CARE

CARE ABOUT CARE

Interface Documentation

Author(s)	Thomas Aigner, Philipp Kolmann
Organisation(s)	ilogs mobile software GmbH
	Fachhochschule Wiener Neustadt (FHWN)

Document Number	D11
Version/Date	0.12 – 24/01/2022
Document Type	Deliverable
Dissemination Level	Public
Checked and released by	Michael Szvetits

Funded by the European Commission and Partner States within the Active and Assisted Living Programme

Document History

Version	Date	Author	Comment
0.1	12/11/2021	Thomas Aigner	Structure
0.2	25/11/2021	Thomas Aigner	OAuth
0.3	29/11/2021	Thomas Aigner	Objects, Method Headers
0.4	03/12/2021	Thomas Aigner	Objects extended
0.5	09/12/2021	Philipp Kolmann	Care Expert Center, minor adjustments
0.6	14/12/2021	Thomas Aigner	Care Service updated, Service Broker added
0.7	16/12/2021	Philipp Kolmann	Matomo and Limesurvey added
0.8	13/01/2022	Thomas Aigner	Object definitions updated, methods added
0.9	14/01/2022	Philipp Kolmann	Preparation for version 1.0
0.10	21/01/2022	Michael Szvetits & Philipp Kolmann	Check for release
0.11	24/01/2022	Philipp Kolmann	Add CXC data interface
0.12	24/01/2022	Michael Szvetits & Thomas Aigner	miscellaneous corrections

Table of Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

The project Care about Care (C^C) – AAL-JP grant number AAL-2020-7-144-CP – has received funding from AAL Programme, cofounded by the European Commission, National Funding Authorities of Austria, Belgium, Luxembourg and Switzerland and the individual project partners. Collaborating partners in the C^C project are University of Applied Sciences Wiener Neustadt (lead partner), ilogs mobile software GmbH, Eichenberger-Szenografie, Vienna University of Economics and Business, Hilfswerk Niederösterreich, Korian, Stëftung Hëllef Doheem, Distrac Group. The C^C project runs from June 2021 until November 2023.

To cite this paper:

Aigner, T., Kolmann, P. (2022): Interface documentation. Project: Care about Care (C^C), Deliverable 11, ilogs mobile software GmbH & Fachhochschule Wiener Neustadt.

1 Introduction

1.1 Purpose of this document

This document represents the official deliverable D11 of the AAL project C^C – Care about Care. This document is mainly for internal use and is intended to document all relevant dataflows within the project.

1.2 Document structure

We start with general settings for the service (Section 2). Then the user authentication mechanism is discussed in detail (Section 3). The Care Service interfaces are described in Section 4. Finally, the Remote Care Assist Service interfaces are discussed in Section 6.

1.3 Definitions, acronyms and abbreviations

C^C	Care about Care
CC	Care Cockpit
CXC	Care Expert Center
HTTP and HTTPS	Hypertext transfer protocol (secure)
JSON	Javascript Object Notation
REST	Representational State Transfer
RS	Remote Support
eRS	Remote Support app (employee version)
cRS	Remote Support app (customer version)
PNG	Portable Network Graphics
SIP	Session Initiation Protocol
TLS	Transport Layer Security
UI	User Interface
URL	Uniform Resource Locator
WS	Websocket

2 General

Foremost we will define URLs, technologies and the authentication mechanism used for all web based APIs.

2.1 URLs

An overview of all C^C-services and their respective base URLs can be seen in the following table.

Service		Base URL
Authentication Service	prod	tbd.
	dev	tbd.
Service Broker Service	prod	tbd.
	dev	tbd.
Care Service	prod	tbd.
	dev	tbd.
Care Expert Center Service Web Frontend	prod	tbd.
	dev	https://dev.careaboutcare.eu/cxc/
Care Expert Center Service Web API	prod	tbd.
	dev	https://dev.careaboutcare.eu/cxc/API/
Care Expert Center Service VoIP Server	prod	tbd.
	dev	SIP/TLS: dev.careaboutcare.eu:5061 WS: https://dev.careaboutcare.eu:8089/ws

Table 1 Service Base URLs

2.2 Technology

Each web service in C^C providing a REST-based API must be encrypted using TLS (Transport Layer Security) and enforce authorization (e.g. OAuth Token).

VoIP Services in C^C use SIP over TLS or SIP over Websocket and all transports must be encrypted using TLS and enforce authorization.

The used data format for data exchange is the JSON format, in special cases (e.g. image transfer) other formats are possible. The format of the transferred or acceptable data is indicated using the "Content-Type" HTTP-Header.

The result of any call to the REST API is indicated using HTTP status codes:

HTTP-Code	Description	Possible Reasons
200	OK	Request Successful. The body of the response contains the resulting data.
400	Bad Request	Invalid request based on a caller error. For example, a required field is missing.
401	Unauthorized	Client is not authorized, i.e. has no or invalid token.

403	Forbidden	Client has a valid token, but is not allowed to perform the requested operation.
404	Not Found	The requested data has not been found.
500	Internal Server Error	The request could not be handled based on a server error. Usually, this type of error is temporary – this means, the request may be successful at a later point of time.
503	Service Unavailable	The Service is not available at the moment.

Table 2 HTTP status codes

2.3 Web API Authentification

Authentification is based on OAuth 2.0. The OAuth Provider used is based on MoBIS middleware by ilogs as described in Section 3. The OAuth Token has to be provided in each request using the Authorization-Header.

Example:

```
GET /api/v1/example HTTP/1.1
Accept: application/json
Authorization: Bearer <TOKEN>
Content-Type: application/json
Host: example.careaboutcare.eu:443
```

3 Authentication Service

The ilogs middleware component MobIS (Mobile Information System) can be used as an identity provider. SSO (Single Sign On) is supported. MobIS implements the OAuth 2.0 standard (for details see (D. Hardt, 2012)). Furthermore, MobIS does support to store key-value pairs for each authenticated user.

3.1 Base URL

See Table 1 Service Base URLs.

3.2 Input/Output Format

Depending on the method two content types are used:

- JSON: `application/json`
- URL-endcoded string: `application/x-www-form-urlencoded`

3.3 Errors

In case of an error, status code unequal 200 is responded or in the case of a redirect the query parameter “error” is set. If there is no redirect, the body contains an error message – either in JSON format (actions: /datastore and /validate) or as URL-encoded string (actions: /token). The reason why there are two different formats is, that /token is part of the OAuth 2.0 standard which demands an URL-endoded string, /datastore and /validate are not part of the standard).

Example Request

Method	GET
URL	<Base-URL>/validate

Example Response

Status-Code	401 (Unauthorized)

Response JSON	{
	"error": "access_denied",
	"error_description": "Token not valid or expired (result=NotFound)."
	}

Example Request

Method	POST
URL	<Base-URL>/token
Body	grant_type=authorization_code&code=a47df3ea-2037-44a8-8e28-0eec61a344f3&client_id=2fb543f5-9311-4363-85fd-75f26256cf9e&redirect_uri=http://localhost:49884/Secure/OAuth

Example Response

Status-Code	400 (Bad Request)
Response	error=unauthorized_client &error_description=Token+not+valid+or+expired+(result%3dExpired)

3.4 Service methods

3.4.1 Authorize

<Base-URL>/authorize?<parameters>

User will be redirected to MobIS login screen. This request will be used for the authorization methods "authorization code" (see Section 3.5.1) or "implicit" (see Section 3.5.2).

3.4.1.1 Request Parameters

Name	Data-Type	Description
response_type	string	"code" (when using method: "authorization code"). "token" (when using method "implicit")
client_id	string	Id of the client, which is generated at MobIS client registration.
redirect_uri	string	MobIS redirectes the user (browser) to this URL (callback), after authentication was successful.
state	string	Any code of the client to manage the state between request and callback and to prevent cross-site request forgery.

Example Request

Method	GET

URL	https://localhost:51956/oauth/authorize?response_type=code&client_id=9b761d45-df68-4e39-a28a-5d4796e1d16f&state=xyz&redirect_uri=https%3a%2f%2fexample.careaboutcare.eu%2fauthcallback

3.4.1.2 Callback

After authorization the client will be redirected to the given redirect URL. The result is added to the URL as parameter.

Parameters when using "authorization code" method

Name	Data-Type	Description
code	string	The authorization code, which is used to request the access token (see Section 3.5.1)
state	string	From the client provided state.

Parameters when using "implicit" method

Name	Data-Type	Description
access_token	string	The access token.
token_type	string	Type of the token – MobIS only issues token of type "Bearer".
expires_in	int	How long the access token is valid, in seconds.
state	string	From the client provided state.

Example Redirect

```
https://example.careaboutcare.eu/authcallback?code=4ca575fb-951b-464e-bd41-7c61b14e8f71&state=xyz
```

3.4.2 Token

<Base-URL>/token

This request will be used for the authorization methods "authorization code" (see Section 3.5.1) after gaining the authorization code or "resource owner password credentials" (see Section 3.5.3).

3.4.2.1 Request

All parameters are posted as URL-encoded string in the body.

Parameters when using "authorization code" method

Name	Data-Type	Description
grant_type	string	"authorization_code"
code	string	Authorization code, which was provided in the callback.
redirect_uri	string	Same redirect uri as used in /authorize method (no redirect will be made, but it is used for validity check).

client_id	string	If the client_id is not provided in the request URL, is has to be set in the body.

Parameters when using "password credentials" method

Name	Data-Type	Description
grant_type	string	"password"
username	string	resource owner username
password	string	resource owner password
client_id	string	If the client_id is not provided in the request URL, is has to be set in the body.

Example Request

Header	`POST /token HTTP/1.1`
	`Host: hostname.ilogs.com`
	`Content-Type: application/x-www-form-urlencoded`

Body	`grant_type=authorization_code&code=4ca575fb-951b-464e-bd41-7c61b14e8f71&client_id=9b761d45-df68-4e39-a28a-5d4796e1d16f&redirect_uri=https%3a%2f%2f example.careaboutcare.eu %2fauthcallback`

3.4.2.2 Response

The response (JSON) contains the access token and data-store objects (if available).

Parameters

Name	Data-Type	Description
access_token	string	Access token
token_type	string	Type of the token – MobIS only issues token of type "Bearer".
expires_in	int	How long the access token is valid, in seconds.
refresh_token	string	Refresh token, for renewing access token.
mobis_parameters	object	Contains datastore object (see Section 3.4.4)

Example Response

Status-Code	200 (OK)

Header	`HTTP/1.1 200 OK`
	`Content-Type: application/json`
	`Cache-Control: no-store`
	`Pragma: no-cache`

Body	`{`
	`  "access_token": "cf1fefe3-8729-4a23-ad6b-d63bc8c81048",`
	`  "refresh_token": "809c7d58-322e-4004-8384-995ae6c2d4b7",`
	`  "token_type": "Bearer",`
	`  "expires_in": 14400`
	`}`

3.4.3 Validate

<Base-URL>/validate

Use this method to validate the access token. The access token is provided in the authorization header. If the token is valid, status code 200 will be returned, otherwise an error code.

Example Request

Header	`GET /WebServiceMobis/oauth/validate HTTP/1.1`
	`Host: localhost`
	`Authorization: Bearer 4fde4d41-bee7-48ab-99dc-5dea4398beb8`

Example Response

Status-Code	200 (OK)

3.4.4 DataStore

<Base-URL>/datastore

In the datastore key-value pairs for each user can be stored. Datastore entries can be read, created or deleted. When reading the datastore, MobIS user id and application id is provided. When creating a key-value pair the application id is optional. If it is set, only this application can access the key-value pair, otherwise all other applications can access it (shared setting).

Each request must contain the access token in the authorization header.

3.4.4.1 Read

Example Request

Header	`GET /WebServiceMobis/oauth/datastore HTTP/1.1`
	`Host: localhost`
	`Authorization: Bearer 4fde4d41-bee7-48ab-99dc-5dea4398beb8`

Body	

Example Response

Status-Code	200 (OK)

Response JSON	`{`
	`"user_id": 1,`
	`"application_id": 20,`
	`"datastore":[`
	`{`

	<pre>"Key": "Name",
	"Value": "Karl Mustermann",
	"AppId": null
	},
	{
	"Key": "Setting1",
	"Value": "asdf",
	"AppId": 20
	}
	]
	}</pre>

3.4.4.2 Create/Update

Example Request

Header	<pre>POST /WebServiceMobis/oauth/datastore HTTP/1.1
	Host: localhost
	Authorization: Bearer 4fde4d41-bee7-48ab-99dc-5dea4398beb8</pre>

Body	<pre>[
	{
	"Key": "Name",
	"Value": "Karl Mustermann",
	"AppId": null
	},
	{
	"Key": "Setting1",
	"Value": "asdf",
	"AppId": 20
	}
	]</pre>

Example Response

Status-Code	200 (OK)
Response JSON	

3.4.4.3 Delete

Example Request

Header	<pre>DELETE /WebServiceMobis/oauth/datastore HTTP/1.1
	Host: localhost
	Authorization: Bearer 4fde4d41-bee7-48ab-99dc-5dea4398beb8</pre>

Body	<pre>[
	{
	"Key": "Name"</pre>

	<pre>},
	{
	"Key": "Setting1"
	}
	]</pre>

Example Response

Status-Code	200 (OK)
Response JSON	

3.4.5 Logout

<Base-URL>/logout

Use this method to logout the user from MobIS. If logout was successful status code 200 will be returned, otherwise an error code.

Example Request

Header	<pre>DELETE /WebServiceMobis/oauth/logout HTTP/1.1
	Host: localhost
	Authorization: Bearer 4fde4d41-bee7-48ab-99dc-5dea4398beb8</pre>

Body	

Example Response

Status-Code	200 (OK)
Response JSON	

3.5 Authorization methods

3.5.1 Authorization Code

User (browser) is redirected to MobIS where the user enters his credentials. Afterwards the user is redirected back (client application provides callback URL) to the client application. The redirect URL contains an authentication code. This code is used to request the access token. Advantage of this method: User (or other applications) can never see the access token.

User is redirected to <Base-URL>/authorize:

```
https://hostname.ilogs.com/oauth/authorize?
response_type=code&client_id=9b761d45-df68-4e39-a28a-
```

5d4796e1d16f&state=xyz&redirect_uri=https%3a%2f%2fexample.careaboutcare.eu%2fauthcallback

If the user is not already signed in (single sign on), she has to enter her credentials in the login mask (Hint: The logo can be exchanged depending on the client id).

Figure 1 example login mask

If the credentials are correct, the MobIS auth server redirects the user to the given redirect_uri. The URL contains the authorization code:

https:// example.careaboutcare.eu/authcallback?code=4ca575fb-951b-464e-bd41-7c61b14e8f71&state=xyz

With this code, the application can request the access token:

Header	POST /token HTTP/1.1 Host: hostname.ilogs.com Content-Type: application/x-www-form-urlencoded
Body	grant_type=authorization_code&code=4ca575fb-951b-464e-bd41-7c61b14e8f71&client_id=9b761d45-df68-4e39-a28a-5d4796e1d16f&redirect_uri=https%3a%2f%2fexample.careaboutcare.eu%2fauthcallback

The response contains the access token:

Header	HTTP/1.1 200 OK Content-Type: application/json Cache-Control: no-store Pragma: no-cache
Body	{ "access_token": "cf1fefe3-8729-4a23-ad6b-d63bc8c81048", "refresh_token": "809c7d58-322e-4004-8384-995ae6c2d4b7", "token_type": "Bearer", "expires_in": 14400

	}

3.5.2 Implicit

Simplified version of the authorization code method – the MobIS server directly delivers the access token in the redirect URL. Advantage: less communication overhead. Disadvantage: Access token can be read easily on client side. This method is only recommended in trustworthy environments.

User is redirected to <Base-URL>/authorize:

```
https://hostname.ilogs.com/oauth/authorize?
response_type=token&client_id=9b761d45-df68-4e39-a28a-
5d4796e1d16f&state=xyz&redirect_uri=https%3a%2f%2fexample.careaboutc
are.eu%2fauthcallback
```

If the user is not already signed in (single sign on), she has to enter her credentials in the login mask (Hint: The logo can be exchanged depending on the client id).

If the credentials are correct, the MobIS auth server redirects the user to the given redirect_uri. The URL contains the access code:

```
https://example.careaboutcare.eu/authcallback#access_token=4ca575fb-
951b-464e-bd41-
7c61b14e8f71&state=xyz&token_type=Bearer&expires_in=14400
```

3.5.3 Resource Owner Password Credentials

Username and password are posted directly in the request body. User has to enter his username and password on the client application, therefore this method should only be used on trusted applications.

Header	`POST /token HTTP/1.1`

	Host: hostname.ilogs.com Content-Type: application/x-www-form-urlencoded
Body	grant_type=password&client_id=9b761d45-df68-4e39-a28a-5d4796e1d16f&username=test&password=testpwd

The response contains the access token:

Header	HTTP/1.1 200 OK Content-Type: application/json Cache-Control: no-store Pragma: no-cache

Body	{
	"access_token": "cf1fefe3-8729-4a23-ad6b-d63bc8c81048",
	"refresh_token": "809c7d58-322e-4004-8384-995ae6c2d4b7",
	"token_type": "Bearer",
	"expires_in": 14400
	}

3.5.4 Client Credentials

Is intended for m2m-communication and has to be activated explicitly on MobIS server. For authentication only the client id is used.

Header	POST /token HTTP/1.1 Host: hostname.ilogs.com Content-Type: application/x-www-form-urlencoded
Body	grant_type=client_credentials&client_id=9b761d45-df68-4e39-a28a-5d4796e1d16f

The response contains the access token:

Header	HTTP/1.1 200 OK Content-Type: application/json Cache-Control: no-store Pragma: no-cache

Body	{
	"access_token": "cf1fefe3-8729-4a23-ad6b-d63bc8c81048",
	"refresh_token": "809c7d58-322e-4004-8384-995ae6c2d4b7",
	"token_type": "Bearer",
	"expires_in": 14400
	}

4 Care Service

4.1 Objects

4.1.1 User

Name	Data-Type	Required	Description
id	string	yes	
firstname	string	yes	
lastname	string	yes	
phoneNumber	string	no	
phoneNumber2	string	no	
emailAddress	string	no	
additionalInfo	string	no	additional information about the user (should not contain sensitve data)
detailsUrl	string	no	URL to the care organization software of this user
type	enum	yes	Type: client; employee; informalCarer (Description: -)
qualifications	Qualification[]	no	see 4.1.2
address	Address	no	see 4.1.3

4.1.2 Qualification

Name	Data-Type	Required	Description
id	string	yes	
name	string	yes	

4.1.3 Address

Name	Data-Type	Required	Description
id	string	yes	
street	string	yes	
houseNumber	string	no	
city	string	yes	
postCode	string	yes	
countryCode	string	yes	address country code in two letter ISO 3166-1
longitude	double	no	center longitude of the address (decimal degrees)
latitude	double	no	center latitude of the address (decimal degrees)

4.1.4 Task

Name	Data-Type	Required	Description
id	string	yes	

userId	string	yes	id of the user the task belongs to
requestingUserId	string	no	set this user id, of the requesting user is not the care taker - e.g. relative is requesting on behalf of the care taker
type	enum	yes	Type: cancelCareMission; postponeCareMission; absence; additionalService; linkUser; phoneCallback
creationDate	DateTime	yes	date and time when task was created
careMissionId	string	no	referencing care mission id, if task refers to a care mission (e.g., cancelCareMission, postponseCareMission)
from	DateTime	no	set when type is absence to set the start date of the absence
to	DateTime	no	set when type is absence to set the end date of the absence
description	string	no	additional information about the task
status	enum	yes	current state of the task Type: open; inProgress; completed
resolution	enum	no	information about the resolution of an completed task Type: done; notdone
resolutionComment	string	no	additional information about the resolution

4.1.5 CareMission

Name	Data-Type	Required	Description
id	string	yes	id of the care mission (can by any string - dependent on the care organization system)
userId	string	yes	id of the user the care mission belongs to
subject	string	yes	
description	string	no	
planedFrom	DateTime	yes	
plannedTo	DateTime	yes	
nameCareGiver	string	no	
status	enum	yes	Type: planned

			canceled	
			completed	

4.1.6 CareMissionDetails

Name	Data-Type	Required	Description
Same as 4.1.5			
plannedServices	Service[]	no	
actualFrom	DateTime	no	
actualTo	DateTime	no	
actualServices	Service[]	no	
isCancelable	boolean	no	true, if the care mission is cancelable by the user
isShiftable	boolean	no	true, if the care mission can be shifted by the user

4.1.7 Service

Name	Data-Type	Required	Description
id	string	yes	
name	string	yes	
description	string	no	
status	enum	no	Type: done; notdone Description:
statusDescription	string	no	textual details about the status type and maybe reason, when it was done/notdone

4.1.8 Notification

Name	Data-Type	Required	Description
id	string	yes	
userId	string	no	id of the user the notification belongs to
subject	string	yes	
description	string	no	
careMissionId	string	no	reference to the care mission
taskId	string	no	reference to the task
priority	enum	no	Type: low; medium; high Description:

4.1.9 Right

Name	Data-Type	Required	Description
id	string	yes	
name	string	yes	
description	string	yes	

4.1.10 Organization

Name	Data-Type	Required	Description
id	string	yes	
name	string	yes	

4.1.11 UserRight

Name	Data-Type	Required	Description
clientUserId	string	yes	user id of the service user
helperUserId	string	yes	user id of the informal carer
organizationId	string	yes	organization id rights belongs to
validTo	DateTime	no	end date, till the right is valid, if null, there is no limitation
rights	Right[]	yes	see 4.1.9

4.1.12 CarePlanItem

Name	Data-Type	Required	Description
id	string	yes	
userId	string	yes	
date	Date	yes	
dayTime	enum	yes	Type: morning; noon; evening Description:
Service	Service	yes	see 4.1.7

4.1.13 Invoice

Name	Data-Type	Required	Description
id	string	yes	
userId	string	yes	id of the user the invoice belongs to
date	Date	yes	
totalAmount	double	no	total amount of the invoice
totalCurrency	string	no	currency of totalAmount
fileReferenceId	string	no	reference to the file - use this id to download the invoice
additionalInfo	string	no	

4.1.14 VitalData

Name	Data-Type	Required	Description
id	string	yes	
userId	string	yes	id of the user the vital data belongs to
timestamp	Date	yes	
value	string	yes	
type	enum	yes	Type: bloodPressure Description:

			heartRate bloodGlucose weight oxygenLevel
assessment	enum	no	**Type** / **Description** ok tooHigh tooLow Assessment should be delivered from the care planning system, because interpreting medical values (good/bad ...) cannot be done in the care app (because this would require a medical product certification)

4.1.15 RegistrationResult

Name	Data-Type	Required	Description
userId	string	yes	
token	string	yes	
validThrough	DateTime	no	

4.1.16 File

Name	Data-Type	Required	Description
id	string	yes	
data	string	yes	base64 encoded string
contentType	string	yes	content type of the data (e.g., image/png, application/pdf, etc.)

4.2 Methods for Care Organization Service (implemented by Care Service)

4.2.1 Register User

Request	POST /users/register
Request Object	User
Response Status	201 (Created)
Response Object	RegistrationResult

4.2.2 Read Rights

Request	GET /users/{userId}/rights
Request Object	
Response Status	200 (OK)
Response Object	UserRights

4.2.3 Update Rights

Request	PUT /users/{userId}/rights
Request Object	UserRights
Response Status	200 (OK)
Response Object	UserRights

4.2.4 Send Notification

Request	POST /users/{userId}/notification
Request Object	Notification
Response Status	201 (Created)
Response Object	

4.3 Webhooks of Care Organization Service (implemented by Care Organization Service)

4.3.1 Read User Care Missions

Request	GET /users/{userId}/caremissions
Query Parameters	from: date to: date
Request Object	
Response Status	200 (OK)
Response Object	CareMission[]

4.3.2 Read Care Mission Detail

Request	GET /caremissions/{careMissionId}
Request Object	
Response Status	200 (OK)
Response Object	CareMissionDetails

4.3.3 Read Tasks

Request	GET /tasks
Query Parameters	userIds: list of user Ids (pipe separated)
Request Object	
Response Status	200 (OK)
Response Object	Task[]

4.3.4 Create Task

Request	POST /tasks

Request Object	Task
Response Status	201 (Created)
Response Object	Task

4.3.5 Read Contact Persons

Request	GET /users/{userId}/contactpersons
Request Object	
Response Status	200 (OK)
Response Object	User[]

4.3.6 Read User

Request	GET /users/{userId}
Request Object	
Response Status	200 (OK)
Response Object	User

4.3.7 Read User Picture

Request	POST /users/{userId}/picture
Request Object	
Response Status	200 (OK)
Response Object	File

4.3.8 Read Care Plan

Request	POST /users/{userId}/careplan
Query Parameters	from: date to: date
Request Object	
Response Status	200 (OK)
Response Object	CarePlanItem[]

4.3.9 Read Invoice

Request	POST /users/{userId}/invoices
Query Parameters	from: date to: date
Request Object	
Response Status	200 (OK)
Response Object	Invoice[]

4.3.10 Read File

Request	POST /users/{userId}/file/{fileId}

Query Parameters	
Request Object	
Response Status	200 (OK)
Response Object	File

4.4 Data Flows

4.4.1 Authentication

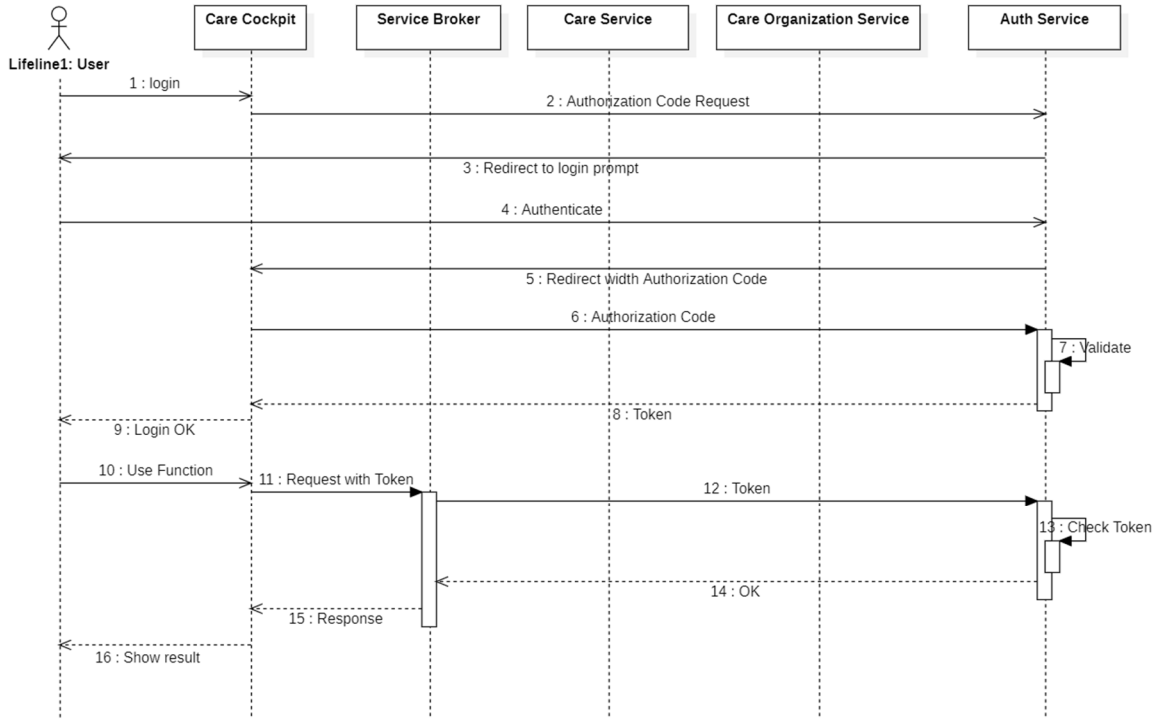

Figure 2 authentication data flow

4.4.2 Care Service Registration

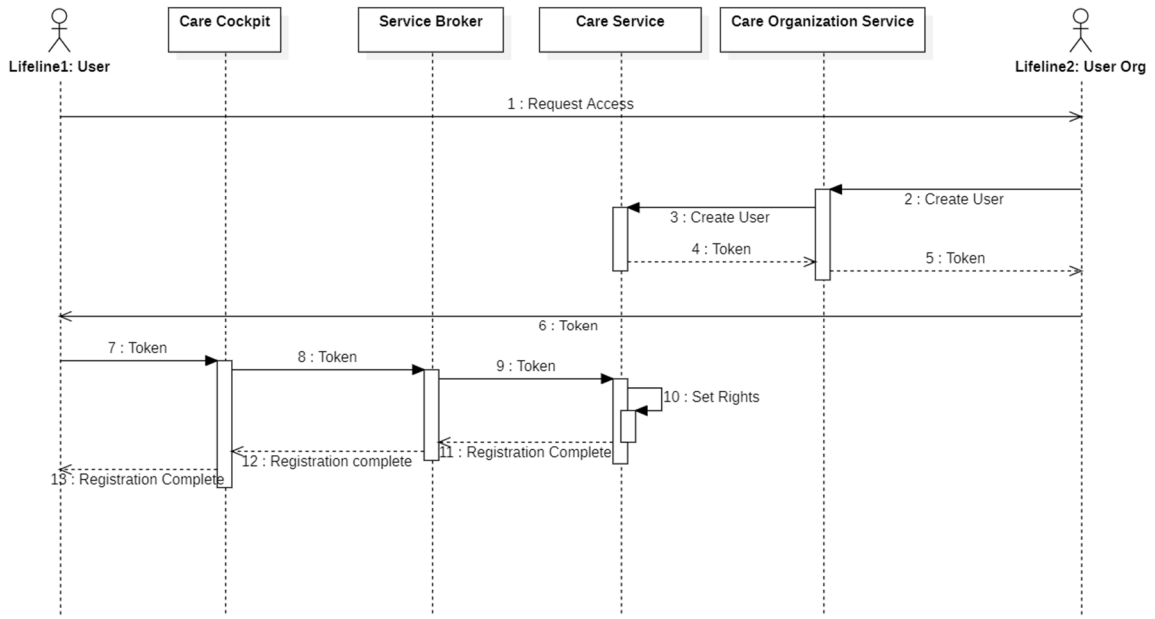

Figure 3 care service registration data flow

4.4.3 Cancel Care Mission

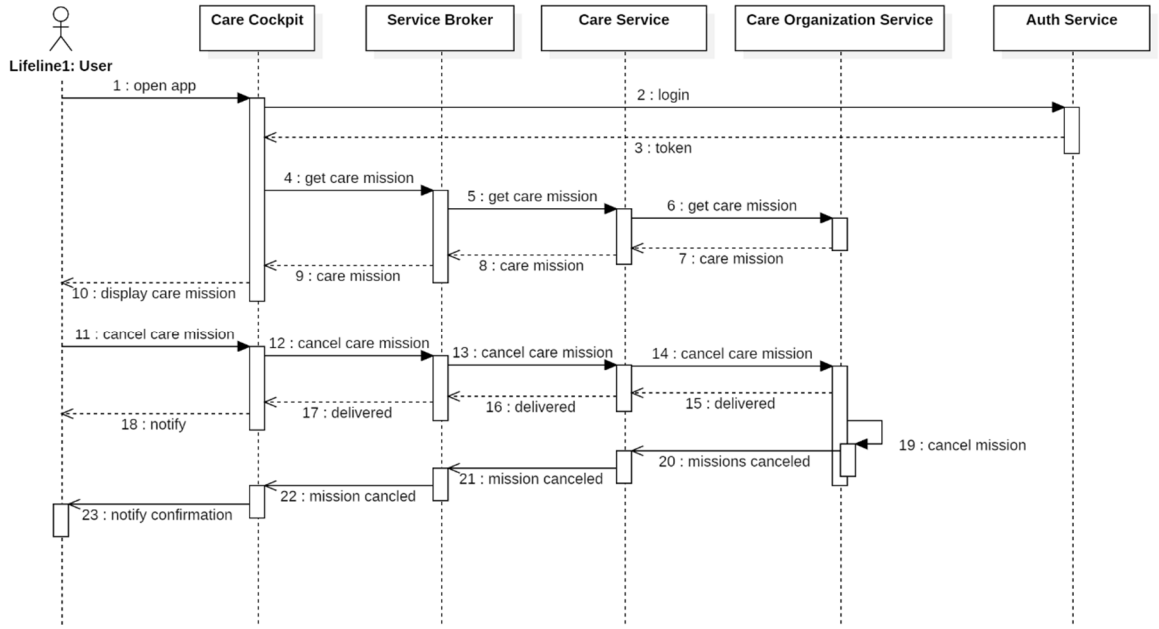

Figure 4 cancel care mission data flow

4.4.4 Appointments

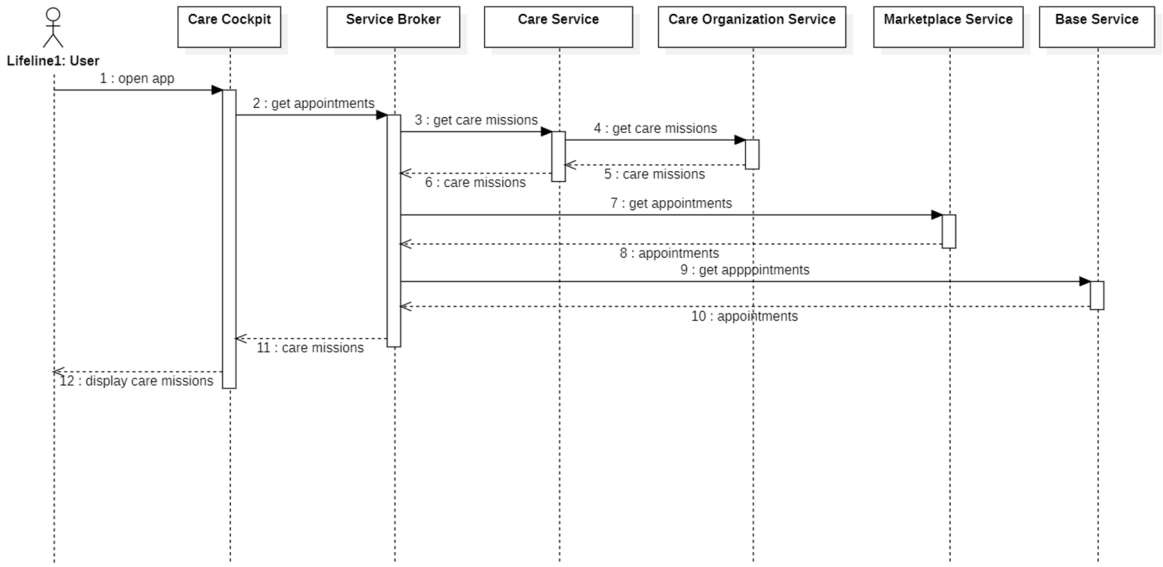

Figure 5 appointments data flow

5 Service Broker

5.1 Objects

5.1.1 User

Name	Data-Type	Required	Description
id	string	yes	
firstname	string	no	
lastname	string	yes	
username	string	yes	
phoneNumber	string	no	
phoneNumber2	string	no	
emailAddress	string	no	
photoSet	boolean	yes	true, if a photo is available
language	string	yes	IETF language tag (e.g., de-AT)
timeZone	string	yes	IANA timezone format (e.g., Europe/Vienna)
role	string	yes	User role name

5.1.2 UserCareDetails

Name	Data-Type	Required	Description
detailsUrl	string	no	link to users details in the care organization system

phoneNumber	string	no	phone number of the user (stored at the care organization system)
additionalInformation	string	no	additional free text information (should not contain sensitve data)
type	enum	no	Type: client; employee; informalCarer (Description: –)
qualifications	Qualification[]	no	see 5.1.4
homeBase	string	no	information of the working place

5.1.3 UserDetails

Name	Data-Type	Required	Description
Same as 5.1.1			
careDetails	userCareDetails	no	

5.1.4 Qualification

Name	Data-Type	Required	Description
id	string	yes	
name	string	yes	

5.1.5 File

Name	Data-Type	Required	Description
id	string	yes	
data	string	yes	base64 encoded string
contentType	string	yes	content type of the data (e.g., image/png, application/pdf, etc.)

5.2 Methods for Care Expert Center

5.2.1 Read Personal Information

Request	GET /users/me
Query Parameters	
Request Object	
Response Status	200 (OK)
Response Object	User

5.2.2 Read Care Experts

Request	GET /users/careexperts
Query Parameters	
Request Object	

Response Status	200 (OK)
Response Object	UserDetails[]

5.2.3 Read User Photo

Request	GET /users/{userId}/photo
Query Parameters	
Request Object	
Response Status	200 (OK)
Response Object	File

5.2.4 Read User Details

Request	GET /users/{userId}/caredetails
Query Parameters	
Request Object	
Response Status	200 (OK)
Response Object	UserDetails

6 Remote Care Assist Service

The Remote Care Assist Service provides backend APIs for the Care Expert Center and the Remote Support apps.

6.1 Objects

6.1.1 Person

The Person API can be called for the currently logged in User to fetch the own details (including SIP session information) or gather details about calling persons (via extension query parameter).

Name	Data-Type	Required	Description
firstName	string	yes	
lastName	string	yes	
ext	string	yes	Extension
contactNumber	string	no	
homeBase	string	no	
additionalInfo	string	no	
photo	string	no	
link	string	no	
sipUser	string	no	Only for own user.
sipPass	string	no	Only for own user.
sipHost	string	no	Only for own user.
sipWebsocketPort	Integer	no	Only for own user.
sipWebsocketPath	string	no	Only for own user.

6.1.2 Experts

The Experts API can be called to get a list of all available experts. Each expert is defined with the following parameters

Name	Data-Type	Required	Description
firstName	string	yes	
lastName	string	yes	
ext	string	yes	Extension
contactNumber	string	no	
homeBase	string	no	
additionalInfo	string	no	
photo	string	no	
link	string	no	

6.1.3 Data

The Data API is used to store screenshots on the server to transfer them from the CXC to the RS device.

A new screenshot can be uploaded to the server via the HTTP PUT method and the base64 encoded PNG image as message body. The API will return a JSON encoded fileId which can be transferred to the RS device via SIP MESSAGE.

The RS device can use the HTTP GET method with the fileId to access the file and can delete the file with the HTTP DELETE method.

6.2 VoIP

The VoIP service used for the CXC will be provided by a self-hosted Asterisk Server. The Service will be accessible via Websocket Port for Browser-Based SIP calls and via SIP over TLS for other clients (HoloLens, Smartphone).

6.2.1 SIP Protocol

The CXC will use standard SIP protocol for signalling. Details regarding the SIP protocol can be found in RFC 3515 (Sparks, 2003).

6.3 Data flows

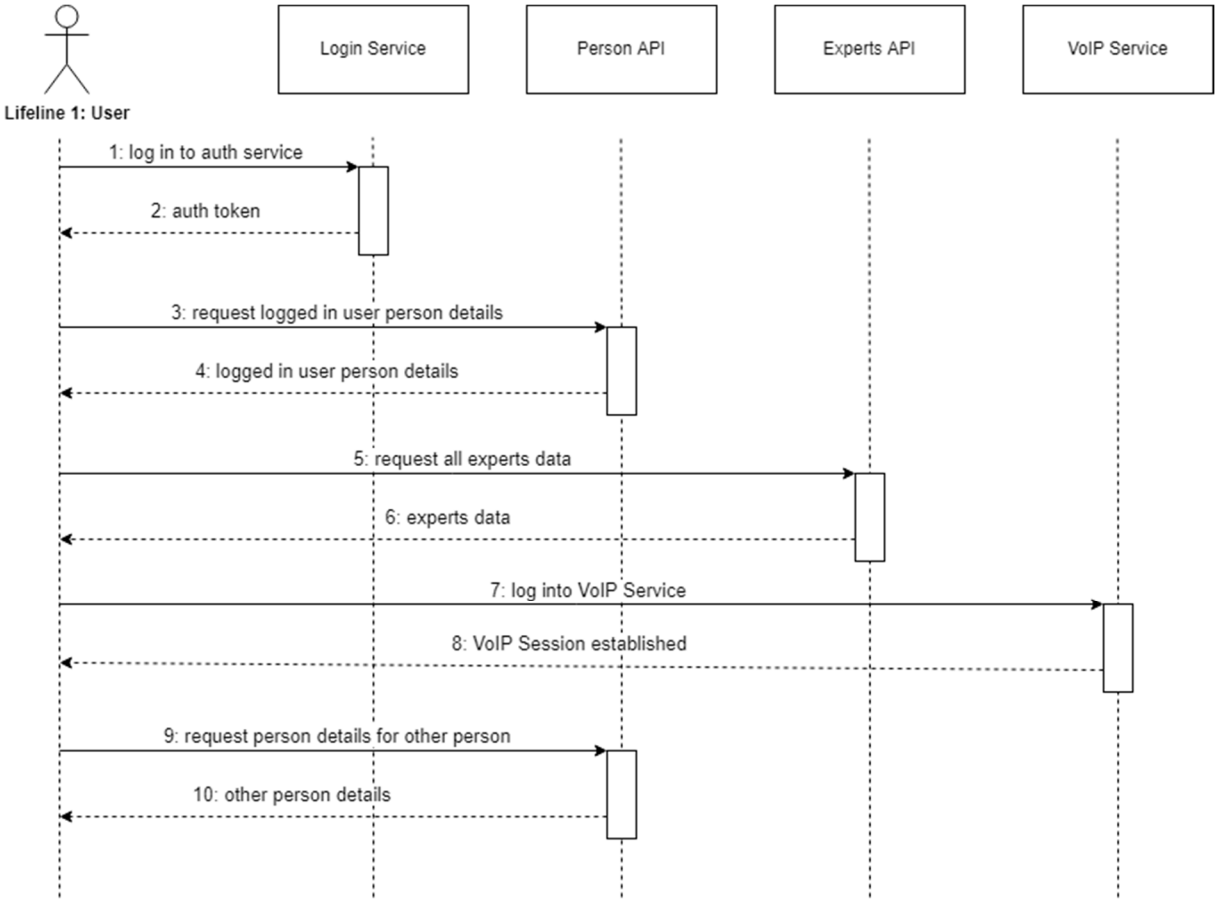

Figure 6 remote support data flow

7 Usage recording

To trace users clicks in the different applications FHWN will deploy and maintain a central Matomo (https://matomo.org/) instance where all applications will log the user interaction to.

7.1 Logging

The Matomo instance is available at https://matomo.careaboutcare.eu/ and each component will receive their own logging instance (site). Logging of user actions can either be performed by using provided SDKs (https://matomo.org/integrate/#programming-language-platforms-and-frameworks) or sending plain HTTP-requests (https://developer.matomo.org/api-reference/tracking-api).

8 After usage online survey

Additionally to the user interaction logging an after usage online survey will be provided by a Limesurvey (https://www.limesurvey.org/) instance hosted by WU Wien. Users will be asked to fill out the online survey after having used a function of the C^C project to document their experience.

9 References

D. Hardt, E. (2012, October). *IETF*. Retrieved from The OAuth 2.0 Authorization Framework: https://datatracker.ietf.org/doc/html/rfc6749

Sparks, R. (2003, April). *IETF*. Retrieved from The Session Initiation Protocol (SIP) Refer Method: https://datatracker.ietf.org/doc/html/rfc3515