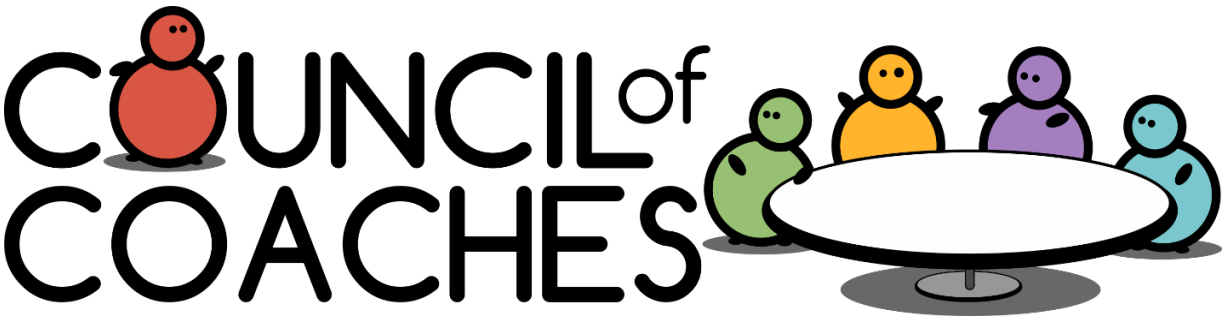


D3.5: Shared Knowledge Base component

Dissemination level: Public
Document type: Demonstrator
Version: 1.0.0
Date: November 27, 2019

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement #769553. This result only reflects the author's view and the EU is not responsible for any use that may be made of the information it contains.

Document Details

Project Number	769553
Project title	Council of Coaches
Title of deliverable	Shared Knowledge Base component
Due date of deliverable	November 30th, 2019
Work package	WP3
Author(s)	Harm op den Akker (RRD), Tessa Beinema (RRD), Dennis Hofs (RRD), Boris van Schooten (RRD)
Reviewer(s)	Alvaro Fides Valero (UPV), Jorien van Loon (CMC)
Approved by	Coordinator
Dissemination level	PU: Public
Document type	Demonstrator
Total number of pages	16

Partners

- University of Twente – Centre for Monitoring and Coaching (CMC)
- Roessingh Research and Development (RRD)
- Danish Board of Technology Foundation (DBT)
- Sorbonne University (SU)
- University of Dundee (UDun)
- Universitat Politècnica de València, Grupo SABIEN (UPV)
- Innovation Sprint (iSPRINT)

Abstract

This document accompanies the software demonstrator release of the Council of Coaches "Shared Knowledge Base" (SKB) component. The SKB serves as a central storage point of information in both the Functional- and Technical project demonstrators. The SKB stores static content (dialogues) and user profile information in such a way that it can be easily injected into those dialogues.

Table of Contents

List of figures

Symbols, abbreviations and acronyms

API	Application Programmer's Interface
CMC	Centre for Monitoring and Coaching
COUCH	Council of Coaches
D	Deliverable
DBT	Danish Board of Technology Foundation
EC	European Commission
HBAF	Holistic Behaviour Analysis Framework
ISPRINT	Innovation Sprint
M	Month
MS	Milestone
R2D2	Roessingh Research and Development Database
RRD	Roessingh Research and Development
SKB	Shared Knowledge Base
SU	Sorbonne University
UDun	University of Dundee
UPV	Universitat Politècnica de València
UT	University of Twente
WP	Work Package

1 Introduction

This document accompanies the software release of the Shared Knowledge Base component as of November 2019 (M27 out of 36) in the Council of Coaches project. The Shared Knowledge Base is the primary deliverable of Task 3.3: Development of Shared Knowledge Base, running between M6 and M27 as part of Work Package 3: "Coaching Strategies and Knowledge Base". The description for Task 3.3 is repeated here below:

> **Task 3.3: Development of Shared Knowledge Base (M6-M27)**
> All relevant identified parameters from T3.1 as well as the input from the user requirements gathering process in T2.3 will be translated into a model of a shared knowledge base. This knowledge base will contain all the information regarding the users (User Model), their previous interactions with the council of coaches or individual coaches (Interaction Model), relevant identified contextual information – e.g. weather data to influence decisions on providing appropriate advise (op den Akker, Jones, & Hermens, 2010) – (Context Model) and relevant health specific parameters (Domain Models) for Diabetes Type 2 and Chronic Pain. Both user- and context models are partly static (e.g. gender, language) and partly dynamic (e.g. self-efficacy, stage of change), dependent on interaction topic, issues under discussion and context. For the technical implementation of the shared knowledge base the project will explore the possibility of extending the FIWARE Orion Context Broker. The Shared Knowledge Base component will be updated coinciding with the three functional prototype releases of T7.2 in M9, M15 and M21, and a final version will be delivered with the technical prototype in M27.

The Shared Knowledge Base is used as the central component for data storage for the two main software demonstrators of Council of Coaches: the Functional Demonstrator and the Technical Demonstrator. Although the general architecture and functionality as described in this document will remain unchanged, additional content will still be added to the knowledge base, leading up to the Functional Demonstrator version that will be used in the Final Evaluation (February 2020) and beyond, as additional content or changes are required during the evaluation. In the context of the Technical Demonstrator, technical development will continue leading up to the release of the Open Agent Platform and accompanying Council of Coaches use case demonstrators up to the end of the project (August 2020) and beyond.

The Shared Knowledge Base software is an embedded part of the server-side Functional Demonstrator prototype (as can be seen live at https://www.council-of-coaches.eu/beta/), and is used within the Technical Demonstrator as well (see D7.5 for additional details). As such, the Shared Knowledge Base does not have a visual GUI or other demonstrable entity that can be shown in and of itself.

2 Objectives

This document has two objectives that are covered in the following two sections of this deliverable. First, in Section 3, a quick overview of the SKB component is provided, explaining its core features, internal architecture and its place within the Functional and Technical demonstrators. Then, in Section 4, we provide technical documentation of the SKB APIs, showing how component developers within the Council of Coaches ecosystem can make use of the SKB service for storing information or using the information provided therein.

In short, the objectives of D3.5 are to:

- Provide an overview of the Shared Knowledge Base components functionality (Section 3).
- Provide API Documentation of the Shared Knowledge Base component (Section 4).

3 Shared Knowledge Base component overview

In Council of Coaches, the "Shared Knowledge Base" component is the central point of data collection. The Shared Knowledge Base collects information about the user, his context and history of interactions with the coaches in the council. As a software implementation, the SKB is not a single software module. Instead, the SKB consists of two main parts, a flat storage of variables (the **Variable Store**) that can be used directly in dialogues through the WOOL Dialogue Framework (see D3.4 for more details), and a more complex data storage solution for sensor data and historical data (**R2D2** – or Roessingh Research and Development Database).

Figure 1 below shows the high-level architecture overview, the components in which are shortly described here:

- **Client:** the user interface through which end-users interact with the system, this can be either the web-based "Functional Demonstrator", or the 3D Unity-based "Technical Demonstrator".
- **COUCH Server:** the server-side software component that directly controls access to the data stored in Council of Coaches, and runs the processes for managing dialogues within the Functional Demonstrator. This server-side component contains:
    - **WOOL Dialogue Framework**: the Java software library that can read/parse .wool scripts (containing the dialogue logic), execute dialogues step-by-step, and inject variables into dialogue steps.
    - **Variable Store Synchronizer:** a process for updating Variable Store values from external sources (through R2D2).
    - **Variable Store:** a flat storage component for a list of variables that can be used in (or set from) .wool dialogues.
    - **COUCH API:** the end-points used to control the interaction between Client and Server in the Functional Demonstrator and that controls access to and from the Variable Store.
- **R2D2:** the Roessingh Research and Development Database (R2D2) is a database management system, consisting of an API, persistent storage mechanisms through SQL or No-SQL databases and additional data processing components. Relevant components include:
    - **Fitbit Synchronizer:** a process for managing the connection between R2D2/COUCH user accounts and Fitbit accounts. Periodically retrieves Fitbit data through the Fitbit API and stores this internally, providing access through the API.
    - **Goal-Setting Module:** a process that periodically sets a personalized step goal for each user based on historical physical activity data.
    - **R2D2 API:** the general set of end-points for all data storage and retrieval through the R2D2 system.
    - **HBAF API:** a specific set of end-points for allowing the HBAF to authenticate and have access to COUCH specific data (Fitbit).
- **HBAF:** the Holistic Behaviour Analysis Framework, a COUCH component that performs data analysis and sends back derived data (see WP4 deliverables for further details).

The two main Shared Knowledge Base components (**Variable Store** and **R2D2**), how they interact with each other and with other Council of Coaches system components will be shortly explained below. In Figure 1 below, a high-level schematic overview is given in which these two components (**Variable Store** and **R2D2**) are placed into context.

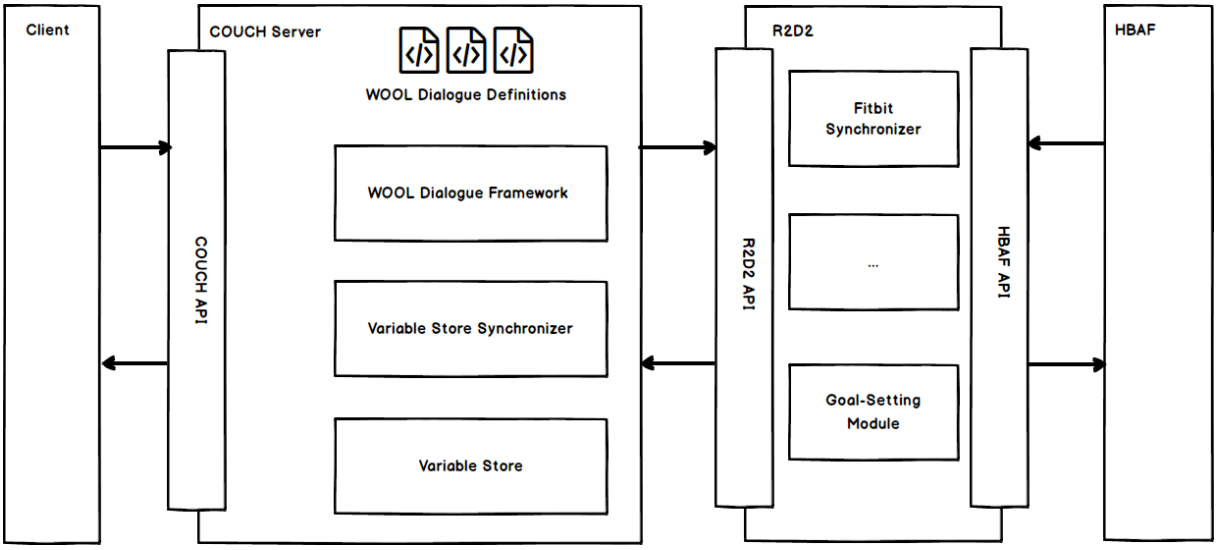
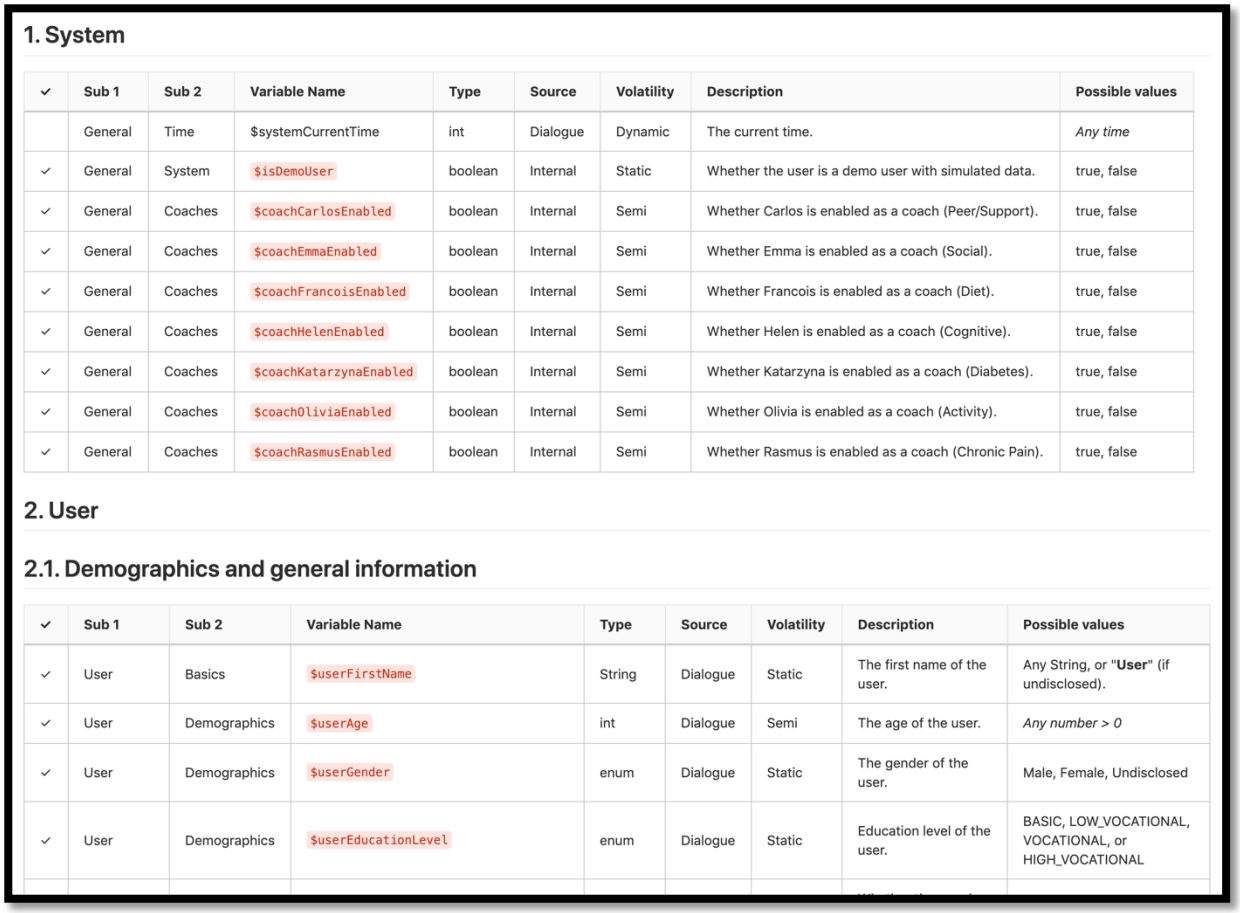


Figure 1: High-level schematic overview of the COUCH Server and R2D2, the two main server-side components that house the data storage in Council of Coaches (Shared Knowledge Base).

3.1 Variable Store

The Variable Store is an extremely simplistic data storage module that stores for each user in the Council of Coaches a flat list of variables that somehow relate to that specific user. The Variable Store is represented in memory as a simple List of Key-Value pairs, stored in persistent storage through R2D2 in a JSON format in a No-SQL database. In Figure 2 a screenshot is given of the internal Variable Store documentation used by the developers in the project. In this example you can see a set of variables that relate to which coaches are enabled for this user (`$coachCarlosEnabled`, `$coachEmmaEnabled`, …) and the first set of demographic variables (`$userFirstName`, `$userAge`, `$userGender`, `$userEducationLevel`). The reason for this simple, flat data representation has to do with the WOOL Dialogue Framework that is used to drive the dialogues between user and agents, especially within the project's Functional Demonstrator.



1. System

✓	Sub 1	Sub 2	Variable Name	Type	Source	Volatility	Description	Possible values
	General	Time	$systemCurrentTime	int	Dialogue	Dynamic	The current time.	*Any time*
✓	General	System	`$isDemoUser`	boolean	Internal	Static	Whether the user is a demo user with simulated data.	true, false
✓	General	Coaches	`$coachCarlosEnabled`	boolean	Internal	Semi	Whether Carlos is enabled as a coach (Peer/Support).	true, false
✓	General	Coaches	`$coachEmmaEnabled`	boolean	Internal	Semi	Whether Emma is enabled as a coach (Social).	true, false
✓	General	Coaches	`$coachFrancoisEnabled`	boolean	Internal	Semi	Whether Francois is enabled as a coach (Diet).	true, false
✓	General	Coaches	`$coachHelenEnabled`	boolean	Internal	Semi	Whether Helen is enabled as a coach (Cognitive).	true, false
✓	General	Coaches	`$coachKatarzynaEnabled`	boolean	Internal	Semi	Whether Katarzyna is enabled as a coach (Diabetes).	true, false
✓	General	Coaches	`$coachOliviaEnabled`	boolean	Internal	Semi	Whether Olivia is enabled as a coach (Activity).	true, false
✓	General	Coaches	`$coachRasmusEnabled`	boolean	Internal	Semi	Whether Rasmus is enabled as a coach (Chronic Pain).	true, false

2. User

2.1. Demographics and general information

✓	Sub 1	Sub 2	Variable Name	Type	Source	Volatility	Description	Possible values
✓	User	Basics	`$userFirstName`	String	Dialogue	Static	The first name of the user.	Any String, or "**User**" (if undisclosed).
✓	User	Demographics	`$userAge`	int	Dialogue	Semi	The age of the user.	*Any number > 0*
✓	User	Demographics	`$userGender`	enum	Dialogue	Static	The gender of the user.	Male, Female, Undisclosed
✓	User	Demographics	`$userEducationLevel`	enum	Dialogue	Static	Education level of the user.	BASIC, LOW_VOCATIONAL, VOCATIONAL, or HIGH_VOCATIONAL

Figure 2: Screenshot of the internal Variable Store documentation (GitLab Wiki, currently not publicly available).

3.2 The WOOL Dialogue Framework

WOOL consists of a dialogue specification language, a visual editor meant for non-technical dialogue authors, and a set of tools to parse and execute the dialogues in client-side or server-side settings. The WOOL Dialogue Framework is described in more detail in Deliverable 3.4. In Example 1 and Example 2 below, two examples are given of typical WOOL dialogue nodes.

```
I can tell you about each of the options, or let me know if you're ready
to decide.

[[Steps taken?|SuggestStepsTaken]]
[[Active minutes?|SuggestActiveMinutes]]
[[Calories burned?|SuggestCaloriesBurned]]
[[I am ready to decide.|DecidePreferredUnit]]
[[Goodbye.|End]]
```

Example 1: Example of simple WOOL Dialogue Node.

The first example above shows how simple dialogue steps with reply options can be encoded.

```
<<set $testMode = true>>

<<if $testMode>>
```

```
  // Set a bunch of variables for testing purposes.
  <<set $paUserTrackerConnected = true>>
  <<set $paUserFinalGoalSet = false>>
  <<set $paUserDailyGoalSet = false>>

  Okay, let's talk about goal setting...

  <<if $paUserTrackerConnected>>

    <<if $paUserFinalGoalSet == false>>
      [[SetFinalGoal]]
    <<else>>
      <<if $paUserDailyGoalSet == false>>
        [[SetDailyGoal]]
      <<else>>
        [[ReviseGoals]]
      <<endif>>
    <<endif>>

  <<else>>
    [[TrackerNotConnected]]
  <<endif>>

<<else>>
  This part of my dialogue is still being written.

  [[Okay, take me back.|olivia-coaching-menu.Start]]
  [[Goodbye.|End]]
<<endif>>
```

Example 2: Example of a more complicated Node in a WOOL Dialogue in which the flow of the dialogue is controlled through various variables from the Variable Store.

The second example above here, shows a more complicated WOOL Node in which various variables are being set and used to personalize the interaction between user and coach by controlling the flow of the dialogue. In order to minimize the complexity of these typical *control flow* nodes in WOOL, the use of variables in if-statements is kept as simple as possible – simply insert the variable you want to use in an if- or set-statement, without being bothered where its value comes from.

This is the pragmatic reasoning behind the flat **Variable Store** that forms a central component within the Knowledge Base of Council of Coaches.

3.3 Variable Store Usage Sequence

In order to further explain the connection between the various components, we zoom in on part of the high-level architecture in Figure 3 below.

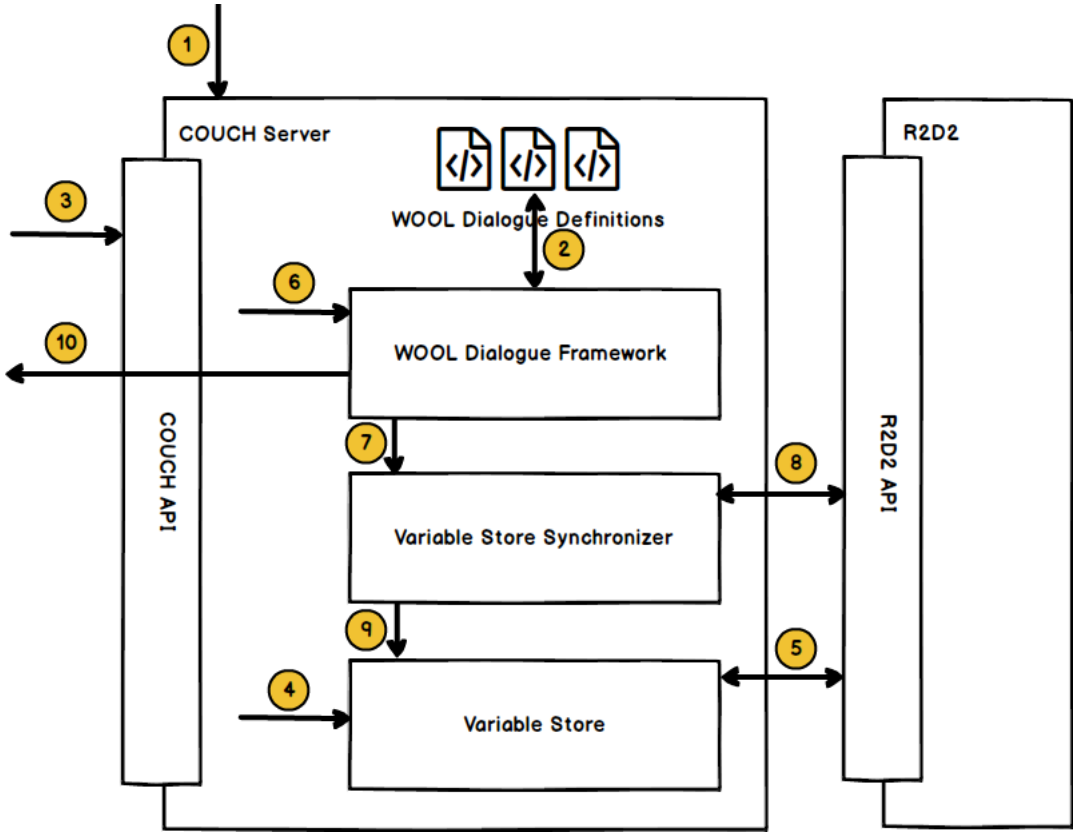


Figure 3: Zoom in of the SKB high-level architecture, demonstrating the process of setting and using variables within WOOL dialogues in the context of the Council of Coaches Functional Demonstrator.

The numbers in Figure 3 are steps in the process of launching the Council of Coaches server component (which happens only once), and starting a dialogue interaction from a Client by a specific User (which happens frequently). Each step is explained in detail below:

- **Step 1:** The Council of Coaches server component (*COUCH Server*) is started.
- **Step 2:** The *WOOL Dialogue Framework* loads in all WOOL Dialogue resources (`.wool` scripts) from file, parsing the scripts and storing the dialogue definitions as WOOL Dialogue Models.
- **Step 3:** A client-side action triggers the need to start a specific WOOL Dialogue (for example, a user taps on François) entering the system through the *COUCH API*, upon which the COUCH Server decides to start e.g. the "`francois-start.wool`" dialogue. This "decision" to start a certain dialogue is either pre-defined or handled by the topic-selection algorithm (discussed in D3.4).
- **Step 4:** If the user who initiated the interaction has not yet been "loaded", the system performs the necessary start-up actions, including loading in the **Variable Store** for that user.
- **Step 5:** The Variable Store contents for the user are requested from the R2D2 component, through a call to the *R2D2 API* (getVariables, see Section 4). R2D2 takes care of maintaining persistent storage of user's variable store and other data.
- **Step 6:** The WOOL Dialogue Framework receives the request to start the specified dialogue (e.g. `francois-start.wool`). The Dialogue Model is loaded and converted into an "Active Wool Dialogue". At this point, the WOOL Dialogue Framework checks which variables are used within this specific dialogue.
- **Step 7:** For all variables used in the dialogue, the **Variable Store Synchronizer** checks whether or not the value of the variable needs to be updated (in the case of *Dynamic Variables* such as step data, these values need to be as up-to-date as possible).
- **Step 8:** For all Dynamic Variables, the latest value is retrieved through the R2D2 API (getVariable, see Section 4).
- **Step 9:** Variables with updated values are stored in the user's **Variable Store**.

- **Step 10:** With all user data up-to-date, the WOOL Dialogue Framework starts the dialogue, parsing the first step of the dialogue and its reply options, and sends this as a response to the initial request through the COUCH API (see Step 3).

This example shows the core loop of using data within dialogue-based interactions within the Council of Coaches system. Additional components listed in Figure 1 are shortly explained below:

- **Fitbit Synchronizer:** a component within the R2D2 (Roessingh Research and Development Database) system that manages the data connection between Council of Coaches users and their Fitbit accounts. The Fitbit synchronizer is used in establishing the connection initially and for periodically retrieving the latest Fitbit data from the Fitbit server to store within the R2D2 database.
- **Goal-Setting Module:** a component within R2D2 that automatically calculates a personalized, reasonable step goal for Council of Coaches users. The goal calculating is based on user's own previous data and runs periodically to update goal values to slowly move towards a guideline-based, and configurable ultimate value. This goal-setting data is converted into a dynamic Variable Store variable.
- **HBAF:** The Holistic Behaviour Analysis Framework is described in more details in the Deliverables related to Work Package 4 of the project. The HBAF is able to retrieve Fitbit (and other) data through the R2D2 API for all Council of Coaches users (authenticating with a public-private key pair). The HBAF can perform additional data analysis and send its results back to be converted to Variable Store variables.

4 Shared Knowledge Base component APIs

The high-level architecture overview in Figure 1 shows a number of different APIs. Without including the full API descriptions (which are available through Swagger for the developers internally to the project), this section shows some examples of the key API functions mentioned in the descriptions of Section 3.

COUCH API

The COUCH API is used for all interactions between the COUCH web client (Functional Demonstrator, see D7.5) and the COUCH Server. As part of this API, there are three API end-points for controlling the Variable Store values externally, that are used in both Functional- and Technical Demonstrator. These three end-points are shown in the Swagger documentation screenshot below (Figure 4). All three end-points accept or deliver simple JSON strings that contain the values of the variables requested or provided.

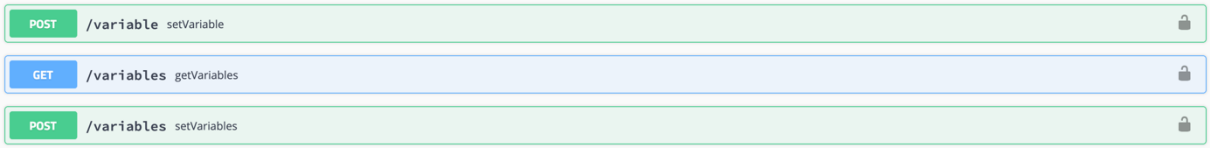


Figure 4: Screenshot of the Swagger documentation for the three Variable Store related end-points of the COUCH API.

HBAF API

An extension to the existing R2D2 API to control requesting and pushing of data from and to R2D2, specifically for the Holistic Behaviour Analysis Framework. The first end-point is a test end-point to check whether authentication (handled through the main R2D2 API) has been set up successfully. The other two end-points are for retrieving (GET) and sending (POST) data from- and to the R2D2 Server.

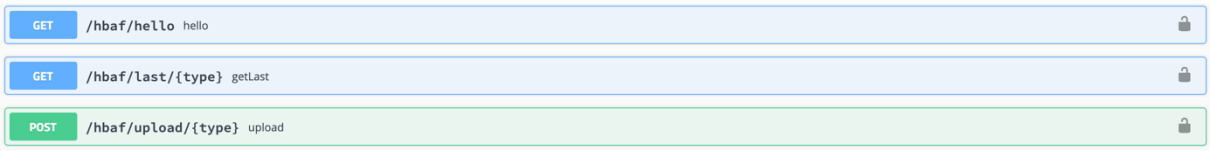


Figure 5: Screenshot of the Swagger documentation for the three end-points added to the R2D2 API to serve the HBAF needs for requesting and pushing data to the Shared Knowledge Base (through R2D2).

5 Bibliography

op den Akker, H., Jones, V. M., & Hermens, H. J. (2010). Predicting Feedback Compliance in a Teletreatment Application. *Proceedings of ISABEL 2010: the 3rd International Symposium on Applied Sciences in Biomedical and Communication Technologies*. Rome, Italy.

Acknowledgements

The Council of Coaches project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement #769553. This result only reflects the author's view and the EU is not responsible for any use that may be made of the information it contains.

Headings and titles in this document, as well as the Council of Coaches logo use the Comfortaa font, designed by Johan Aakerlund and Cyreal and licensed under the Open Font License[1].

Additional text in this document uses the Roboto font, designed by Christian Robertson and licensed under the Apache License, Version 2.0[2].

The Council of Coaches logo and Blobmen graphics were *drawn freely* in Inkscape, licensed under the GNU General Public License[3].

[1] Open Font License: http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL_web
[2] Apache License, Version 2.0: http://www.apache.org/licenses/LICENSE-2.0
[3] Inkscape License Information: https://inkscape.org/about/license/