# POWER2DM

**"Predictive model-based decision support for diabetes patient empowerment"**

**Research and Innovation Project**
**PHC 28 – 2015: Self-management of health and disease and decision support systems based on predictive computer modelling used by the patient him or herself**

## POWER2DM Deliverable 3.5

## D3.3.1a - Action Plan Engine I

| | |
|---|---|
| *Due Date:* | 30th April 2017 (M15) |
| *Actual Submission Date:* | 28th April 2017 |
| *Project Dates:* | Project Start Date: February 01, 2016<br>Project End Date: July 31, 2019<br>Project Duration: 42 months |
| *Deliverable Leader:* | SRFG |

| **Project co-funded by the European Commission within H2020 Programme (20015-2016)** | | |
|---|---|---|
| **Dissemination Level** | | |
| **PU** | Public | X |
| **PP** | Restricted to other programme participants (including the Commission Services) | |
| **RE** | Restricted to a group specified by the consortium (including the Commission Services) | |
| **CO** | Confidential, only for members of the consortium (including the Commission Services) | |

**Document History:**

| Version | Date | Changes | From | Review |
|---|---|---|---|---|
| V0.1 | 2017-04-18 | Implementation report template and content structure | SRFG | |
| V0.2 | 2017-04-21 | Initial version of chapters | SRFG | |
| V0.3 | 2017-04-27 | Integrated Final Draft | SRFG | SRDC, SRFG |
| V1.0 | 2017-04-28 | Final Version | SRFG | |

**Contributors (Benef.)** Felix Strohmeier (SRFG)
Dietmar Glachs (SRFG)
Oliver Jung (SRFG)
Robert Mulrenin (SRFG)

**Responsible Author** Felix Strohmeier (SRFG) **Email** felix.strohmeier@salzburgresearch.at

# POWER2DM Consortium Partners

| Abbv | Participant Organization Name | Country |
|---|---|---|
| TNO | Nederlandse Organisatie voor Toegepast Natuurwetenschappelijk Onderzoek | Netherlands |
| IDK | Institute of Diabetes “Gerhardt Katsch” Karlsburg | Germany |
| SRDC | SRDC Yazilim Arastirma ve Gelistirme ve Danismanlik Ticaret Limited Sirketi | Turkey |
| LUMC | Leiden University Medical Center | Netherlands |
| SAS | SAS Servicio Andaluz de Salud | Spain |
| SRFG | Salzburg Research Forschungs Gesellschaft | Austria |
| PD | PrimeData | Netherlands |
| iHealth | iHealth EU | France |

# TABLE OF CONTENTS

# 1 Introduction

## 1.1 Purpose and Scope

In the scope of Task 3.3 the Action Plan Engine has been developed. The purpose of deliverable D3.5 is to provide the software implementation and demonstrator for POWER2DM Action Plan Engine. This document provides an implementation report illustrating its architecture, technological background and gives an overview of the provided functionality for Prototype 1. Finally, the interventions for diabetes patients are explained in more detail.

## 1.2 References

- D1.2 Requirement Analysis of POWER2DM
- D1.3 Conceptual Design of POWER2DM
- D3.1 Dynamic Health Behaviour Change Intervention Models for Self-management
- D3.3 Recommender Engine (Communication Engine)
- D3.7 Mock-ups for Web and Mobile User Interfaces for SMSS Interventions
- D4.1 Personal Data Model and Service API
- D4.2 Personal Data Store Service Implementation
- D4.9 Privacy/Security Enablers for POWER2DM Services - I

# 2 Approach and Architecture

## 2.1 Action Plan Engine Overview

The Action Plan Engine is designed as a Self-Management-Support-System (SMSS) fostering behavior changes for diabetic patients. As shown in Figure 1, the Action Plan Engine is developed as part of the POWER2DM SMSS.

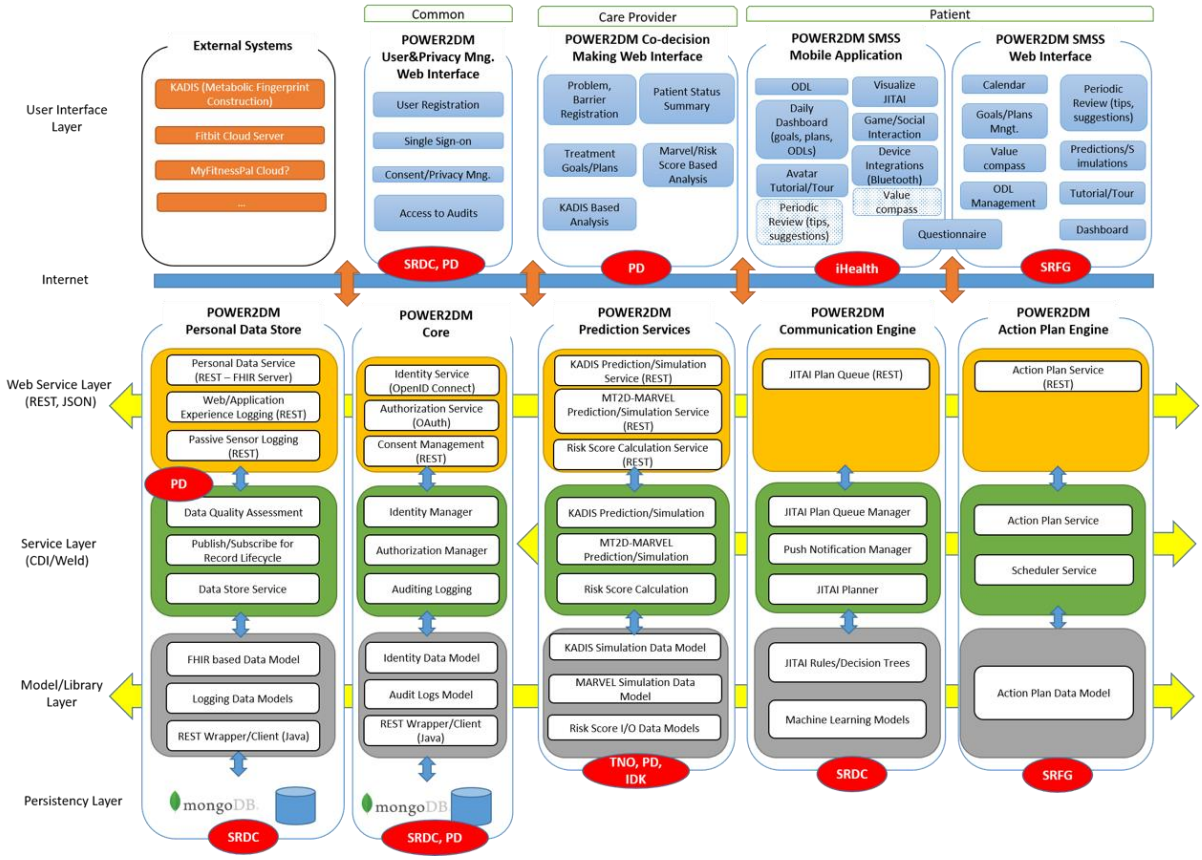


**Figure 1: POWER2DM Conceptual Design**

The Action Plan Engine represents a single pillar in the POWER2DM Conceptual Design and adds the Action Plan Engine UI (which is a major part of the POWER2DM SMSS Web Interface) on the user interface layer to the entire POWER2DM SMSS.

The POWER2DM user interface requires dedicated web services where the Action Plan Service represents the dedicated service endpoint for the Action Plan Engine UI. Figure 1 outlines the distinct modules of the POWER2DM SMSS. However, it also outlines the use of Internet technologies for service communication (REST/JSON) and the use of Dependency Injection (CDI/Weld) for service discovery. These technologies as well as the internal structure of the Action Plan Engine are explained in detail in the following sections.

## 2.2 Technology Foundation

External software dependencies have been carefully selected to bring in existing functionality with as little effort as possible, while still avoiding lock-in effects. Therefore, software products released under permissive open source licenses have been preferred over commercial third party software. In this section we describe the major libraries and frameworks used for the implementation of the Action Plan Engine and User Interfaces.

The Action Plan Engine is developed using Java Enterprise (JEE[1]) technologies. It requires a Servlet Container such as Apache Tomcat[2] or Jetty[3] as runtime environment. For the dependency injection the reference implementation CDI/Weld[4] is used. For the build and dependency management, Maven[5] as the de-facto standard is used to configure the distinct sub-components to be compiled into the final Action Plan Engine distribution.

The REST/JSON based web services for the user interface (UI) are implemented using the Jersey RESTful Web Services framework. A security layer based on Apache Shiro[6] is used to ensure that only authenticated and authorized users can access private resources.

The UI is implemented by means of a HTML5/CSS3/JS single-page web application focusing on a responsive, user-centered design. Native HTML5 components, such as web storage, are used for higher level functionality. The following open-source JS libraries are included in order to assure proper cross-browser support of complex functionality:

- Bootstrap: Responsive design framework (MIT license)
  http://getbootstrap.com/
- Fullcalendar: Calendar plugin (MIT license)
  https://fullcalendar.io/
- i18next: Internationalization (MIT license)
  http://i18next.com/
- jqPlot: Plotting and charting (MIT license)
  http://www.jqplot.com/
- jQuery: Multi-feature framework (MIT license)
  https://jquery.com/
- jsSHA: Client-side hashing (BSD license)
  https://caligatio.github.io/jsSHA/
- Marked: Markdown syntax support (MIT license)
  http://www.javascriptoo.com/marked
- momentJS: Date and time validation and manipulation (MIT license)
  https://momentjs.com/

Finally, the Action Plan Engine uses the rule engine "EasyRules"[7] for interventions.

## 2.3 Internal Structure and Data Model

The Action Plan Engine adheres to principles and technologies as shown above. It provides the backend intelligence service for POWER2DM SMSS Mobile and Web applications that helps patient manage his/her self-management goals and action plans. It contributes the Action Plan Data Model to the Model/Library Layer of the POWER2DM SMSS. It implements requested functionality to provide interventions based on the collected ODLs e.g. feedback on planned activities, motivational messages and tips for fostering or improving self-management activities. Finally, it exposes its functionality via

[1] http://www.oracle.com/technetwork/java/javaee/overview/index.html

[2] http://tomat.apache.org

[3] https://www.eclipse.org/jetty/

[4] http://weld.cdi-spec.org/

[5] http://maven.apache.org/

[6] https://shiro.apache.org/

[7] http://www.easyrules.org/

REST Web Services to the UI Layer of POWER2DM SMSS providing JSON encoded data to be used either in the Action Plan Engine UI or in the Mobile Application.

A more detailed component overview of the Action Plan Engine is given in Figure 2. As shown, the Action Plan UI communicates with its dedicated backend component. This backend component implements the required functionality to answer all requests from the UI.

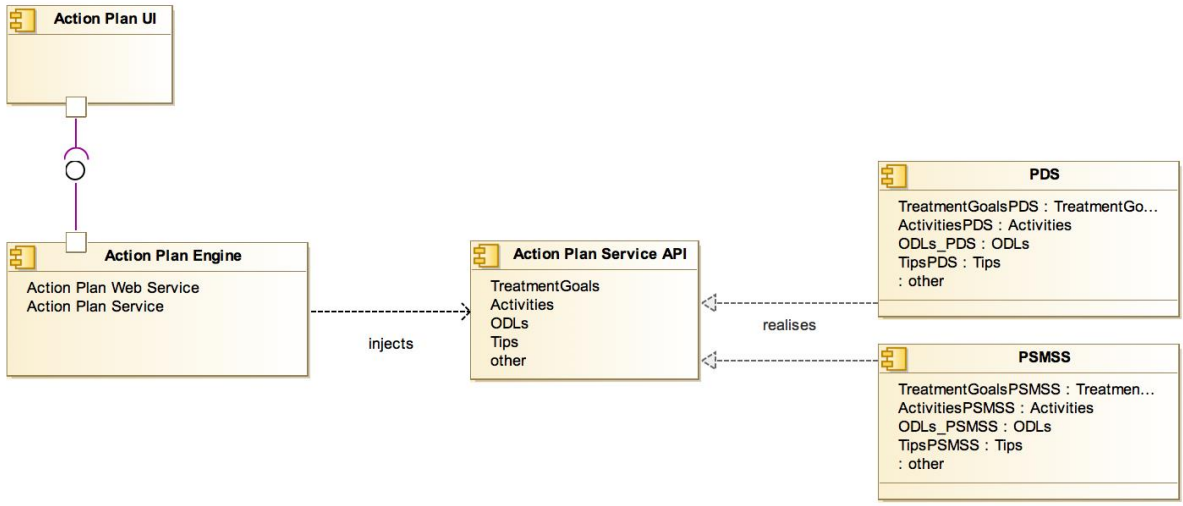


**Figure 2: Action Plan Engine Internal Structure**

In addition, the Action Plan Engine defines its own internal data model for applying the algorithms for matching activities to observations, describing schedules, calculating calendar events and providing reviews for self-assessment by the patient.

However, to show the actual data (e.g. treatment goals or activities) to the current user, the Action Plan Engine needs access to the storage (Persistency Layer), which is – in the context of POWER2DM SMSS – provided by the Personal Data Store. To allow using the Action Plan Engine with different storage solutions, the respective components are loosely coupled and injected on demand. This allows switching the used storage component or adding new implementations at any time. As a result of this approach, the Action Plan Engine is usable as a stand-alone component or integrated within bigger environments such as the POWER2DM SMSS, even without using its internal storage mechanism.

The Action Plan Engine uses the Personal Data Store (see Section 3.4) as its persistence layer in the context of the POWER2DM SMSS. In order to allow other usages, for example as stand-alone service, the Action Plan Engine is able to facilitate another persistence solution based on Apache Marmotta[8], a linked open data platform with a pluggable triple store. The use of a triple store allows flexible storage, indexing and retrieval of different data types, without a predefined schema. It therefore can be easily used for storing distinct data items such as observations (e.g. glucose measurement, blood pressure measurements, and food intakes), personal records, events, predefined messages or self-management goals and plans.

# 3 Action Plan Implementation

## 3.1 Project Structure Overview

The Action Plan Engine uses Maven as its build management. The Project Object Model (POM) in the project's root folder is used when building the distribution and defines the required dependencies and

[8] http://marmotta.apache.org/

sub modules to be included for each build. Maven supports a hierarchical structure of the modules and also allows the integration of pre-existing dependencies and libraries.

The current structure of the Action Plan Engine distribution is given in Figure 3.

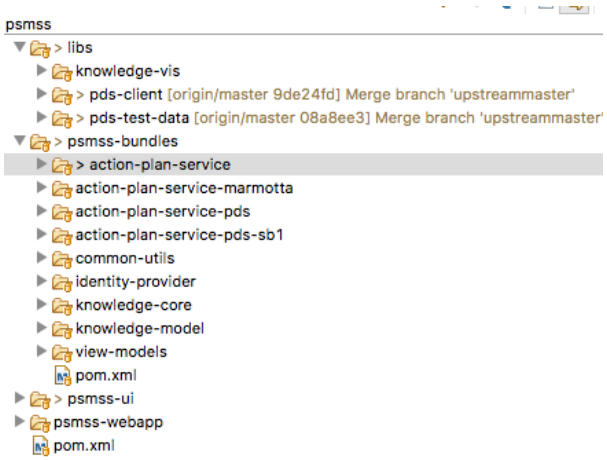


**Figure 3: Building Blocks of Action Plan Engine**

As shown in Figure 3, there is one module named `action-plan-service` and several modules named `action-plan-service-xxx`. Each of the modules holds one implementation to connect the Action Plan Engine to a provided and configured storage backend. Beside the backend modules, other modules contribute to the action plan engine. Depending on the project configuration, the modules are part of the final deployment.

Table 1 gives an overview of the project's (sub) modules and their usage in the different usage scenarios.

| Module | **Description** | **Notes** |
|---|---|---|
| action-plan-service | Core Module of the Action Plan Engine. This module provides the Web Interface layer and selects/uses the configured storage modules. | |
| action-plan-service-marmotta | Provides Marmotta-specific storage functionality. In a stand-alone environment, this is the default storage module for the Action Plan Engine. | Standalone Mode only |
| action-plan-service-pds | Provides the Connectivity to the Personal Data Store. This is the default storage module for the POWER2DM SMSS | To be used in the POWER2DM Pilot Studies |
| action-plan-service-pds-sb1 | Provides the data used for the Demo Mode – does not connect to a backend but provides static data based on Storyboard 1 (Simon). | To be used for demonstration purposes. |
| common-utils | As the name implies, helpful methods to be used throughout the Action Plan Engine | |
| identity-provider | Provides information of the logged user and provides the (secure) user specific secure identifier used for storing and retrieving patient health records. See section 3.3 for details. | |
| knowledge-core | Holds the functionality for reading and storing Patient Health Records in RDF based triple stores. | Standalone Mode only |
| knowledge-model | Specifies the (internal) data model for the RDF based storage | Standalone Mode only |
| knowledge-vis | Base functionality for visualizing arbitrary RDF graphs – used for the decision trees in standalone mode. | Standalone Mode only |
| view-models | Provide the exchange format for Patient Health records used by the Action Plan Engine UI (psmss-ui). | |
| psmss-ui | Keeps the HTML/JS Files for the Action Plan Engine Web Interface. Can be hosted on a separate web server, or deployed together with the Action Plan Engine Web Application onto a servlet container. | |
| psmss-webapp | Runtime project for the Action Plan Engine. Provides the runtime environment and configuration that is finally deployed in a Servlet Container. | |

**Table 1: Action Plan Engine Module Overview**

## 3.2 Storage Backend Selection

The distinct backend storage modules are selected based on a configuration value. Based on this configuration value, the respective backend storage endpoint is injected and used. Figure 4 outlines the following process:

1. The Action Plan Engine Webservice receives a request, e.g. for example to display the upcoming activities.
2. The (injected) Action Plan Service uses a ServiceFactory to obtain the requested service implementation.
3. From a list of all available service implementations, the desired implementation is selected.
4. The desired API method is invoked in order to retrieve the data properly formatted for the Action Plan Engine UI.
5. A dedicated transformer is used to ensure the validity and format of the returned data to the Action Plan Engine UI.

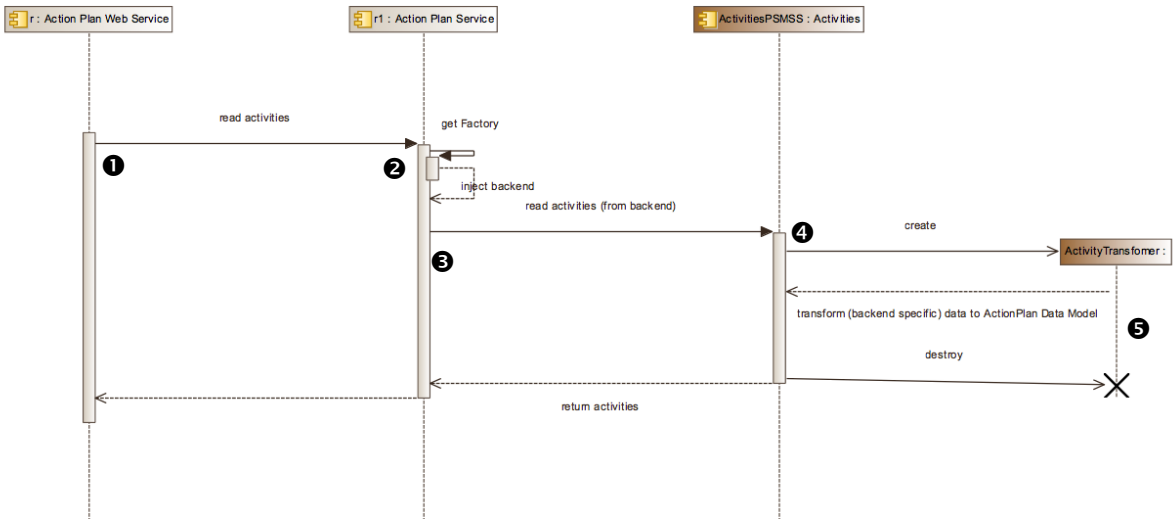


**Figure 4: Storage Endpoint Selection Example**

This approach allows adding new storage endpoints at any time in separate project modules, without the need to change the existing modules. A similar approach is used for the identity management described in the next section.

## 3.3 Security

Patient Health Records (PHRs) represent very sensible, personal data. As a result, it is common to not store any of the PHR (glucose, weight measurements) along with identifying data such as name, birthdate or user credentials. Instead, any of the PHR is identified with a secure identifier representing the owner of the PHR. This particular identifier must not be provided to the user; instead it must be maintained in a trusted identity management service holding the user credentials and other personal information.

The Action Plan Engine however requires this secure identifier upon any request in order to identify the user's PHRs. To achieve this, any of the requests from the Action Plan Engine Web UI must be accompanied with the current user's secure identifier, which in turn must be obtained from the trusted identity management service. Only in case the user is logged in and the identity management provides his/her secure identifier, the subsequent access to the backend storage components is possible.

The corresponding process is outlined in Figure 5.

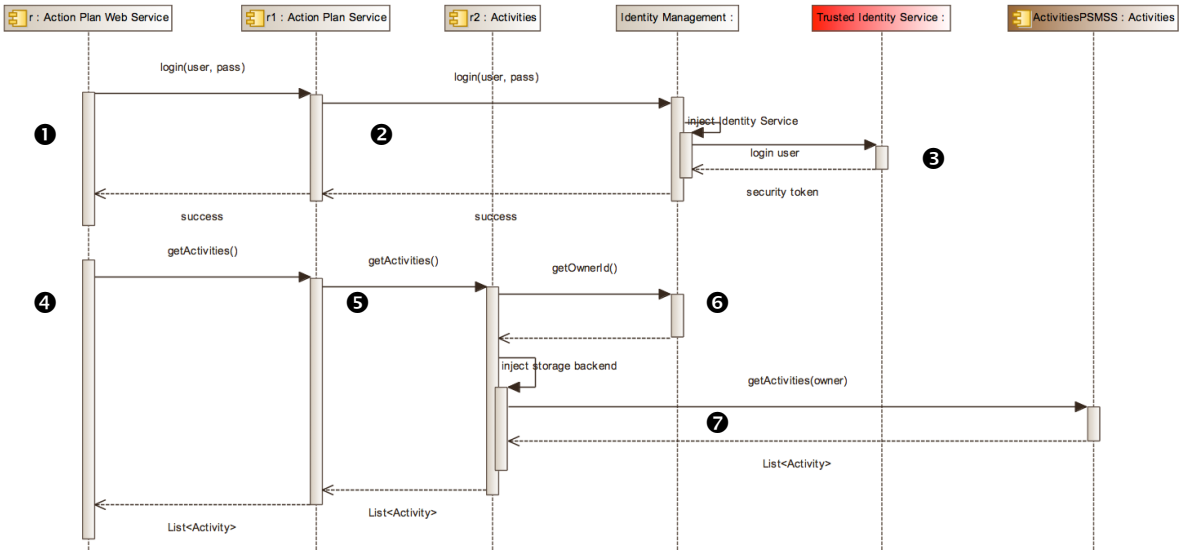


**Figure 5: Obtaining Logged User's secure identifier**

Whenever a user is not yet logged in, the Action Plan Engine UI displays a login window where the user can provide his/her login credentials (❶), which are passed along to the Action Plan Engine (❷). Internally, the identity management component is responsible to manage the logged users. This component has to select the currently configured trusted identity service (❸) and to verify the credentials. Such a trusted identity service can be an "Open ID Connect" endpoint as described in D4.9. This mechanism also allows integrating the services of the Action Plan Engine into a Single Sign-on environment.

Upon successful login, the user may access his/her data. The secure identifier must not be given to the Action Plan Engine UI for subsequent data access; instead it is retrieved from the currently logged user. Figure 5 outlines this process in step (❹) where the Action Plan Engine UI issues a request to the Action Plan Service. This request is handled by first delegating the request to the proper API service implementation (❺), obtaining the required secure identifier (❻) and finally selecting the configured storage backend and retrieving the requested data along with the secure identifier (❼).

As a result of this approach, no personal information is kept in a single database with patient health records.

## 3.4 Personal Data Store (PDS) Integration

In addition to internal data storage, the Action Plan Engine (APE) is also able to connect to external storages, such as the Personal Data Store (PDS) developed within POWER2DM. The PDS is used to store all patient related data and has been described in Deliverable D4.1 (Data Model) and D4.2 (Service API). Connectors and transformers form the APE to the PDS are described in this subsection.

### 3.4.1 PDS Health Data Model Transformers

The PDS Health data model contains the schema of all datatypes that can be stored in the PDS (D4.1). The major datatypes that need to be handled by the Action Plan Engine are:

- Observations and Clinical Results
- Goals (Treatment Goals and Self-Management Goals)
- Action Plans (Treatment Plans and Self-Management Plans)
- Appointments and Encounters
- Medication Orders and Administration
- User Settings

- User Profile Information

For each of the required data types, a separate Transformer implements the mappings between the PDS data model (Persistence, "P") and the view model ("V") that is delivered to the PSMSS web user interface.
Each Transformer should be able to transform a single persistence object or a list of objects from PDS object to the view objects delivered via web services.

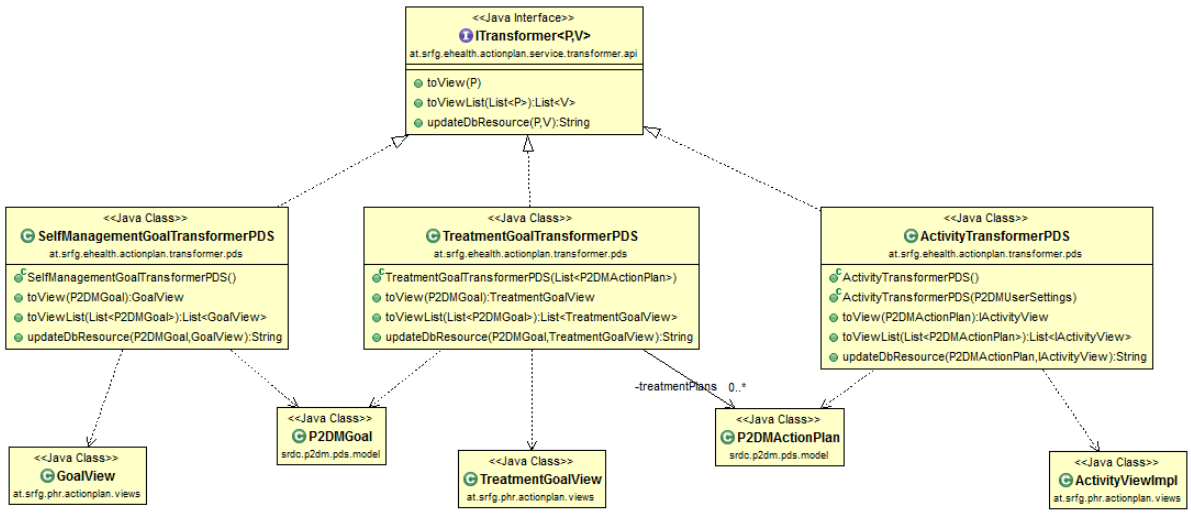


**Figure 6: Transformer Interface and Example Implementations**

Figure 6 shows three example implementations of the Transformer Interface for Activities, Self-Management Goals and Treatment Goals:

- The *Self-Management Goal Transformer* transforms a P2DM Goal and provides a simple Goal view.
- The *Treatment Goal Transformer* transforms a P2DM Goal with an associated P2DM Action Plan (e.g. Treatment plan) to a Treatment Goal view, which includes activities already planned during the Shared-Decision Making Phase.
- The *Activity Transformer* transforms a P2DM Action Plan to an Activity view. The Activity view is not only used for listing or editing activities, but also for generating single scheduled activity events within the Action Plan Calendar views according to the user assigned schedules of his/her activity plan and the P2DM User Settings[9].

Further Transformers are available for Prototype 1 are shown in Figure 7, additional transformers can be easily added on demand, e.g. after the updates of the POWER2DM Personal Health Model for Prototype 2.

[9] The user settings include user-defined timing information such as lunch, breakfast or dinner times, etc.

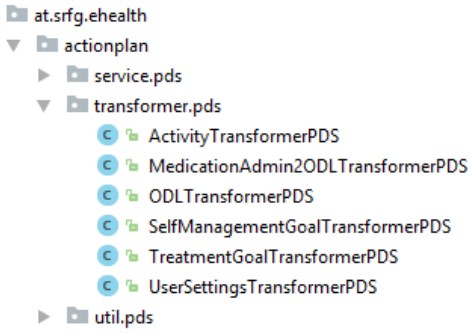


**Figure 7: ActionPlan Transformers for PDS**

### 3.4.2 PDS API Service Adaptors

Before the data elements can be transformed, they have to be queried from the data source. For each of the required data types, a separate service adaptor is responsible for querying the required data from the PDS. Before the results are returned, the service adaptor is using the corresponding transformer to generate the view models. This sequence is depicted in the steps 4 and 5 of Figure 4. For the PDS, two alternatives for Service Adaptors as depicted in Figure 8 have been implemented. One can access a remote PDS service and one returning static information based on the storyboard data.

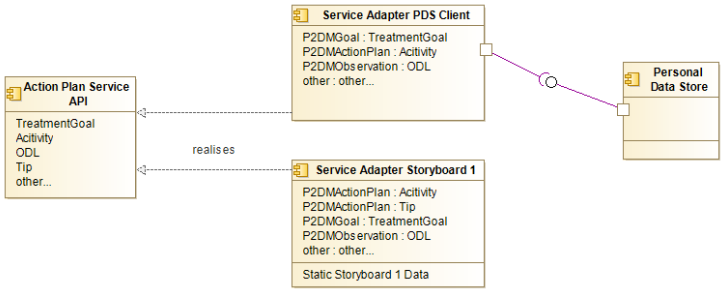


**Figure 8: Service Adaptors for Static and Dynamic use of PDS Data**

For development, testing and demonstration purposes, special, read-only data access service adaptors have been developed to serve the User Interface with prepared data. Storyboards have been developed in work-package 1 to illustrate the POWER2DM SMSS framework and care process. According to this process, individual steps have been pre-defined, which a patient is usually passing through when using the POWER2DM Self-Management Support System. Several steps of this process include the use of the Action Plan Engine (as background engine for the POWER2DM SMSS Web Application). To demonstrate the functionality of the SMSS, corresponding test data has been created using the PDS client API, which can now be directly injected into the backend as static storyboard data.

As shown in Figure 9, the normal username/password login screen provides additional “one-click” login buttons, which will be available in demonstration setups only:

- “Demo Account”: This will login with a pre-defined demo user, which is able to navigate through the different screens, add and remove data from the account. However, to keep this account clean, all data will be reset to an original status in predefined intervals. Further limitations can be applied for sake of security.
- Simon Persona Step 1 – 11: Using those buttons, the static storyboard data that should be available in the specific step will be provided. It is not possible to add/delete or modify data from this account.

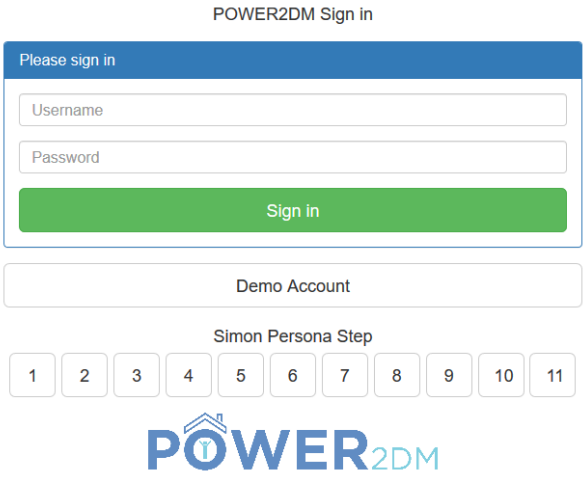


**Figure 9: Login Page with Selection of Storyboard Step**

For Prototype I, test data from the "Simon" storyboard (Focus: T1D – Blood Glucose Measurement) is used. The following test data is currently provided and integrated from the storyboard:

- Step 1: Initial Resources (Users, Organisations, Devices, etc.)
- Step 2: Basic Patient Data (Conditions, Clinical Observations, etc.)
- Step 3: KADIS treatment goals and plans
- Step 4: Self-Management goals adopted from KADIS treatment goals and plans
- Step 5: Journal Data (ODLs) for the KADIS data collection period
- Step 7: New Treatment Goals for the upcoming period (30 days)
- Step 8: New Self-Management Goals and Plans for ODLs for the upcoming period

Further steps and data will be added for Prototype 2.

## 3.5 GUI Overview

### 3.5.1 UI Components

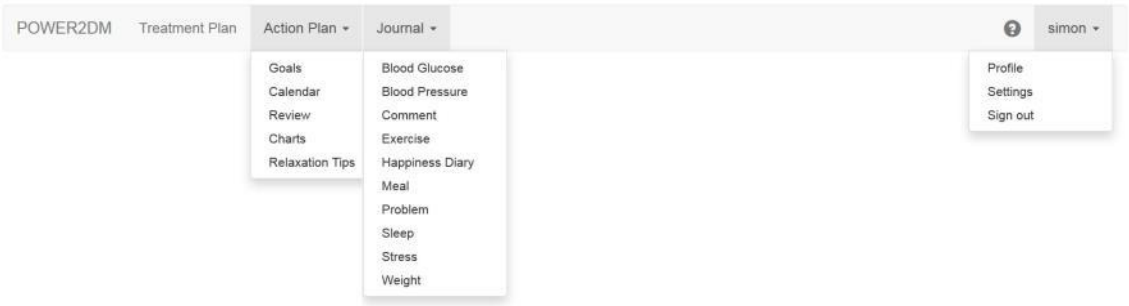


**Figure 10: Main navigation**

The UI is composed of the following elements:

- Dashboard (reachable by clicking on "POWER2DM"): Overview page of the most relevant data (such as performance of past week, upcoming activities, charts). Presented elements can be customized by the user in their settings.
- Treatment Plan: Lists relevant goals and activities ordered by a practitioner. This list is read-only but goals and activities can be imported to the Action Plan.
- Action Plan: Handles all components related to self-management.

  - Goals: Lists relevant self-management goals (Self-management Goals can be referenced to Treatment Goals).
  - Calendar: Presents self-management activities in an adjustable calendar view (Activities can be referenced to Self-management Goals).
  - Review: Shows a performance summary for the past week (or month, quarter, etc.). Detailed information is given in Section 3.7 “Review and Interventions”.
  - Charts: Presents recorded patient ODL data in chart views with an adaptable timeframe.
  - Relaxation Tips: Lists customizable tips presented to the user in case of stress situations.
- Journal: Lists recorded patient ODL data in various categories.
- Profile and Settings (reachable by clicking on the user name drop-down): Collects all patient-related (Profile) and display-related (Settings) data.

Detailed Mock-ups and Screenshots for each menu item are available in D3.7.

### 3.5.2 Single-page Layout

The web application pursues a single-page layout by loading content dynamically using various methods described in this section. In general, all globally relevant HTML and JS content is added to the central index.html file while the corresponding content for each view is retrieved on runtime.
In order to retain the browser native back/forward/reload methods and allow for deep linking the URL hash notation is used (e.g. https://someurl.com/#somepage). The hashes can be broken down further for additional functional or navigational purposes using underscores (e.g. https://someurl.com/#someaddition_somepage).

Whenever the URL hash changes (**onhashchange** event), the function **init()** is called. It splits the current location href into base and hash and passes the hash on for further processing.
The function **show(page,id)** handles loading the relevant resources into the DOM. It splits the hash by underscores (in case there are any), loads the HTML file for the requested page and passes on the page and additional underscore attributes for further processing. By default, the HTML file is loaded into the main content DIV but the id parameter allows for loading content into any DIV – this enables the developer to assemble a page using any amount and layout of components without code duplication.

The function **performAction(page,add)** takes care of executing onload methods (e.g. loading data via AJAX) and processing the additional underscore attributes.

### 3.5.3 Dynamic Content Generation

Some content and layout types are very congruent throughout HTML pages (e.g. input forms). In order to keep code duplication and complexity to a minimum, JSON templates are used for definitions that are translated to HTML using JS on runtime.
For example, all ODL tables and input forms are generated from the following template format:

```
{
    “ODLTYPE”: {
        “label”: STRING,
        “ATTRIBUTE”: {
            “chart”: BOOL,
            “dataType”: STRING,
            “display”: STRING,
            “label”: STRING,
            “outer”: STRING,
            “type”: STRING,
            “unit”: STRING,
```

```
                    “values”: [STRING]
                }
        }
}
```

Each ODLTYPE has a label (an internationalization tag as a STRING) for display purposes and various ATTRIBUTEs. Those contain the following keys:

- chart: BOOL indicating whether or not the attribute should be displayed in the charts
- dataType: The datatype of the attribute (e.g. “STRING”, “NUMBER”, “ISO_DATE”, etc.)
- display: STRING indicating if and where to display the attribute
- label: Internationalization tag as a STRING for display purposes (attribute name)
- outer: Potential outer JSON key as STRING when retrieving/sending the attribute
- type: STRING indicating the attribute type (e.g. “REQUIRED”, “OPTIONAL”, “GENERATED”, …)
- unit: Internationalization tag as a STRING for display purposes (attribute unit name)
- values: ARRAY of internationalization tags as STRINGs in case of predefined answers for multiple choice or likert scale dataTypes

## 3.6 Decision Trees

### 3.6.1 Decision Tree Modelling

A separate tool has been developed using the following open-source JS libraries:

- Bootstrap: Responsive design framework (MIT license) http://getbootstrap.com/
- jQuery: Multi-feature framework (MIT license) https://jquery.com/
- visJS: Visualization framework (Apache 2.0 and MIT license) http://visjs.org/

The general purpose is to enable non-technical partners (e.g. practitioners) to define decision tree workflows in a simple yet structured way that allows for a consistent and fast implementation on the technical level.

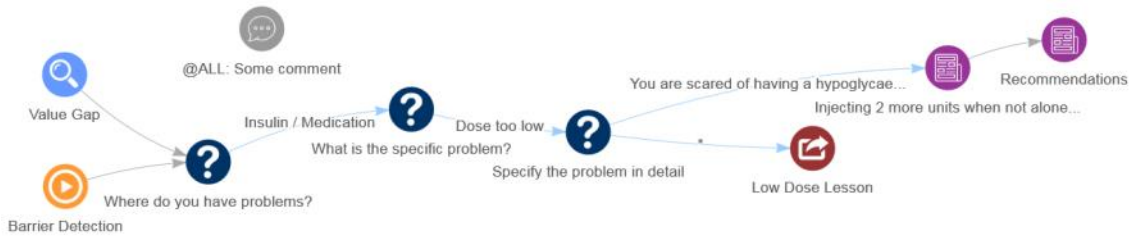


**Figure 11: Example decision tree workflow**

There are various node types available:

- Action: Manually activate specific triggers
- Comment: Add comments when collaborating on a workflow
- Content: Show recommendations, specific pages, static content, etc. to the user
- Condition: Check specific conditions
- Question: Any type of question
- Trigger: Manually or periodically trigger a (sub-)workflow

Furthermore, there are two edge types:

- Answer: Forward to the next node if the response data of the preceding question matches
- Forward: Forward after the preceding node has finished

A complete manual on the usage of the tool and associated notations can be found here: https://power2dm.salzburgresearch.at/workflow_tool/docs/workflow_tool_manual.pdf

### 3.6.2 Decision Tree Execution

The decision tree execution process is based on JSON templates that are generated from visJS JSON exports and translated to the following format:

```
{
        "DECISIONTREE": {
                "id": STRING,
                "NODE": {
                        "id": STRING,
                        "text": STRING,
                        "options": [
                                {
                                        "text": STRING,
                                        "action": STRING
                                }
                        ]
                }
        }
}
```

Each DECISIONTREE has an id as a STRING to reference it and various NODEs. Those contain the following keys:

- id: STRING to reference the node (special id "start" for first node)
- text: Internationalization tag as a STRING of the node / question text
- options: ARRAY of JSON elements with the following keys:
    - text: Internationalization tag as a STRING of the answer text
    - action: STRING to reference the target node (special action "end" for last node)

More complex functionality (such as different dataType inputs, condition checking, back-end integration, etc.) is not yet supported since it is out of scope for the first prototype.
For the second prototype this JSON structure and associated functionality is going to be expanded and automated export methods to the open-source tool Node-RED[10] (Apache 2.0 license) will allow for easy back-end integration.

## 3.7 Review and Interventions

The first prototype review component supports the periodic review cycle that the patient is encouraged to perform at least on a weekly basis. By default the last seven days are assessed, although the patient can modified the assessment time over longer periods.

The Action Plan review does following Kate Lorig's[11] model for patient self-management, providing an action plan to record an action plan that enables a patient to plan and thereby schedule an activity such as glucose monitoring. The adherence performance scoring mainly aims to support behavior

[10] https://nodered.org/
[11] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1497631/

change and provide intervention tips to help them adhere to an activity plan to support his/her goals designed by the patient following treatment recommendations.

### 3.7.1 Performance Assessment

For the current prototype, only adherence or compliance performance is considered. The patient might adhere to the activity plan she first made and monitor herself by recording (or automatically) by posting activity ODLs or acknowledgements that she performed the planned activity. However, so as not demotivate anyone, simply posting ODLs is also fine. For the second prototype, additional user settings should be provided to enable the patient to define their performance assessment rules.

#### 3.7.1.1 Activity adherence performance

The performance is based on the adherence to the planned activity for the given time interval selected by the user. A score is calculated to not only give feedback to the patient, but also trigger an appropriate tip or textual intervention. The text tip can include additional links to either activate a decision tree enabled intervention or provide supplemental external infos (videos, patient info, and additional tips)

The current implementation focuses on supporting behavior change. Instead of providing "judgmental" assessment of goals based on measurements e.g. Body weight goals, the behavior change focuses on the assessment of performing small efforts, e.g. activities, that will slowly lead to the achievement of particular goals. Activities are scheduled to support the building of habits. Scoring of activity performance is based on adherence or compliance to a schedule plan. For particular ODL activities, scoring might require adherence to a strict time e.g. eat a snack at about 10:00 with a narrow time window (+/- 30 minutes) versus go jogging at 18:00 with a wider time window (+/- 3 hours). In the future, users should set their own scoring rules. However, even if the patient chooses not to create an activity plan, they are scored positively when they at least record their activities.

#### 3.7.1.2 Goal performance based on activity adherence (performance)

The adherence performance of each goal, consequently, is based on one or more activities. A goal's associated activities are assigned when a new activity is created; the patient is asked to assign one or more previously defined goals. Furthermore, the activity can be updated anytime to update the assigned goals.

### 3.7.2 Implementation Overview

#### 3.7.2.1 Review Services

Review Services build a review evaluation object that can be shared with various components, including the UI. The review evaluation object includes scored activities, goals and their respective intervention tips that include internal links (decision tree links) or external links (videos, web sites, etc.). The intervention table was created by domain experts and rules were derived from the table. The tip text format supports a lightweight markup language called Markdown[12] so there is no dependency on HTML for URLs, or lists formatting.

[12] https://en.wikipedia.org/wiki/Markdown

#### 3.7.2.2 Rule Engine Component

Initially, the decision support components included a rule engine was based on JBOSS Drools[13] and the rules were based on drools rule language (DRL). However, the implementation later migrated to Easy Rules[14] mainly because these rules are easier to maintain and debug.

The following steps outline the general rule template used to evaluate parametrized data from the Intervention table. In the next prototype, these parameters will be stored in an application rule configuration file.

Firing of rule evaluations, filter or validate

1. Condition: Filter Evaluation Categories
   E.g. overall performance evaluation,
   goal by activity type evaluation,
   or activity by type evaluation

2. Condition: Filter ODL e.g. exercise, glucose monitoring, medication

3. Condition: Filter ODL subtype(s): medication[insulin] or exercise[jogging]

4. Condition: Filter Patient Profile
   [Demographics e.g. gender & age, conditions, risks, problems]

5. Condition: Compare performance factors
   a. Adherence
   b. Or ODL average value or activity occurrences

6. Perform Action(s)
   a. Assigned appropriate Intervention text group ID specified for the rule parameters. Each group contains one or more text variations.

7. Perform Summarization Action(s)
   a. For each text group ID, choose one or more random intervention texts.

#### 3.7.2.3 Intervention Table - Intervention Motivational Tips

The domain experts provided a table that was subsequently modified for modification to better group the variations of intervention texts. One group is associated with a rule and from that group one or more tips are randomly selected so that the patient does not always receive the same intervention motivational tip.

Table 2 is an extract of the intervention table, although the domain experts provided addition details to build the rule conditions associated with each rule text group ID. Texts utilize Markdown especially to represent hyperlinks or lists e.g. *[label or text](hyperlink*). Hyperlinks containing only an anchor symbol ("#") are internal links, especially to enable the user to start a particular decision tree process in the decision tree UI or to begin a diary, etc. e.g. at row 3/011:

[13] https://www.drools.org/
[14] http://www.easyrules.org/

*You can use our [well-done diary](#well-done-diary)*

The complete intervention table will be part of D3.2 (“Dynamic Behavior Change Intervention Models for Self-Management II”).

**Table 2 – Intervention Table sample**

| Rule Text Group ID | Text ID | Intervention Text |
|---|---|---|
| 2 | 004 | You are making progress on your goal. Good! There might also be things that you would still like to improve. Could you think about what could you do different next week? Try to plan one specific thing for the next week, of which you think it may even further improve your progress. You can plan it on your Power2DM calendar/agenda. |
| 3 | 009 | "Well done! You are doing a great job" |
| 3 | 010 | Keep up the good work! Hopefully the progress on your goal makes you feel more positive about yourself. Try to reflect on positive feelings about making progress on this goal. |
| 3 | 011 | You are doing well! Try to consciously think about how it makes you feel to achieve your goal. Do you feel proud, confident, happy, cheerful, excited? Or is there another positive emotion that you recognize? Think about it, and try to write down all the positive feelings that you have. Even the "smallest" positive feelings are important to recognize. You can use our [well-done diary](*#well-done-diary)* |
| 4 | 012 | Do you feel adequately informed about diabetes and its effects on you? If yes, that is important and really good! If not, you might take a look the *[following information material](http://www.diabetes.co.uk/emotions/)*. Many people with diabetes underestimate the effect it has on them, or feel that they should not complain about it. |
| 5 | 013 | Good that you are still trying! Continue your good work! |
| 5 | 014 | Keep going! It might help to consider in which situations you have already succeeded in achieving your goals |
| 5 | 015 | Good that you are still trying! Continue your good work! |
| 5 | 016 | Have you made your goal sufficiently clear and realistic so that it can reach it with reasonable effort? If not, try to reformulate your goal and your activity planning and put them into practice. |