_STATUS_REQUEST

Property	Class	Description
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[2] value of the property can be null

HEALTH_STATUS_REQUEST example

```
{
  "topic": "eu/vcare/event/health/status/request",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472578,
    "USER_ID": "123456",
    "CONTENT": {
      "Types": [
        "BMI"
      ]
    }
  }
}
```

5.5.5. HEALTH_THERAPY_ENDED

This is a message that is sent from the therapy module to indicate a specific therapy was ended

Table 49. Details of HEALTH_THERAPY_ENDED

Name	HEALTH_THERAPY_ENDED
Topic	eu/vcare/event/health/therapy/ended
Use	A message sent from the therapy module
References	CS-9

This event has no properties.

HEALTH_THERAPY_ENDED example

```
{
  "topic": "eu/vcare/event/health/therapy/ended",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "THERAPY_TYPE": "speech-swallowing",
    "THERAPY_ID": "890",
    "TIMESTAMP": 1584106472588,
    "USER_ID": "123456"
  }
}
```

5.5.6. HEALTH_THERAPY_START

This is a message that is sent to the therapy module to start a specific therapy

Table 50. Details of HEALTH_THERAPY_START

Name	HEALTH_THERAPY_START
Topic	eu/vcare/event/health/therapy/start
Use	A message sent to the therapy module
References	CS-9

Table 51. Properties of HEALTH_THERAPY_START

Property	**Class**	**Description**
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
THERAPY_TYPE	TherapyType	A enum describing the therapy type
THERAPY_ID [1]	String	A string describing the game id
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC

Property	Class	Description
USER_ID	String	A string specifying a user id

[1] property is optional

HEALTH_THERAPY_START example

```
{
  "topic": "eu/vcare/event/health/therapy/start",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "THERAPY_TYPE": "speech-swallowing",
    "THERAPY_ID": "890",
    "TIMESTAMP": 1584106472592,
    "USER_ID": "123456"
  }
}
```

5.5.7. HEALTH_THERAPY_STARTED

This is a message that is sent from the therapy module to indicate a specific therapy was started

Table 52. Details of HEALTH_THERAPY_STARTED

Name	HEALTH_THERAPY_STARTED
Topic	eu/vcare/event/health/therapy/started
Use	A message sent from the therapy module
References	CS-9

Table 53. Properties of HEALTH_THERAPY_STARTED

Property	Class	Description
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
THERAPY_TYPE	TherapyType	A enum describing the therapy type
THERAPY_ID [1]	String	A string describing the game id
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[1] property is optional

HEALTH_THERAPY_STARTED example

```
{
  "topic": "eu/vcare/event/health/therapy/started",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "THERAPY_TYPE": "speech-swallowing",
    "THERAPY_ID": "890",
    "TIMESTAMP": 1584106472600,
    "USER_ID": "123456"
  }
}
```

5.5.8. HEALTH_THERAPY_TASKS_REQUEST

This is a message that is sent to the KRF to retrieve a list of pathway related tasks for a patient

Table 54. Details of HEALTH_THERAPY_TASKS_REQUEST

Name	HEALTH_THERAPY_TASKS_REQUEST
Topic	eu/vcare/event/health/therapy/tasks/request
Use	A message sent to the KRF
References	CS-1, CS-3, CS-6

Table 55. Properties of HEALTH_THERAPY_TASKS_REQUEST

Property	**Class**	**Description**
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC

HEALTH_THERAPY_TASKS_REQUEST example

```
{
  "topic": "eu/vcare/event/health/therapy/tasks/request",
  "properties": {
    "TIMESTAMP": 1583836342656,
    "USER_ID": "123456"
  }
}
```

5.5.9. HEALTH_THERAPY_TASKS_RESPONSE

This is a message that is sent from the KRF containing a list of pathway related tasks for a patient

Table 56. Details of HEALTH_THERAPY_TASKS_RESPONSE

Name	HEALTH_THERAPY_TASKS_RESPONSE
Topic	eu/vcare/event/health/therapy/tasks/response
Use	A message sent from the KRF
References	CS-1, CS-3, CS-6

Table 57. Properties of HEALTH_THERAPY_TASKS_RESPONSE

Property	Class	Description
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC

HEALTH_THERAPY_TASKS_RESPONSE example

```
{
  "topic": "eu/vcare/event/health/therapy/tasks/response",
  "properties": {
    "TIMESTAMP": 1583836342656,
    "USER_ID": "123456",
    "CONTENT": {
      "tasks": [
        {
          "id": "MOTOR_EXERCISE_123",
          "type": "rehability_game"
        },
        {
          "id": "RF_NUTRITION_WEIGHT",
          "type": "elearning"
        }
      ]
    }
  }
}
```

5.6. EVENT CATEGORY "LOCATION"

Event topics linked to this category are all related to the location of a user.

This category contains the topics as described in Table 58:

Table 58. Overview of the message topics discussed in this section.

LOCATION_MESSAGE	A message sent from the data services
LOCATION_REQUEST	A message sent to the data services

5.6.1. LOCATION_MESSAGE

This is a message that is sent from the data services with the location for a specific user

Table 59. Details of LOCATION_MESSAGE

Name	LOCATION_MESSAGE
Topic	eu/vcare/event/location/message
Use	A message sent from the data services
References	DS-1, DS-8

Table 60. Properties of LOCATION_MESSAGE

Property	Class	Description
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[2] value of the property can be null

LOCATION_MESSAGE example

```
{
  "topic": "eu/vcare/event/location/message",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472727,
    "USER_ID": "123456",
    "CONTENT": [
      {
        "Type": "GPS",
        "Lat": "N48Â°16\u002703.60\"",
        "Lon": "E16Â°22\u002717.00\""
      }
    ]
  }
}
```

5.6.2. LOCATION_REQUEST

This is a message that is sent to the data services to request the location for a specific user

Table 61. Details of LOCATION_REQUEST

Name	LOCATION_REQUEST
Topic	eu/vcare/event/location/request
Use	A message sent to the data services
References	DS-1, DS-8

Table 62. Properties of LOCATION_REQUEST

Property	Class	Description
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[2] value of the property can be null

LOCATION_REQUEST example

```
{
  "topic": "eu/vcare/event/location/request",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472741,
    "USER_ID": "123456",
    "CONTENT": {
      "Types": [
        "GPS"
      ]
    }
  }
}
```

5.7. EVENT CATEGORY "MESSAGE"

Event topics linked to this category are all related to general messages.

This category contains the topics as described in Table 63:

Table 63. Overview of the message topics discussed in this section.

MOTIVATION_TEXT_MESSAGE	A message sent to the requesting coaching services

ELEARNING_CONTENT_MESSAGE	A message sent to the e-learning coaching services
STANDBY_MESSAGE	A message to notify modules about the standby mode
MOTIVATION_TEXT_REQUEST	A message sent to the motivational message service
MOTIVATION_OUTPUT_REQUEST	A message sent to the motivational message module to directly generate an output message
ELEARNING_CONTENT_REQUEST	A message sent to the knowledge representation layer
USER_FEELING_STATUS_MESSAGE	A message sent to the knowledge representation framework
USER_MESSAGE	A message sent to the coaching services

5.7.1. MOTIVATION_TEXT_MESSAGE

This is a message that is sent to the service which requested a motivational message

Table 64. Details of MOTIVATION_TEXT_MESSAGE

Name	MOTIVATION_TEXT_MESSAGE
Topic	eu/vcare/event/message/service/message
Use	A message sent to the requesting coaching services
References	CS-2, CS-5

This event has no properties.

MOTIVATION_TEXT_MESSAGE example

```
{
  "topic": "eu/vcare/event/message/service/message",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472628,
    "USER_ID": "123456",
    "CONTENT": {
      "message": "Everyone can have a bad day. You trend is positive, keep on training."
    }
  }
}
```

5.7.2. ELEARNING_CONTENT_MESSAGE

This is a message that is sent to the e-learning coaching service to trigger an e-learning session with specific content

Table 65. Details of ELEARNING_CONTENT_MESSAGE

Name	ELEARNING_CONTENT_MESSAGE
Topic	eu/vcare/event/message/service/message
Use	A message sent to the e-learning coaching services
References	CS-4

This event has no properties.

ELEARNING_CONTENT_MESSAGE example

```
{
  "topic": "eu/vcare/event/message/service/message",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472634,
    "USER_ID": "123456",
    "CONTENT": {
      "content_type": "low_salt_diet",
      "content_description": "Information why a low salt diet is important for ischaemic
heart disease.",
      "video_id": 1135
    }
  }
}
```

5.7.3. STANDBY_MESSAGE

This is a message that is sent communicationg modules to inform about an enabled/disabled standby mode.

Table 66. Details of STANDBY_MESSAGE

Name	STANDBY_MESSAGE
Topic	eu/vcare/event/message/service/message
Use	A message to notify modules about the standby mode
References	SS-3

This event has no properties.

5.7.4. MOTIVATION_TEXT_REQUEST

This is a message that is sent to the motivational message service for generating a context related message for the user

Table 67. Details of MOTIVATION_TEXT_REQUEST

Name	MOTIVATION_TEXT_REQUEST
Topic	eu/vcare/event/message/service/request
Use	A message sent to the motivational message service
References	CS-2, CS-5

This event has no properties.

MOTIVATION_TEXT_REQUEST example

```
{
  "topic": "eu/vcare/event/message/service/request",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472641,
    "USER_ID": "123456",
    "CONTENT": {
      "context": "physical_activity",
      "performance": "medium"
    }
  }
}
```

5.7.5. MOTIVATION_OUTPUT_REQUEST

This is a message that is sent to the motivational message module which generates an output message directly based on the data and context provided.

Table 68. Details of MOTIVATION_OUTPUT_REQUEST

Name	MOTIVATION_OUTPUT_REQUEST
Topic	eu/vcare/event/message/service/request
Use	A message sent to the motivational message module to directly generate an output message
References	CS-5

This event has no properties.

MOTIVATION_OUTPUT_REQUEST example

```
{
  "topic": "eu/vcare/event/message/service/request",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472645,
    "USER_ID": "123456",
    "CONTENT": {
      "context": "physical_training",
      "content": "increase_intensity"
    }
  }
}
```

5.7.6. ELEARNING_CONTENT_REQUEST

This is a message that is sent to the knowledge representation layer to retrieve content for an e-learning session for a specific user

Table 69. Details of ELEARNING_CONTENT_REQUEST

Name	ELEARNING_CONTENT_REQUEST
Topic	eu/vcare/event/message/service/request
Use	A message sent to the knowledge representation layer
References	CS-4

This event has no properties.

ELEARNING_CONTENT_REQUEST example

```
{
  "topic": "eu/vcare/event/message/service/request",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472647,
    "USER_ID": "123456"
  }
}
```

5.7.7. USER_FEELING_STATUS_MESSAGE

This is a message that is sent to the knowledge representation framework to store the user's subjective feeling

Table 70. Details of USER_FEELING_STATUS_MESSAGE

Name	USER_FEELING_STATUS_MESSAGE
Topic	eu/vcare/event/message/status/message
Use	A message sent to the knowledge representation framework
References	CS-8

This event has no properties.

USER_FEELING_STATUS_MESSAGE example

```
{
  "topic": "eu/vcare/event/message/status/message",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472659,
    "USER_ID": "123456",
    "CONTENT": {
      "status_description": "dizzy",
      "status_value": 3
    }
  }
}
```

5.7.8. USER_MESSAGE

This is a message that is sent to the coaching services to inform the user

Table 71. Details of USER_MESSAGE

Name	USER_MESSAGE
Topic	eu/vcare/event/message/user
Use	A message sent to the coaching services
References	CS-4

Table 72. Properties of USER_MESSAGE

Property	Class	Description
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC

Property	Class	Description
USER_ID	String	A string specifying a user id

[2] value of the property can be null

USER_MESSAGE example

```
{
  "topic": "eu/vcare/event/message/user",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472663,
    "USER_ID": "123456",
    "CONTENT": {
      "content": "not defined yet"
    }
  }
}
```

5.8. EVENT CATEGORY "POSITION"

Event topics linked to this category are all related to the position of a user.

This category contains the topics as described in Table 73:

Table 73. Overview of the message topics discussed in this section.

POSITION_MESSAGE	A message sent from the data services
POSITION_REQUEST	A message sent to the data services

5.8.1. POSITION_MESSAGE

This is a message that is sent from the data services with the position for a specific user

Table 74. Details of POSITION_MESSAGE

Name	POSITION_MESSAGE
Topic	eu/vcare/event/position/message
Use	A message sent from the data services
References	DS-2, DS-4

Table 75. Properties of POSITION_MESSAGE

Property	Class	Description
CONTENT [2]	JsonElement	A JSON object describing the content

Property	Class	Description
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[2] value of the property can be null

POSITION_MESSAGE example

```
{
  "topic": "eu/vcare/event/position/message",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472748,
    "USER_ID": "123456",
    "CONTENT": [
      {
        "Type": "Body",
        "Position": "Standing"
      }
    ]
  }
}
```

5.8.2. POSITION_REQUEST

This is a message that is sent to the data services to request the position for a specific user

Table 76. Details of POSITION_REQUEST

Name	POSITION_REQUEST
Topic	eu/vcare/event/position/request
Use	A message sent to the data services
References	DS-2, DS-4

Table 77. Properties of POSITION_REQUEST

Property	Class	Description
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests

Property	Class	Description
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[2] value of the property can be null

POSITION_REQUEST example

```
{
  "topic": "eu/vcare/event/position/request",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472752,
    "USER_ID": "123456",
    "CONTENT": {
      "Types": [
        "Body"
      ]
    }
  }
}
```

5.9. EVENT CATEGORY "REMINDER"

Event topics linked to this category are all related to reminders.

This category contains the topics as described in Table 78:

Table 78. Overview of the message topics discussed in this section.

REMINDER_MESSAGE	A message sent to the coaching services
REMINDER_RESPONSE	A message sent back to the source of the reminder message
REMINDER_POSTPONE	A message sent from the coaching services

5.9.1. REMINDER_MESSAGE

This is a message that is sent to the coaching services to remind the user e.g. about a specific task

Table 79. Details of REMINDER_MESSAGE

Name	REMINDER_MESSAGE
Topic	eu/vcare/event/reminder/message

Use	A message sent to the coaching services
References	CS-1, CS-3, CS-6, CS-8, CS-9

Table 80. Properties of REMINDER_MESSAGE

Property	**Class**	**Description**
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
REMINDER_TYPE	ReminderType	A enum describing the reminder type
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[2] value of the property can be null

REMINDER_MESSAGE example

```
{
  "topic": "eu/vcare/event/reminder/message",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472673,
    "USER_ID": "123456",
    "CONTENT": {
      "content": "not defined yet"
    },
    "REMINDER_TYPE": "medication"
  }
}
```

5.9.2. REMINDER_RESPONSE

This is a message that is sent back to the origin of the reminder message and contains an optional confirmation or reply from the user.

Table 81. Details of REMINDER_RESPONSE

Name	REMINDER_RESPONSE
Topic	eu/vcare/event/reminder/message
Use	A message sent back to the source of the reminder message

References	CS-1, CS-3, CS-6, CS-8, CS-9

This event has no properties.

REMINDER_RESPONSE example

```
{
  "topic": "eu/vcare/event/reminder/message",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472688,
    "USER_ID": "123456",
    "CONTENT": {
      "Type": "Acknowledgement",
      "Value": "True"
    }
  }
}
```

5.9.3. REMINDER_POSTPONE

This is a message that is sent from the coaching services to indicate that a reminder should be postponed

Table 82. Details of REMINDER_POSTPONE

Name	REMINDER_POSTPONE
Topic	eu/vcare/event/reminder/postpone
Use	A message sent from the coaching services
References	CS-1, CS-3, CS-6, CS-9

Table 83. Properties of REMINDER_POSTPONE

Property	Class	Description
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
REMINDER_TYPE	ReminderType	A enum describing the reminder type
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

2 value of the property can be null

REMINDER_POSTPONE example

```
{
  "topic": "eu/vcare/event/reminder/postpone",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472691,
    "USER_ID": "123456",
    "REMINDER_TYPE": "medication"
  }
}
```

5.10. EVENT CATEGORY "SUPPORT"

Event topics linked to this category are all related to the supporting services.

This category contains the topics as described in Table 84:

Table 84. Overview of the message topics discussed in this section.

SUPPORT_SERVICE_MESSAGE	A message sent from the supporting services
SUPPORT_SERVICE_REQUEST	A message sent to the supporting services

5.10.1. SUPPORT_SERVICE_MESSAGE

This is a message that is sent from the supporting services with data for a specific user

Table 85. Details of SUPPORT_SERVICE_MESSAGE

Name	SUPPORT_SERVICE_MESSAGE
Topic	eu/vcare/event/support/service/message
Use	A message sent from the supporting services
References	SS-1, SS-2, SS-3

Table 86. Properties of SUPPORT_SERVICE_MESSAGE

Property	Class	Description
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC

Property	Class	Description
USER_ID	String	A string specifying a user id

[2] value of the property can be null

SUPPORT_SERVICE_MESSAGE example

```
{
  "topic": "eu/vcare/event/support/service/message",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472710,
    "USER_ID": "123456",
    "CONTENT": [
      {
        "Type": "WeatherForecastToday",
        "Data": {
          "coord": {
            "lon": 139,
            "lat": 35
          },
          "sys": {
            "country": "JP",
            "sunrise": 1369769524,
            "sunset": 1369821049
          },
          "weather": [
            {
              "id": 804,
              "main": "clouds",
              "description": "overcast clouds",
              "icon": "04n"
            }
          ],
          "main": {
            "temp": 289.5,
            "humidity": 89,
            "pressure": 1013,
            "temp_min": 287.04,
            "temp_max": 292.04
          },
          "wind": {
            "speed": 7.31,
            "deg": 187.002
          },
        "rain": {
          "3h": 0
        },
        "clouds": {
          "all": 92
        },
        "dt": 1369824698,
        "id": 1851632,
        "name": "Vienna",
        "cod": 200
      }
    }
  ]
 }
}
```

5.10.2. SUPPORT_SERVICE_REQUEST

This is a message that is sent to the supporting services to request data for a specific user

Table 87. Details of SUPPORT_SERVICE_REQUEST

Name	SUPPORT_SERVICE_REQUEST
Topic	eu/vcare/event/support/service/request
Use	A message sent to the supporting services
References	SS-1, SS-2, SS-3

Table 88. Properties of SUPPORT_SERVICE_REQUEST

Property	**Class**	**Description**
CONTENT [2]	JsonElement	A JSON object describing the content
INTERACTION_ID	String	A string specifying an interaction id to match responses to requests
TIMESTAMP	Long	A timestamp representing the number of seconds since 01.01.1970 00:00:00 UTC
USER_ID	String	A string specifying a user id

[2] value of the property can be null

SUPPORT_SERVICE_REQUEST example

```
{
  "topic": "eu/vcare/event/support/service/request",
  "properties": {
    "INTERACTION_ID": "155742af-4942-4142-8b81-184d2edd1393",
    "TIMESTAMP": 1584106472718,
    "USER_ID": "123456",
    "CONTENT": {
      "Types": [
        "WeatherForecastToday"
      ]
    }
  }
}
```

5.11. EVENT CATEGORY "SYSTEM"

Event topics linked to this category are all related to intra system messages.

This category contains no topics.

6. CONCLUSIONS AND OUTLOOK

This document provides a more detailed technical description of the coaching and supporting services as initially defined in D5.1. This technical description forms the base for the ongoing development until project month 36 and start of the lab trials. The development phase is supported by the tech labs which aim at an iterative development approach with multiple release cycles. One cycle lasts one month and at the end of each month, a new version of the services with new features will be released during the tech lab phase. The next objective pursued followed by this document is to establish the inter-module communication based on the messages standard as defined herein in terms of the protocol (MQTT) and message format (JSON). For doing so, an open-source MQTT broker (e.g. [RabbitMQ]) will be set up and made available to all partners. Additionally, the authentication process using KeyCloak as central authentication broker will be realized the same way as for the MQTT broker. Although the inter-module communication has been discussed very deeply among the technical partners, some shortcomings might appear during the tech-lab phase and may lead to an update of this document either in terms of some technical details, or additional messages and/or additional topics. However, no fundamental or structural change of the document is expected and the current document content will remain valid. Since parts of this document are automatically derived from the source code of the coaching services layer implementation, this update procedure is a very straight forward and logical procedure and is inherent in every new release of the vCare system.

As soon as the above-mentioned features will have been implemented and tested, the system will be successively enriched by adding one service after another with the full-featured vCare system is expected to be available by project month 36. To achieve this goal, a detailed release plan which is monitored weekly has been set up. For the first upcoming release, the services CS-1 - *Physical training*, CS-2 - *Health Status*, CS-6 - *Intelligent Notifications / Scheduler* as well as SS-1 - *Weather Information* and SS-3 *Standby* will be made available in a first version each.

The results of the tech labs phase and the following living lab phase will be reported in the deliverable D5.4 *Coaching services test and validation report* to be delivered by month 52.

REFERENCES

[Apache Soft. Found.] The Apache Karaf Manual. http://karaf.apache.org/manual/latest

[Edstrom et al, 2013] Edstrom J., Goodyear J., Kesler H. (2013). Learning Apache Karaf. Packt Publishing Ltd

[Kyriazakos et al, 2018] Kyriazakos, S., Valentini, V., Cesario, A., & Zachariae, R. (2018). FORECAST–A cloud-based personalized intelligent virtual coaching platform for the well-being of cancer patients. Clinical and translational radiation oncology, 8, 50-59.

[iSprint, 2017] The New Era in Personalized Health Systems. http://www.cloudcare2u.com.

[OSGi Alliance a] Osgi Compendium Release 7. https://osgi.org/specification/osgi.cmpn/7.0.0/service.event.html.

[Snaith et al, 2003] Snaith R. P. (2003). The Hospital Anxiety And Depression Scale. Health and quality of life outcomes, 1, 29. https://doi.org/10.1186/1477-7525-1-29.

[ISO, 2016] Iso/iec 20922:2016. https://www.iso.org/standard/69466.html

[OSGi Alliance b] https://osgi.org/specification/osgi.enterprise/7.0.0/service.jmx.html.

[Jones et al, 2015] Jones M., Bradley J., Sakimura N. (2015). Json Web Token (jwt). ISSN 2070-1721. https://tools.ietf.org/html/rfc7519.

[Keycloack Team] Open Source Identity and Access Management. https://www.keycloak.org.

[Hanke et al, 2017] Hanke, S., Kreiner, K., Kropf, J., Scase, M., & Gossy, C. (2017). Reasoning and Data Representation in a Health and Lifestyle Support System. Stud Health Technol Inform, 235, 8-12.

[Zyl & Fox, 2020] Zyl J. and Fox B. (2020). Introduction to repositories. https://maven.apache.org/guides/introduction/introduction-to-repositories.html

[Hardt, 2012] Hardt D. (Ed.) (2012). The Oauth 2.0 Authorization Framework. https://tools.ietf.org/html/rfc6749.

[Oasis, 2020] Advancing Open Standards For the Information Society. https://www.oasis-open.org.

[OSGi Alliance, 2013] The OSGi Alliance (2013). OGi Compendium Release 5, 1030 pages, https://www.osgi.org/release-5/.

[Sonatype] OSGi Bundle Repositories. https://help.sonatype.com/repomanager2/osgi-bundle-repositories.

[RabbitMQ] RabbitMQ is the most widely deployed open source message broker. https://www.rabbitmq.com.