LLMCMSS

Long Lasting Memories

Central Management System

Functional and Implementation Specification

IMPLEMENTATION READY V 0.9

Author: Wolfgang Scherer

Date Issued: 16-MAR-2010

Contents

Date Issued: 16-MAR-2010 1

LLMCMSS.01 - LLM CMS Specification Main Part 6

ATTENTION: The online-versions of this pages in the LLM-Wiki shall only be read with Firefox (3.5 or newer) browsers! Microsoft Internet Explorer does not render some styles used by the Wiki framework correctly. 6

Contents: 6

Overview 6

LLMCMSS.01.00 Document Control 6

LLMCMSS.01.02 Nature of this Specification 7

LLMCMSS.01.03 Functional Specification Areas 7

LLMCMSS.01.04 - Non-functional specification areas 8

LLMCMSS.01.05 - Implementation and Architectural Areas 8

LLMCMSS.01.06 - Optional System Features 8

Motivation for Optionality 8

LLMCMSS.01.06.02 - Optional parts not part of LLM System Version 1 9

LLMCMSS.01.06.03 - Optional part included in LLM System Version 1 9

LLMCMSS.01.07 - Conventions in this Specification 9

LLMCMSS.01.08 - Sources and additional information for this Specification 10

LLMCMSS.01.09 - Derived Requirements for other Parts of the LLM System 10

LLMCMSS.01.10 - Parts of this specification 10

LLMCMSS.01.10.01 - Listing of parts 10

LLMCMSS.01.10.02 - Recommended Reading Sequence 11

LLMCMSS.02 - LLM LUI Functional Specification 12

LLMCMSS.02.00 Document Control 12

LLMCMSS.02.01 General 12

LLMCMSS.02.02 Usage Scenarios 12

LLMCMSS.03 - LLM CMS LUI functional specification 16
LLMCMSS.03.00 Document Control 16
LLMCMSS.03.01 CMS LUI functions 16
LLMCMSS.03.02 CMS LUI Implementation requirements 17
LLMCMSS.04 - CMS service component functional specification 18
LLMCMSS.04.00 Document Control 18
LLMCMSS.04.02 Overview 18
LLMCMSS.04.02 CMS API functions 19
LLMCMSS.04.03 CMS function interfaces 22
LLMCMSS.04.04 CMS implementation issues 22
LLMCMSS.04.05 CMS RUI functionality 23
LLMCMSS.05 - LLM Local User Interface Framework (LLF) Specification 27
LLMCMSS.05.00 Document Control 27
LLMCMSS.05.01 LLF platform 27
LLMCMSS.05.02 LLF Functionality 27
LLMLCMSS.05.03 - LLF Implementation and Modularity 29
LLMCMSS.05.04 - LLF Configuration Management 30
LLMCMSS.06 - LLM CMS Use Cases 31
LLMCMSS.06.00 - Document Control 31
LLMCMSS.06.01 - LLM Service Scenarios - LLMTOS.05.01 31
LLMCMSS.06.02 - LLM Use Case Scenarios - LLMTOS.05.02 32
LLMCMSS.06.03 - LLM Use Case Model - LLMTOS.05.03 36
LLMCMSS.06.04 - LUI Screens 37
LLMCMSS.06.05 - administration and management use cases 42
LLMCMSS.06.06 - RUI screens 43
LLMCMSS.07 - LLM CMS Aspects: Configuration, Security, Modularity, Performance, Volumetrics ... 47
LLMCMSS.07.00 - Document Control 47
LLMCMSS.07.02 - Aspects Details 47

LLMCMSS.08 - LRF functional specification 52
LLMCMSS.08.00 - Document Control 52
LLMCMSS.08.01 LRF Motivation and driving factors 52
LLMCMSS.08.02 LRF Functional parameters 53
LLMCMSS.08.03 Report and Feedback Algorithm Examples 54
LLMCMSS.09 - CMS requirements in LLM-Database 56
LLMCMSS.09.00 - Document Control 56
LLMCMSS.09.03 LLM-DB interface required by CMS function 56
LLMCMSS.10 - LLM CMS reference architecture 59
LLMCMSS.10.02 - Concepts in the reference architecture 60
LLMCMSS.10.02.01 - Seperated and not separated components: LUI and RUI 60
LLMCMSS.10.02.02 - Integration of already existing components 61
LLMCMSS.10.02.03 - LUI - Local User Interface and LLF - Local user interface framework 61
LLMCMSS.10.03 - Components of the reference architecture 61
LLMCMSS.10.03.01 - 1.1.1 - The LLM-LUI 61
LLMCMSS.10.03.02 - 1.2.1 - The ILC-LUI 62
LLMCMSS.10.03.03 - 1.2.3 - The CTC-wrapper-LUI 62
LLMCMSS.10.03.04 - 1.2.2 - The CMS-LUI 63
LLMCMSS.10.03.05 - 1.3.2 - The CMS 63
LLMCMSS.10.03.06 - 1.5.5 - The LLM-Database / LLM-DB 63
LLMCMSS.10.03.07 - 1.4.5 - The LRF - LLM reporting and feedback functionality 63
LLMCMSS.10.03.08 - 1.4.4 - The PTC - Physical training component 64
LLMCMSS.10.03.09 - The LLF instance 64
LLMCMSS.10.03.10 EXTAUTH - The optional external authenticator 64
LLM Abbreviations 66
LLM CMS Specification Appendix B - Questions and Answers 68
LLMCMSS.B.00 - Document Control 68
LLMCMSS.B.01 - Overview and Motivation 68

LLMCMSS.B.02 - Questions and answers in Detail 68

LLMCMSS.01 - LLM CMS Specification Main Part

This page and its child pages contain the specification work and results for the LLM Central management system (LLMCMS).

ATTENTION: The online-versions of this pages in the LLM-Wiki shall only be read with Firefox (3.5 or newer) browsers! Microsoft Internet Explorer does not render some styles used by the Wiki framework correctly.

Contents:

1. Overview
2. Nature of this specification
3. Functional specification areas
4. Non-functional specification areas
5. Implementation and architectural requirements
6. Optional system features
7. Conventions in this specification
8. Sources and additional information for this Specification
9. Derived Requirements for other parts of the LLM system
10. Parts of this specification

Overview

The LLMCMS is part of the LLM system.

The LLM system, detailed in LLMTOS is the technical system providing the LLM service, also detailed in LLMTOS.

The LLMCMS shall provide all necessary over-all management functionality in the LLM system:

- Administering user account data (Password, user credentials)
- Administration of training data (programs, schedule, results, performance)
- Presentation of reports about training performance
- provide a local user interface for the LLM system user
- provide a remote user interface for local and remote system administration

NOTE: This specification consists of 10 parts. When you are starting to study this specification afresh, the recommended reading sequence is given in the parts section of this part.

LLMCMSS.01.00 Document Control

LLMCMSS.01 document properties	
Item	**Value**
document/part	LLMCMSS.01

Issue	0.9
History - 0.1	Created
History - 0.2	added LLMCMSS.08
History - 0.3	updated to Ref Arch V .03
History - 0.4	added LLMCMSS.09
History - 0.5	added document control and updated parts list
History - 0.6	- correction of typos and chapter numbering - include LLF and LRF in the implementation areas
History - 0.7	- refer to reference architecture V 0.4
History - 0.8	- refer to reference architecture V 0.7 - make list of part to hyperlinks - add numbering-constance requirements in LLMCMSS.01.07.03 - add hint for reading with FireFox only, IE problems. - list LLMCMSS.05 and LLMCMSS.09 as ENVREQ's explicitely - refer PTC and CTC integration modularity issues in LLMCMSS.01.03
History - 0.9	- added recommended reading sequence

LLMCMSS.01.02 Nature of this Specification

The specification is intended to be 2 things in one:

1. A requirements specification, containing all the requirements for the LLM-CMS, functional, non-functional, etc.
2. An implementation specification, allowing to be handed over to an implementation software team to be the complete base to implement the software comprising the LLF, LRF, LLM-LUI, the CMS-LUI and the CMS components

By this approach, any traceability issues between requirements and implementation specification should be able to be minimized.

As a result, this approach adds further areas to the specification:

- requirements concerning specific structural elements of the implemented software.
  These may also result from project constraints and already existing components, e.g. the LUI already existing in the eHome system.
- Implementation and modularity hints and rules in every part of the specification

The specification builds on a Reference Architecture (open picture in new window for reference)

LLMCMSS.01.03 Functional Specification Areas

This specification specifies implementation requirements for 5 Components of the reference architecture architecture:

- The LLM-LUI, specified in LLMCMSS.02
- The CMS-LUI, specificied in LLMCMSS.03
- The CMS itself, specified in LLMCMSS.04, including its RUI, accessing the LLM-Database to manipulate and investigate persistent data and to indirectly control the operative components ILC, PTC and CTC and including notification and workflow processing

- The LLF - LLM system local user interface framework, on which all the components of the LUI are based, specified in LLMCMSS.05
- The LRF - LLM reporting and feedback functional component, specified in LLMCMSS.08

However, there are a set of components in the reference architecture whaich are NOT covered in LLMCMSS as they are not part of the LLM CMS subsystem:

- The CTC and PTC integration. However, there are assumptions about the behaviour of these components that have impact on LLMCMSS.06. Furthermore, the modularity issues to be considered in order to efficiently integrate these components are covered specifically in LLMCMSS.07.02.01.02.

LLMCMSS.01.04 - Non-functional specification areas

The specification also specifies other areas:

1. models and concepts for data management
2. interfaces between components and (if necessary) internal to components
3. volumetrics
4. Performance
5. Availability
6. Safety
7. Security
8. Modularity
9. ~~Multi-Tenancy -~~ NOT COVERED, no requirement
10. LLMCMSS.06 - Use-Cases: e-Home, Day-Care-Center, Clinical Environment, ~~Home-Care Professional~~
11. Scalability
12. ~~Accounting~~ - NOT COVERED, no requirements
13. Evidence Collection & Archiving

LLMCMSS.01.05 - Implementation and Architectural Areas

- LLMCMSS.05 - The LLM LUI Framework (LLF) specification
- LLMCMSS.08 - The LLM Reporting and Feedback functionality (LRF) specification
- LLMCMSS.10 - The LLM system reference architecture
- LLMCMSS.07 - General and Implementation Aspects

LLMCMSS.01.06 - Optional System Features

This section lists optional system features and separates those included in version 1 of the LLM system implementation from those left for later versions.

Motivation for Optionality

This Specification describes part of the first version of the LLM system. The extent to which features can be implemented is determined by the project resources in the LLM project itself: project timeline and available development resources and funding.

Although the LLM project in principle tries to use existing components, there is a development effort necessary to develop the "glue" functionality. Especially the CMS and the LUI framework are part of this "Glue" functionality providing a more seamless look-and-feel and an extended amount of manageability of otherwise separated sub-systems. The LLMTOS describes a rather large number of ideas and concepts assisting in boosting user-friendlyness and lowering usage barriers, be it physiological or psychological.

However, as project timeframe is limited (the core aim of the project, the operational trial phase, has to commence as soon as possible) and so are person-hour-resources for software development and integration. All that said, it appears that not all usability features sketched in LLMTOS will make it into version 1 of the LLM system.

LLMCMSS.01.06.02 - Optional parts not part of LLM System Version 1

For the current version of the LLM system the following parts are considered optional and, as of now, not part of version 1 implementation:

1. Avatar implementation
2. Gesture support for enhanced controllers like Wii-Remote or Multi-Touch
3. Voice Recognition

However, implementation of Version 1 shall not build up roadblocks or unnecessary difficulties for a later incorporation of these features in later versions of the system. Therefore, implementation must be with such expansion in mind. Designs shall consider hooks and other expansion mechanisms as far as the knowledge of possible integration of the later-to-be-added features is at the time of design. Reviews must contain checks for this expasion mechanisms while keeping overhead low in order not to waste resources where they should be saved.

LLMCMSS.01.06.03 - Optional part included in LLM System Version 1

However, of all the optional parts there may be a few that are indespensable when it comes to at least providing the least required level of user-friendlyness and adaptation to the needs of the target clientéle. As of the time being, these are:

1. elimination of keyboard typed input for authentication: logging on by contactless ID card. However, there are methods existing for authentication by security devices like fingerprint-scanners or other devices. These methods shall be used along with the software support provided in the target operation system(s).
   To further ease acceptance, mobile device methods like NFC/GSM-interoperability shall be checked in conjunction with movement detection also feasable with these devices.

LLMCMSS.01.07 - Conventions in this Specification

1. Where this specification is referenced, it will be called **LLMCMSS**.
2. As this work is a specification, it is extremely important to cater for traceability. By tracebility, we understand the connections between different requirements and between the different types of requirements. These connections allow to determine if a requirement is caused by another one or is a pre-requiisite for another one. In order to establish these connections, all parts of this specifications are uniquely numbered by a hierarchical naming system. These numbers are prefixed by the ID of the specification itself.

An example for such a requirements ID is: LLMCMSS.02.01.02 which refers to the 2nd paragraph in the first chapter of the LLM LUI specification, being part 02 of this specification (LLMCMSS).

3. In order to keep requirements-IDs constants, lists in LLMCMSS shall only be appended to. New elements shall never be insterted in the midlle of a list in order for the already existing requirements to keep their numbering. In the same way, requirements rendered obsolete shall never be deleted from a list, but rather marked as "OBSOLETE:" or striked-out (~~like this~~).
4. Other referenced documents will be used in a similar, appropriate manner.
5. When it comes to references to an external document (external means not part of the LLMCMSS), only a specific issue of the respective document will be referenced. If it is impossible to rely on a consistent version numbering and administration in the external document, a snapshot of the external document will be taken and kept as a copy with LLMCMSS.

LLMCMSS.01.08 - Sources and additional information for this Specification

[LLMTOS] D3.1 LLM technical and operational specifications Issue v1.9 at the files section of Work Package 3 of the LLM site. There are tracebility issues in connection with this document as it may change in the future. To have at least one stable reference point, issue 1.9 will be tried to be marked up and linked with requirements items in the LLMCMSS. To assist this case, the document will be referenced here as **LLMTOS *(trace/link Work to be continued)***.

LLMCMSS.01.09 - Derived Requirements for other Parts of the LLM System

At numerous places in this specification, the specified functionality requires pre-requisites or certain functionality from other parts of the LLM system. At the time of writing for this specification, it is not known if the referred other parts of the LLM system are able to fulfil these requirements. One exemption from this is the LLM-DB, because its functionality for the time being is well defined by the LLM web-service reference. Anyway, the requirements to other system parts will be marked with "**ENVREQ:**" to denote a requirement of the LLM CMS against its environment. At review of this specification the ENVREQs will have to be checked and appropriate work packages scheduled to implement tham if not yet existing.

Specifically mentioned here shall be parts LLMCMSS.05 and LLMCMSS.09 . These two parts are, by definition, ENVREQs in their totality as they define requirements for external components.

LLMCMSS.01.10 - Parts of this specification

This specification consists of 10 parts. These parts, together with the documents referenced in the sources section above, comprise everything necessary for implementing LLMCMS. No other information source shall be necessary.

LLMCMSS.01.10.01 - Listing of parts

1. LLMCMSS.01 - this part and top level of the hierarchy
2. LLMCMSS.02 - the LLM LUI specification
3. LLMCMSS.03 - the CMS LUI specification
4. LLMCMSS.04 - the CMS service specification

5. LLMCMSS.05 - the LLF - LLM LUI Framework - specification
6. LLMCMSS.06 - LLM CMS Use Cases
7. LLMCMSS.07 - LLM CMS Aspects and Implementation Issues
   This part is concerned with Topics like Performance, Security, Safety, Modularity, Implementation and other orthogonal aspects
8. LLMCMSS.08 - LLM reporting and user feedback data and flow models and implementation concepts
   This part specifies the function to flexibly and expandaby generate human readable and comprehensible reports and feedback information from performance and logging data throughout the LLM system or even from external systems. The common denominator is that the input information has to be stored in the LLM-database before and the result is generated in HTML form.
9. LLMCMSS.09 - CMS requirements in LLM-Database
   This part summarizes everything the CMS subsystem specified in this specification requires to be provided by the LLM-database component.
10. LLMCMSS.10 - LLM CMS Reference Architecture

LLMCMSS.01.10.02 - Recommended Reading Sequence

If you are starting studying this section from scratch, the recommended sequence of reading is:

- 01 - Main Part
- 06 - Use Cases
- 10 - Architecture
- 02 - LLM LUI specification
- 03 - CMS LUI specification
- 04 - CMS service specification
- 05 - LLF specification
- 08 - LRF specification
- 09 - LLM-DB requirements
- 07 - Aspects and Implementation

LLMCMSS.02 - LLM LUI Functional Specification

The LLM system Local User Interface - LLM LUI - is the central entry point for the local user access to the LLM system.

In other parts of this document it will be referenced as LLMLUI or LLMCMSS.02.

LLMCMSS.02.00 Document Control

LLMCMSS.02 document properties	
Item	Value
Document	LLMCMSS.02
Issue	0.4
History - 0.1	Created based on Reference Architecture V 0.1
History - 0.2	Updated revised section format
History - 0.3	- no multiple instance login in LLMCMSS.02.01.02, obsoleting .2,.3,.4 - moving user admin to RUI, thus moving LLMCMSS.02.02.03 to LLMCMSS.04.05.01 - schedules: LUI only shows, need RUI (LLMCMSS.04.05.02) to edit.
History - 0.4	- LLMCMSS.02.02.01.6 clearly marked as provision, no additional development - LLMCMSS.02.02.01.4: add re-set of inactivity timeout by long-running activities - LLMCMSS.02.02.05.03: shall be able to transfer control, but not always or automatically

LLMCMSS.02.01 General

1. The LLMLUI shall be based on the LLF - LLM LUI Framework
2. Access to any function shall be controlled by the rights a user has by their credentials

LLMCMSS.02.02 Usage Scenarios

LLMCMSS.02.02.01 System Start / Shutown /Session housekeeping

1. When switching on the LLM local terminal, the LLMLUI login screen shall be presented
2. It shall be kept in mind to enable a future enhancement: When switching off the LLM local terminal, the user's session shall be locked and cancelled (timeout logout) after a configurable amount of minutes. Additionally, switching off or loss of power of the local terminal shall be reported as an alarm condition.
3. Inactivity/Logout timeout shall be configurable as a system item overridable by a user item
4. Inactivity timeout shall be made variable depending on the active screen(s)/window(s)
5. An application shall be able to "re-arm" the inactivity timeout (in case a longer activity does not require user LUI activity)

6. Sessions shall be saveable and recall-able, also from other local terminals in the same LLM system. However, no additional functionality shall be developed, only the method of handling sessions shall allow this "transportability"
7. Sessions shall have unique IDs identifying one sesion uniquely in activiity logs and for audit purposes

LLMCMSS.02.02.02 Logging in

1. Login shall grant user access to the system and authenticate them by various credentials:
    1. enter username and password onscreen
    2. provide ID by contactless smart card (smarcard reader required on local terminal)
    3. provide ID from Wii-Controller or similar non-standard device by opened library API
2. ~~It shall be possible to log into one of more LLM system instances~~
3. ~~Alternatively, a user having rights in more than one LLM system instance may be automatically logged into all of them~~
4. ~~Login screen shall be able to be pre-populated when provided with data from ILC or a trainer/care-person~~
5. ILC or other application shall be able to initiate logout upon movement detection or patient change situations. Provisions shall be made as in LLMCMSS.02.01.02.01 above, but no real implementation, as automatic logout has still to be defined not to do more harm than being useful.

LLMCMSS.02.02.03 User Administration

As of LLMCMSS Issue 0.8 and newer, there is no administration task to be handled using the LUI. All administration is to be handled via RUI. Therefore, the requirements in this section are moved to LLMCMSS.04.05.01, leaving this section here only as a forward pointer.

1. ~~LLMLUI shall allow to create a user account, provided creating user has appropriate rights~~
    1. ~~new user account shall be pre-filled from a template, template shall be selectable at creation time~~
2. ~~LLMLUI shall allow user account deletion, provided user has appropriate rights~~
3. ~~user accounts shall have arbitrary number of attributes, LLMLUI shall take account for this fact:~~
    1. ~~UserCategory: Senior, User~~
    2. ~~UserRole: Relative, Therapist, Administrator, Unknown~~
    3. ~~All attributes assigned by LLMDB web service interface~~
    4. ~~Additional Attributes as seen fit to fulfil additional features:~~
        1. ~~Multi-Tenancy~~
        2. ~~Sub-Grouping~~
        3. ~~Privacy Attributes~~
        4. ~~Judisdiction imposed properties~~
4. ~~unique user ID shall be constant across database versions and shall allow~~
    1. ~~moving a user across database instances~~
    2. ~~exporting / importing single or multiple users~~
    3. ~~merging and splitting of LLMDB instances during re-organization~~
    4. ~~manipulation of users on database level without breaking validity on web service level~~

LLMCMSS.02.02.04 Invoking xTC and other subsystems, switching LUIs

1. LLMLUI and all subsystem LUIs shall have a set of controls to switch to another subsystem (for this part, LLMLUI is assumed to be just another subsystem LUI to be switched to, which will hopefully prove as the intuitives way also for the user).

2. If the subsystem to be switched to is not already started, it shall be started when switching to it, creating a look-and-feel of "always there" which may be more convenient for the user perception.
3. each subsystem shall have its own window. Usually, only one subsystem (the one called to be "in front") will be visible at a given time. However, on larger screens more than one subsystem LUI will be visible with possible overlap, but with a clear notion of one being "in front". Thgis shall be implemented by means of LLF, LLF using the appropriate OS library functions available in all target OSs for the local terminal platform.
4. subsystem LUI processes shall be able to request to be put "in front " in case of important information to be displayed.
5. Apart from being "in front", LUI windows shall be able to request attention by drawing on their windows anyway (also if not "in front", screen display may find another way) and by generating audible output, which shall be possible also for not-"in front" LUI windows. Audible prompts may be a convenient way anyway for appointments or schedules without any need to put the generating window it front if the sound messages contains the appointment or schedule information (text-to-speech).

LLMCMSS.02.02.05 Scheduling and appointments

1. The LLM and CMS LUI shall allow to view scheduled appointments on request by providing a "calendar" button on each menu screen.
2. The scheduler-LUI shall show a day/month/week/list view with appointments marked.
3. If an appointment is due, the schedule-LUI shall be able to transfer control to the LUI the appointment is for:
    1. CTC and PTC for a scheduled training session, passing training schedule information to the respective xTC to automatically start with exercises and instructions
    2. ILC with additional information for invoking phone or video conversations for trainer's or doctor's appointments or to start conversations with relatives on their birthdays
    3. simply displaying a message or playing an audible or video announcment for general reminders
4. Editing schedules and appointments shall be done with the CMS and CMS RUI by therapist, relative, trainer etc..
5. Schedules shall be stored in the LLMDB associated to their "owner" user account
6. Schedules shall be able to trigger workflows of activities to be executed in sequence. Such chains shall be implemented as chains of scheduled acticvities.
7. To enable chains of activities, there shall be a number of special "trigger" conditions apart from a certain one-time or repeating interval:
    1. At logon. Most probably linked to a specific local terminal or group of terminals. Also shall be qualifyable by the logon method, i.e. using an ID card or one of a set of ID cards.
    2. After completion of another activity, with optional time delay.

LLMCMSS.02.02.06 OPTIONAL: Remote Control and Remote Assistance

This section is optional and shall be implemented if provided by underlying technologies as the windows terminal server or remote access functionalities. Further added functionalities must be re-evaluated during operation of LLM system V 1 to assess more detail support requirements.

1. A support person shall be able to share user's local terminal view or at least a set of windows and their state.
2. The support person shall be able to assist the user and perform actions on the LUI on their behalf in parallel with a audio or video conversation.

3. This remote support scenario shall help in implementing use cases like trainer-assisted PTC and CTC usage.
4. In a usage scenario of workflow and remote control the support-person's local terminal shall have a pop-up with the user's local terminal address and shall be able to connect to the local terminal by clicking on a control in the pop-up.
5. Seeing the user's local terminal window, the support person shall also be enabled to know the cause for support by seeing the appointment pop-up.
6. Alternatively, provisions shall be developed to have cross-terminal and cross-session workflows

LLMCMSS.03 - LLM CMS LUI functional specification

This part of LLMCMSS provides the functional specification for the local user interface (LUI) for the LLM system central management system (CMS).

LLMCMSS.03.00 Document Control

LLMCMSS.03 Document properties	
Item	**Value**
Document	LLMCMSS.03
Issue	0.4
History - 0.1	Created due to Reference Architecture V 0.1
History - 0.2	- Updated cross-references to LLF and LRF in Reference Architecture V 0.3 - Details on LUI functions
History - 0.3	- correct numbering in LLMCMSS.03.01.xx - LLMCMSS.03.02.03 is obsolete due to obsolence of multiple LLM instances - LLMCMSS.03.01.02 applies only for picture code - LLMCMSS.03.01.04 only lists schedules on LUI, editing only at CMS RUI - LLMCMSS.03.01.07 and .08 are OPTIONAL as of LLM V 1 - LLMCMSS.03.09.02 is OPTIONAL as of LLM V 1 - LLMCMSS.03.09.03 is obsolete due to the callback-event-driven notifications
History - 0.4	- LLMCMSS.03.01.9: is now optional, not mandatory for V 1

LLMCMSS.03.01 CMS LUI functions

1. The CMS LUI shall provide all functionality to be performed by the consumer of the LLM service, also referred to as the **senior** .
2. Change Authentication secrets: picture code
    1. This optional function shall only be available if the user is entitled to it by their rights
    2. As LUI does not allow for arbitrary text entry, the change function applies only to the picture code and only if a user has configured one.
    3. icture code change shall require the old code and twice the new one
    4. This option shall NOT be available if authentication is done by alternative method like smart-card or sound recognition
3. Display xTC and other (medical, etc) available performance data and feedback information, prepared by the LRF, detailed in LLMCMSS.08
    1. Display shall be initiated either by notification or deliberate user selection
    2. User selection shall include selecting only reports applicable for the user. Applicability shall be performed by matching user report selection properties against report properties. For LLM V 1.0 there shall be a collection of properties in the user profile that shall be exactly equal to the same subset of report properties.
    3. Notification selected report display shall provide a hyperlink (button) on the notification page invoking exactly one report for display
4. Display ~~and Edit~~ xTC and other schedules (repeating) and appointments (one time). Editing is done on CMS RUI
5. Access to the functions shall be controlled by the security credentials of the user's session
6. Apart from the UI-functions listed above the user may be presented with other display pages/windows initiated by workflows, schedules and notifications

7. OPTIONAL: After login, a special "Login-workflow" shall be triggered in the CMS service component that may provide additional notification-triggered activity-prompts
8. OPTIONAL: Before logout, a "logout-workflow" shall be triggered by a "logout" event in the CMS service component
9. OPTIONAL: Activity timeouts shall be able to supervise the users activity or in-activity and shall be able to trigger corresponding events in the CMS service component to generate notifications to the user or also to other users or sessions:
    1. timeout responding to notifications
    2. OPTIONAL timeout performing no activity for a certain interval
    3. ~~periodic timeout as long as LUI session exists to check for available information or schedules~~

LLMCMSS.03.02 CMS LUI Implementation requirements

1. CMS LUI shall be based on the LLF, detailed in LLMCMSS.05
2. CMS LUI shall use the CMS service component, detailed in LLMCMSS.04 to perform each activity
3. ~~CMS LUI shall allow for multiple sessions by multiple users to multiple LLMDB instances on one local terminal~~
4. The LLF platform and environmental specifications also apply to the CMS LUI

LLMCMSS.04 - CMS service component functional specification

This part of LLMCMSS specifies the functionality and behaviour of the LLM CMS service software.

LLMCMSS.04.00 Document Control

LLMCMSS.04 document properties	
Item	**Content**
Document(Part)	LLMCMSS.04
Issue	0.9
History - 0.1	Created
History - 0.2	
History - 0.3	
History - 0.4	
History - 0.5	- create user accounts from templates in config - move LLM-DB interface to LLMCMSS.09 - precised user account management - precised session management - some details on credentials - some details on workflows/schedules/notifications/event processing
History - 0.6	- added LLMCMSS.04.02.14 for generic data management
History - 0.7	- corrected issue in header - added Overview chapter - added hint for possible obsolesence/optionality of service API functionality, markups ***MANDATORY*** and ***OPTION_CHECK***
History - 0.8	- added LLMCMSS.04.05 for explicit RUI functionality specification: - LLMCMSS.04.05.01 for user administration - LLMCMSS.04.05.02 for schedule administration - LLMCMSS.04.05.03 for GOE, general object editing - LLMCMSS.04.05.04 for GPE , generic property editing - LLMCMSS.04.05.05 for Training Result Viewer
History - 0.9	- clarify picture Code uniqueness and other conditions in 4.5.1.3.5 and 4.5.1.5

LLMCMSS.04.02 Overview

1. The CMS service is the core of the central management of the LLM system. It has no user interface, but provides a web service API for all of its functionality. The LLM CMS LUI and other higher-level functionality use this API to provide a user frontend to the management functions and to perfrom management and adminitrative actions.
2. The CMS component provides also a Web-based RUI (Remote User Interface) including, aside from the functions provided locally by the CMS LUI, all enhanced administrations activity necessary for LLM CMS configuration management.
3. This part of LLMCMSS specifies API functions to be accessible via web-service.

4. All API methods shall be usable also as locally invoked functions. However, documentation shall state where side-effects are to be expected because of the then following multi-instantiation or security issues of underlying objects. As an example, sessions may be created locally and per web-service. Sessions are held persistent in the LLM-DB. A local applicaiton may not have access to the LLM-DB, only CMS service shall have it. Therefore, session management MUST be called remotely.
   On the other hand, there may be idempotent helper objects whose manageing methods may be invoked also locally.
5. IMPORTANT: (new in issue 06) There are a number of functions specified in this part. Not all othem may be required by the higher-level applications. Their existence has been expected at some time in the specification process. However, the implementation may take other paths as foreseen by the specification and may render some functions no longer necessary. In order to avoid redundance, higher level applications should be designed together with the top-level-design of the CMS service in order to identify obsolete functions.
   Functions that are already known to be possibly not required, at least not in Version 1 of the LLM system, are marked up with ***OPTION_CHECK***. This means that before implementing this function, the higher level applications should be examined for the requirement.
   NOTE however, that a few functions in this part are marked as ***MANDATORY*** and are expected to be required anyway, either by near future enhancements or by plug-ins in either LRF , LLF, PTC or CTC integrations. Therefore, these shall be implemented anyway.
6. The CMS service component exposes 2 interfaces: a web-service API and a web user interface. The former is specificd in LLMCMSS.04.02, the latter, also titled "CMS RUI", is specified in LLMCMSS.04.05.

LLMCMSS.04.02 CMS API functions

1. Not all API functions need to be available as CMS RUI functions, CMS RUI functions are explicitely named in LLMCMSS.04.05
2. Every API function shall be protected by access security and shall be logged including security credential information and terminal name(if applicable)
    1. User credentials shall be kept in the session and associated with a terminal.
    2. credentials shall be valid only with the terminal kept in the session. Therefore, each API call needs to have a terminal ID with it. API calls not from real terminals must provide valid virtual terminal IDs.
    3. Sessions may be transferred from one terminal to another by a specific "change terminal" API call. This API call shall not be available to all users, only to special ones.
3. There shall be a generic strcuture editor API to perform low-level viewing and changing of LLM-DB PROPERTIES structure (for generic editing, the API specified in LLMCMSS.04.02.14 shall be employed):
    1. Session shall keep "edit point" in the PROPERTIES structure
    2. navigation commands/methods to go up/down/previous(n)/next(n)/list(n) in the PROPERTIES tree
    3. commands/methods to add/change/delete current item
    4. commands/methods to export/import from download/upload XML file
4. User logon
5. Session management
    1. Create Session, associate with terminal (local/remote) and user account
    2. Lock/Unlock Session
    3. Dump / Restore Session
    4. Terminate / Shutdown Session
    5. Manage Session variables/properties (create/query/delete/associate)

6. change terminal

6. user account management
    1. A user account may have arbitrary properties in hierarchical manner associated with it.
        1. Managing secret values shall avoid sending cleartext values across any network (even LAN)
        2. Especially, managing picture code loigin tokens shall be in conformance with LLMCMSS.06.02.01
    2. Create Account
        1. There shall be a template for each user account created
        2. The template shall have default property values copied into the newly created account
        3. If there is no template name given, a default template, the name of which is configured in the property "/sysconfig/sysdefaults/default_user_template_name", shall be used. The hardcoded default for this shall be that no template shall be used.
        4. If there is no template applicable, the empty template will be used, this means, no properties will be assigned to the user account.
    3. Lock/Unlock Account
    4. Export / Import Account
7. Preferences management
    1. Screen readability
    2. Audio levels
    3. Language Settings
    4. skinnig
    5. avatar hooks and avatar state
        1. preparation for integration of Audio TTS notification reading
        2. preparation for integration of alternative forms entry methods:
            1. provide user credentials / authorisation information via:
                1. smartcard
                2. sound recognition
            2. provide dialogue box (yes/no/...) responses via:
                1. sound recognition
8. Equipment Management
    1. PTC Equipement and Settings
    2. CTC Equipment and Settings
    3. export / import settings
9. Statistics data management
    1. query logging/statistics data
    2. filter
    3. export / import
    4. delete
    5. space management
10. schedule/appointment processing
    1. create scheduled activities/appointments with repeat/interval/calendar characteristics
    2. Execute the scheduled activity or generate notifications when schedules are due
    3. optional / for later versions: pre-announcements and repeat announcements if not confirmed
11. workflow processing
    1. CMS component shall be able to manage workflows conformant to WMFC standards, preferrably XPDL

There are a number of open source XPDL processors available, shall be checked.
Events and Agents and Activities shall be mapped to the respective LLM object types

12. notification management
    1. CMS shall keep information about the state of users and sessions
    2. any application shall be able to issue a notification request for a user or a specific session of a user via CMS SOAP API
       A notification shall be a displayable entity and shall have properties structuring the data and giving hints about presentation of the data
    3. CMS shall display the notification in the appropriate window and requested way plus bringing the window to the front if requested
    4. CMS shall keep track of confirmation of a notification (press a button when notification is read) if requested and perform follow-up activity in case of no confirmation from the user
    5. Notification shall be able also to take alternative forms additionally to local terminal display: send email, forward events to other applications and/or workflows
13. general functions for schedule/workflow/notification management
    1. General event API
       There shall be set of methods to inject events into the CMS subsystem.
       Events shall have arbitrary properties that can be used as mapping/routing criteria or to carry information up to displayable text or processing instructions
    2. Event mapping
       Events shall be able to be mapped to activities inside the CMS component configurably depending on property values
    3. Timing facility
       There shall be a function to execute activities based on absolute or relative time/date values or intervals. This facility shall be able to interact with the calendar/schedule management to perform activities and generate notifications based on schedule/calendar entries.
14. ***MANDATORY (needed by LLMCMSS.06.05.02)*** There shall be an umbrella API for the use of applications and the configuration editor use case in LLMCMSS.06.05.02. The same mechanisms for structure navigation and manipulation as in LLMCMSS.04.02.03.
    1. Session shall keep "edit point" in the PROPERTIES structure
    2. navigation commands/methods to go up/down/previous(n)/next(n)/list(n) in the PROPERTIES tree
    3. commands/methods to add/change/delete current item
    4. commands/methods to export/import from download/upload XML file; handling parameters as in LLMCMSS.06.05.02 and in conformance with screen definitions in LLMCMSS.06.06.06
    5. The virtual unified "tree" shall look like this (sub-structures of the elements at root level except for "PROPERTIES" shall be as specified in the LLM-WS definition):
        1. root "/"
            1. "MEMBERS"
            2. "USERS"
            3. "SENIORS"
            4. "USER_SENIOR_RELATIONS"
            5. "CTC"
                1. "COMPONENTS" - CTC components
                2. "ACTIVITIES" - CTC activity
                3. "SENIOR_ACTIVITIES" - CTC senior's activity
            6. "PTC"
                1. "COMPONENTS"
                2. "ACTIVITIES"

3. "SENIOR_ACTIVITIES"
7. "ILC"
    1. "COMPONENTS"
    2. "ACTIVITIES"
    3. "SENIOR_ACTIVTIES"
8. "PROPERTIES"
    1. "CLASSES" - tree templates for standard objects, with classes and types for object attributes
    2. "TYPES" - type definitions of tree leaves, including subtyping and ranges
    3. "SESSIONS" - session persistent structure

6. The mapping (as described above) of disparate structures into the virtual tree shall not be hard-coded. There shall be a struture in the PROPERTIES subtree defining the tree. The employed pattern shall be that of a Unix file system tree with mounted file systems. Each subtree (="mounted filesystem") may have its own implementing dynamically loaded class library.
7. ***OPTION_CHECK (implement only if the need by applications arises)*** The above mapping structure shall also allow for "links" as already specified in LLMCMSS.09.03.01.06, but this time explicitely allowing to access one and the same structure by different names (=points in the information tree)

LLMCMSS.04.03 CMS function interfaces

The CMS function block has 4 interfaces:

1. Web-Service interface provided to be used by CMS LUI or other CMS function consumers
2. Web-Service client to LLM-DB
3. local Filesystem for local configuration, basic logging/auditing and data caching
4. web-application environment for basic bootstrap configuration

LLMCMSS.04.03.01 CMS function provided web service interface

The CMS function block shall expose a web service interface providing all the functions listed in LLMCMSS.04.01.

LLMCMSS.04.03.02 LLM-DB functions needed by CMS

All data structures and functions required by the whole CMS subsystem are kept together in LLMCMSS.09

LLMCMSS.04.04 CMS implementation issues

1. CMS shall be implemented as a web-application providing SOAP web services running in a JEE container
2. CMS shall be implemented stateless, only using internal structures for buffering, persistent storage shall be kept in LLMDB, most probably in the sessions context.
3. Instantiation and life cycle management
4. domain scope
    1. one instance of CMS shall be associated with one administrative domain, i.e. one instance of LLMDB. However, multiple database connections shall be possible to address redundancy issues

LLMCMSS.04.05 CMS RUI functionality

The functions mentioned in this section shal be offered by the CMS on the web-based RUI interface:

LLMCMSS.04.05.01 user administration

As of LLMCMSS Issue 0.8 and newer, there is no administration task to be handled using the LUI. All administration is to be handled via RUI. Therefore, the requirements in this section are moved from LLMCMSS.02 to here.

1. CMS RUI shall allow to create a user account, provided creating user has appropriate rights
    1. new user account shall be pre-filled from a template, template shall be selectable at creation time
2. CMS RUI shall allow user account deletion, provided user has appropriate rights.
    1. Real Deletion shall be effected only in certain cases. In order to avoid data referencing "into the void", user accounts shall be marked "deleted", but not deleted really if avoidable.
3. user accounts shall have arbitrary number of attributes, CMS RUI shall take account for this fact:
    1. UserCategory: Senior, User
    2. UserRole: Relative, Therapist, Administrator, Unknown
    3. All attributes assigned by LLMDB web service interface
    4. Additional Attributes as seen fit to fulfil additional features:
        1. Multi-Tenancy
        2. Sub-Grouping
        3. Privacy Attributes
        4. Judisdiction imposed properties
        5. Managing picture code loigin tokens shall be in conformance with LLMCMSS.06.02.01
4. unique user ID shall be constant across database versions and shall allow
    1. moving a user across database instances
    2. exporting / importing single or multiple users
    3. merging and splitting of LLMDB instances during re-organization
    4. manipulation of users on database level without breaking validity on web service level
5. The user administration shall be only a configuration of the general object editor (GOE) specified in LLMCMSS.04.05.03. General functionality shall be re-used as much as possible. If there seems to be special functionality required by the user administration, it shall be investigated if it can be generalized and to be put in the general object editor. One special functionality for user administration is the management of the picture code. It shall be managed inside the GOE with a specific Add-On to handle picture codes like specified in LLMCMSS.06.02.01.

LLMCMSS.04.05.02 schedule administration

1. CMS RUI shall allow creation, modification and deletion of scheduled events for any user
2. schedules shall be administered according to functionality specified in LLMCMSS.04.02.10
3. The schedule editor shall be only a configuration of the general object editor specified in LLMCMSS.04.05.03. General functionality shall be re-used as much as possible. If there seems to be special functionality required by the schedule editor, it shall be investigated if it can be generalized and to be put in the general object editor.

LLMCMSS.04.05.03 GOE - general object editor

1. Every Administration or data management web user interface shall be derived from a general object editor. The common functionality of each admin function shall be kept in a common framework in order to implement it only once and then re-use it. Moreover, this also allows for future enhancements without having to develop an extra administration or editing software component for future additional configuration or other structured data.
2. The general object editor, also abbreviated GOE, shall allow creation, modification, inspection, deletion, export and import of any structured data in the LLM database.
3. The GOE shall rely on the API specified in LLMCMSS.04.02.14 for uniform tree-based access to the LLM-DB
4. Metadata shall be used to define:
    1. what properties (memeber variables) an object of a specific class shall have
    2. the attributes of the properties of an objects
    3. user entry, validation and editing methods for specific property data types
    4. at what point in the LLM-DB-tree which types/classes of objects are allowed
5. Metadata shall also be stored on the LLM-DB tree
6. protection mechanisms shall be obeyed to determine if the user using the GOE is allowed to perform the desired object editing operation. If the standard LLM-DB security control mechanism is not sufficient to determine the specific access control. the metadata shall augment the security definitions to a greated level of detail. One such enhancement may be that a specific role of user may only add a property at a certain location of the LLM-DB-tree or that another role of a user only may set a specific property to a certain value range.

LLMCMSS.04.05.04 - Generic property editor

1. The CMS RUI shall provide a generic access to the LLM-DB as viewed by the LLM-DB tree API specified in LLMCMSS.04.02.14.
2. This function shall be abbreviated as GPE - the Generic Property Editor
3. GPE shall be a low-level access to each single property as stored in the LLM-DB. No interpretation shall be done, all properties shall be displayed and editable in their primitive data types with the full value range allowed for these:
    1. NUMBER - decimal text representation of the number
    2. STRING - display the string with escapes for non-prinbtable characters
    3. PROPERTIES - display a hyperlink leading to another list displaying the contents of the respective tree branch level
4. GPE shall display one level of the tree at one time.
5. Each property shall be displayed as 3 text fields:
    1. The name
    2. The Type: NUMBER, STRING and PROPERTIES
    3. The value
6. All fields shall be editable text fields
7. Each property row shall have a "submit changes" button and a "delete" button, performing the respective action directly, if allowed by security settungs in the database.
8. If the number of entries shown at one time exceeds a configurable limit, the listing shall be split up in pages. The RUI shall provide controls to navigate between the pages
9. each page shall have an empty row if input fields to add a new property. It shall have an "add" button to commit the new row to the LLM-DB
10. GPE shall have a "select file" form to specify a file name to import from or to export to
11. GPE shall have an "import" button. Pressing this button shall effect in importing property values from the specified file in XML form
12. GPE shall have an "export" button. Pressing this button shall effect in exporting the current property tree into the specified file in XML form

13. The "import" and "export" functionality shall support:
    1. moving a subtree inside the whole tree
    2. backing up of a subtree or a complete LLM-DB
    3. restoring a subtree or a complete LLM-DB
14. After submitting changes (delete, submit changes, import, export, add) the current page shall be re-displayed with possible effect caused by re-arranging the subtree. If applicable, the first page shall be displayed in spite of the current page (after deletion or importing)
15. GPE RUI screens shall be taken from LLMCMSS.06.06.06

LLMCMSS.04.05.05 - Training Result Viewer

1. CMS RUI shall provide a web-based viewer for training results intended for the audience class the current RUI user belongs to
2. The training results shall be assumed to have been prepared by the LRF in the same manner as the feedback for the senior.
3. Training results for audience classes like therapist or relative may contain more details than the results for the senior.
4. Training results displayed shall be able to have additional auxiliary data related to the training results. Which data to select shall be configurable but need to be retrievable based on a training result record like:
    1. senior name
    2. training session time
    3. additional senior properties
    4. training session details retrievable from the raw training result records
5. The RUI training result viewer shall allow selection of training results by criteria like:
    1. senior name or ID
    2. training type
    3. time range
    4. target audience class
6. Allowed values for selection criteria, target audience class and presented auxiliary data shall be restrictable by the security credentials of the RUI user:
    1. A relative for one senior may be allowed only to view results for this senior and only for the relative target audience class.
    2. A therapist may be allowed only to view results for the seniors assigned to her/him in the "attending doctor" relation
    3. A training programme developer may be able to view all training result for a specific programme but not the names or other personal data of the respective seniors
7. RUI Training result view screen shall have 2 panes:
    1. entry form for selection criteria
    2. display form for report HTML output and auxiliary data
8. If the selection criteria result in more than one report record to be displayed, the display pane shall have navigation control to browse amongst the result.
9. In order to use screen real estate as efficient as possible, the display pane shall be a scrolling area potentially capable of displaying more than one report in a scrollable area
10. The user shall be able to choose amongst a number of configurable display pane templates to select the displayed data items and to be able to flexibly choose amongst list/page/... views
11. display and selection pane contents shall be printable of exportable in CSV/PDF, detailed output depending on the applicable representation types. generated exports shall be able to be sent via Email

LLMCMSS.05 - LLM Local User Interface Framework (LLF) Specification

This part of the LLM CMS Specification specifies the framework components on which all local user interface components - LUI - shall be built upon.

As the LLF is considered as an external component to the CMS subsystem, it can be considered as if had an "ENVREQ:" in front of it

LLMCMSS.05.00 Document Control

LLMCMSS.05 document properties	
Item	**Value**
Document	LLMCMSS.05
Issue	0.5
History - 0.1	Created
History - 0.2	Added Callback Web Service Interface
History - 0.3	added document control
History - 0.4	added data-dependent controls and control sets
History - 0.5	updated according to findings in meeting @ ISTU on 2010-02-09: - only one window and always on top plus monopolizing display and input devices - event paradigma: LLF appearances not to be controlled by application, LLF in charge - detailed difference between "uncooperatice" and "cooperative" applications - precise alternative input methods handling with webn service event API

LLMCMSS.05.01 LLF platform

1. The LLF shall operate on a local terminal platform
2. The local terminal shall have following attributes:
    1. Hardware platform(s):
        1. ASUS nettop as in eHome installations
        2. Desktop WinXP/Win7 PC specified for CTC software
    2. Operating System Environments: Windows variants:
        1. Win XP 32 bit
        2. Win 7 32 bit

LLMCMSS.05.02 LLF Functionality

1. Basic Operation Mode: The LLF shall present the user a dialogue with a collection of controls, displayed in a window. Controls may have varying graphical design and functionality.
   As of LLM V 1, only the following controls shall be used:

    1. a "button": arbitrary 2D-shape with action at touch, with properties like active/inactive, (in)visible and configurable changing pictures, all controllable by actions performed in the LLF instance
    2. an "HTML-rectagle": a rectangualr area where HTML-code is rendered into. Apart from directly passing HTML document, an HTTP-URL shall be able to be passed and the control then itself GET-ting the HTML document.
2. Navigation: There shall be activities, triggered by events or control interaction, to lead to another dialogue with a different set of controls. The new dialogue shall be able to be displayed in the same window like the invoking dialogue, replacing the invoking dialogue
3. A control shall be able to perform arbitrary functionality, especially performing actions on instances of Web-Service-Interfaces. Possibilities may be limited by the fact that a control is executed in the context of a specific platform. The activity carried out may be intended to affect a software component running on a different physical or logical platform or a seperated security realm. In all such cases, the LLM architecture calls for interaction via a web service interface.
4. There shall be a set of controls capable of "wrapping" already existing software applications running on the same platform. Depending of the ability of such applications, there may be different types of interactions and therefore different control implementations to interact with such an application:
    1. "Uncooperative Application": Such an application cannot be controlled in any way by the LLF control except for invoking it with an operating system call. The control shall at least monitor the existance of the invoked process and be able to modally block the LLF window during execution of the application or allow parallel execution.
        1. LLF shall release its monopolizing grip on display Z-axis (LLF normally always on top) and input as well as pointing devices
        2. However, a set of safety and stability measures must be implemented:
            1. A timeout for modal or quasi-modal application windows to "escape" in case of a locked up application. Such application windows would prevent other activities while executing. If such an application does not terminate properly, all LUI would be locked up. Such a case needs to be avoided under all circumstances.
            2. process supervision for the application to detect creashed application processes
            3. measures to communicate the application status back to LLF to update the state of controls and other state-induced data
    2. "Cooperative Application": Such an application at least supports bringing another LLF window to the front of the display when the latter requests it due to an alarm, notification or similar. This requires amending the application, but only to a certain extent:
        1. The application needs to provide UI controls to send events to the LLM application system, e.g. for an "emergency/distress" button
        2. LLF shall be able to push itself back to fron in certain cases, thus obstructing the applications window and again monopolizing the input devices.
    3. "Integrated application": such an application can be run inside a client window created by the LLF instance and therefore fully blended with the LLF look-and-feel. A model for such a mechanism would be the ILC LUI as its based on the predecessor of the LLF and therefore can be deployed into an LLF instance.
5. Apart from the interaction on the GUI level, other integration needs to be taken care of:
    1. User credentials and authentication. The invoked application needs to run within the security allowances of the logged-in user on the LLF level. If such a "token switch" is not possible, the LLF instance shall at least run under a accoiunt without elevated privileges and carry real user authentication information and credentials as parameters on the application level.

    2. Data storage. The invoked application shall be enabled to place generated data in the LLM-DB via the WS-API
    3. Configuration. Like data storage, the invoked application needs to be enabled to take configuration information from the LLM-DB via the WS-API.
    4. The Data storage and Configuration Issues can be resolved in the worst case by "brute force wrappers": Such a wrapper would wrap around the application to be invoked and, at the start, retrieve the configuration and input data from the LLM-DB using session information and credentials. It would generate the input data in a format suitable for the invoked application (e.g. config files) and then invoke tha application. After completion of the application, the wrapper would collect all output (e.g. from statistics files etc.) and stuff it into the LLM-DB. This looks like a very crude process but could be the last resort with very non-cooperative application the LLM-project has no control over.
6. Call-Back Functionality: The LLF shall allow for an external application to control certain behaviour of the local terminal display:
    1. LLF shall offer a SOAP-API for each LLF instance on a local terminal
    2. API shall allow to inject events into the LLF instance, actions for handling the event shall be to the sole discretion of the LLF configuration
7. Data Dependent-Controls: There shall be controls whose appearance shall be dependent on data values, i.e. on the result of a library call:
    1. Any arbitrary control or control group present or absent based on library call result
    2. button active or inactive ("greyed") dependent on call result
    3. number of controls in a control set dependent on cardinality of a function return set
    4. Button image or text dependent on the return value of a library call, e.g. image file name returned by library call, e.g. as a property from LLM-DB
8. There shall be a well-separated way to employ different input devices, such as security tokens, smartcards and input devices not mapping to mouse and keyboard abstractions. Such devices shall have their own integrations and shall propagate application-specific events to the LLF-instances, much like in the callback functionality above. Examples are:
    1. login via smart card: card reader in scanned by integration application. Inserting, removing or swiping a card may result in generating a "login_request(credentials)" or "logout_request()" event to the LLF which may handle it or even ignore it in states noit appropriate of loggin in or out.
    2. dialogue responses via WiiMote-gestures: the integration software may decode the gestures (like "yes", "no", "up", "down" or even "emergency") and forward them as events with the respective meaning to the LLF-instance. If there is a dialogue with responses required like yes and no, the yes and no events will be accepted, otherwise ignored.

LLMLCMSS.05.03 - LLF Implementation and Modularity

1. LLF Implementation Code and Application Implementation Code shall be separately developed and maintainable. A system of dynamically loadable libraries (DLLs) along with a data-driven configuration data set shall be employed to achieve configuration flexibility, expandability and separation.
2. A sensible application of an MVC-(Model/View/Controller)-Pattern shall be followed. However, it is a definitive NON-GOAL of LLF to greatly re-design the basis software framework taken over from eHome, but all measures available shall be taken to carefully separate appearance from content from function flow.
3. Types of controls and other visual elements shall be able to be packaged separately and deployed independently. Multiple packages shall be able to be deployed into the same LLF instance and be distinguishable by a prefix/namespace-paradigm.

4. LLF shall allow for separated 3rd-party code libraries to be used by control implementations. One example for such a library would be a web-service client-library to access the LLM-DB or to access other LLM functional components
5. The variant of implementing LLF as a web-browser based frontend shall be kept in mind and re-evaluated at every new release of LLF. Existing CTC implementations already have web-frontends. Component-based web-application frameworks like Apache Wicket provide a large number of GUI controls providing native-like look-and feel by freeing LLF design from tasks and sotware components that are already existing in the market. Moreover, implementations like Apache Wicket are Open Source and free software.

LLMCMSS.05.04 - LLF Configuration Management

1. There shall be multiple LLF configurations per HW/OS-platform instance. These may be named "configuration" or "instance". One instance shall be configured by one set of configuration data and shall use one specific version of software components. Instances shall be able to be functionally and security-wise isolated from each other. However, only one LLF instance may be active at the same time as LLF monopolizes the Z-priority and input devices. Multi-display configurations like citrix or terminal server or virtualization shall be definitely NON-GOAL for an LLM local terminal.

LLMCMSS.06 - LLM CMS Use Cases

This part of LLMCMSS specifies LLMCMS functionalities required by use cases as far they are concerning the LLMCMS.

The use cases are taken from LLMTOS.05 - LLM Operational Specifications.

The use cases described in this part of the specification are the mandatory set of use cases the LLM system shall be able to perform. This part may therefore also serve as the blueprint for the functional part of the system acceptance test.

LLMCMSS.06.00 - Document Control

LLMCMSS.06 document properties	
Item	**Value**
document	LLMCMSS.06
Issue	0.7
History - 0.1	Created topics based on LLMTOS
History - 0.2	added document control
History - 0.3	Details for use cases, models and scenarios
History - 0.4	- added screen layout pictures - adaption to changed login token handling - add idle screen display - add use cases for admin and management
History - 0.5	- formatting corrections - clarifications concerning separation of responsibilities between LUI and CMS/LRF - RUI screen layout pictures - pending: hyperlinks to screen definitions not yet completely inserted (cosmetic) - add LUI schedule editing screens
History - 0.6	- add LOGOUT Button to LUI Main Menu Screen #02 - LUI schedule editor is only OPTIONAL, just for study
History - 0.7	- add LLMCMSS.06.02.01.03 for picture code uniqueness - remove LUI schedule editor including "+" button in LLMCMSS.06.04.11 - add general design hints in LLMCMSS.06.04 in front - point out non-goal of automatisms in training program selection in LLMCMSS.06.01.01.3> - correct editing use case in LLMCMSS.06.05.02 - add navigation use cases to LMCMSS.06.06.06 and LLMCMSS.06.05.02 - change Import to seperate dialogue (LLMCMSS.06.06.06.02) in LLMCMSS.06.06.06 - join LLMCMS.06.01.02 and LLMCMSS.06.01.03

LLMCMSS.06.01 - LLM Service Scenarios - LLMTOS.05.01

The scenarios described in this section are not detailed in their steps and sequence. The functionality required by each requirements item in this section is either covered by other part(s) of the LLM system or further detailed in section LLMCMSS.06.02.

LLMCMSS.06.01.01 - LLMTOS.05.01.01 - LLM for elderly remaining at home

1. Home safety monitoring to be performed by ILC
2. Integrated (video) telephony to be performed by ILC
3. Opposed to LLMTOS.05.01.01.01, training schedules are assumed to be created by a human user in the role therapist.
   However, provisions shall be foreseen to enable the following for a post-V1 development (nothing implemented in V 1):
   Therapist pre-creates a set of different level training programs.
   "applicability" mechanism to use previous results as parameter for the selection of the appropriate training program.
   To achieve this, the handling of the "training complete" event shall deposit a "training level" property in the seniors profile in the LLM-DB. The therapist then can e.g. create 2 or more "physical training - treadmill" programs tagged with various "training level" properties themselves. So, when the senior selects the "treadmill" program, only the program with the appropriate currrent training level will be selected.

LLMCMSS.06.01.02 - LLMTOS.05.01.02 - LLM in a day-care center

1. ENVREQ: ILC shall have a "speed dial" functionality, accessable by a button from the senior main menu screen, to call the currently assigned personal trainer.
2. A danger or distress alert either generated by ILC or by the "alert/help/unwell" button pressed by the senior shall also call the currently assigned personal trainer.
3. ENVREQ: PTC shall produce notifications to be displayed by CMS at LUI to drink water etc.
4. There shall be arbitrary notifications on the LUI to be confirmed by the logged-in senior. If confirmation is not given after a certain time, an alert situation shall be entered. At alert, any active PTC device shall be frozen/disegaged/powered off. Unfreeze shall be possible by pressing an "OK" button in the displayed notification window.
5. Additionally, all requirements of LLMCMSS.06.01.03 apply here also as the clinical institution from the viewpoint of LLM is related to daycare centers

LLMCMSS.06.01.03 - LLMTOS.05.01.03 - LLM in a clinical institution for elderly

1. There shall be a simple means to assign a senior to an ILC environment. This can help monitoring a person in a hospital room and generate appropriate alerts. This association shall be stored in the LLM-DB by building a reference form an ILC environment to a senior record.
2. Additionally, all requirements of LLMCMSS.06.01.02 apply here also as the clinical institution from the viewpoint of LLM is related to daycare centers

LLMCMSS.06.02 - LLM Use Case Scenarios - LLMTOS.05.02

LLMCMSS.06.02.01 - Login to LLM system

Environment: Local Terminal

Start state: 1.0

State 1.0: Local Terminal logged out, local terminal software running, idle screen #0 displayed
Event 1.0.1: user requests manual login by pressing "manual login" button
Activity 1.0.1: manual login screen #1 displayed, -> state 1.1
Event 1.0.2: user presents login token (by placing smart card on, in or in proximity to reader, depending on technology), token reader software issues "login" event to LLF instance
Activity 1.0.2: login method with token id called in CMS, CMS verifies credentials, creates session and associates with local terminal instance, LLF displays senior main menu screen

LLMCMSS.06.02.01.03 - Picture code uniqueness: Picture code is both secret authentication and identification. Therefore ist must be encrypted (hashed like Unix password) before transmitted onto the network or stored. Additionally, it must be unique across one LLM system instance to uniquely identify exactly one user. This means it must be checked for being unique when created or changed, otherwise rejected.

State 1.1: manual login screen #1 displayed
event 1.1.1: user enters valid picture code, presses "submit"
Action 1.1.1: login method with picture code called in CMS, CMS verifies credentials, creates session and associates with local terminal instance, LLF displays senior main menu screen
event 1.1.2: user enters invalid picture code, presses "submit"
Activity 1.1.2: login method with picture code called in CMS, CMS returns error message, LLF display error message on login screen, -> state 1.1
event 1.1.3: user cancels manual login by pressing "back" button
Activity 1.1.3: idle screen displayed, -> state 1.0
event 1.1.4: user presses "help" button
Activity 1.1.4: display help screen with help for manual login display and procedure

LLMCMSS.06.02.02 - Senior performing traning session

Start State: 2.0

State 2.0: Session logged in, LLM main menu displayed
Event 2.0.1: user presses "cognitive training" and there is only 1 active cognitive training program
Action 2.0.1: LUI queries CMS for CTC programs and start CTC-LUI with the only active training program ID as parameter
Event 2.0.2: user presses "cognitive training" and there is more than 1 active cognitive training program
Action 2.0.2: LUI queries CMS for CTC programs and display cognitive training program selection screen #6, -> to State 2.3
Event 2.0.3: user presses "physical training" and there is only 1 active physical training program
Action 2.0.3: notify PTC and display PTC active screen, -> state 2.4
Event 2.0.4: user presses "physical training" and there is more than 1 active physical training program
Action 2.0.4: display physical training program selection screen #6, -> state 2.3
Event 2.0.5: schedule notification for training
Action 2.0.5: pop up schedule notification screen #05, -> state 2.1

State 2.1: Session logged in, LLM main menu displayed, notification for training program reminder popped up
Event 2.1.1: user presses "start training" button and scheduled training type is physical
Action 2.1.1: notify PTC and display PTC active screen, -> state 2.4

Event 2.1.2: user presses "start training" and scheduled training is cognitive
Action 2.1.2: start CTC wrapper LUI with scheduled training program, -> state 2.5

State 2.2: cognitive training selection screen displayed
Event 2.2.1: user selects one cognitive training
Action 2.2.1: start CTC wrapper LUI with selected training program, -> state 2.5

State 2.3: physical training selection screen displayed
Event 2.3.1: user selects one physical training
Action 2.3.1: notify PTC and display PTC active screen, -> state 2.4

State 2.4: PTC training active, PTC active screen displayed
Event 2.4.1: training completion event from PTC
Action 2.4.1: display training complete and result available hint in PTC active screen, -> State 2.6

State 2.5: CTC training active, CTC controls front window
Event 2.5.1: CTC training complete (event notification from LRF upon termination of feedback calculation)
Action 2.5.1: display training result screen from CTC feedback event, -> State 2.7
Event 2.5.2: user presses "distress" control on CTC user interface
Action 2.5.2: LLF re-acquires screen control, displays distress confirmation screen, -> state 3.5

State 2.6: PTC active screen with training complete hint displayed
Event 2.6.1: user presses "OK" button for training hint
Action 2.6.1: display training result screen from PTC complete event, -> state 2.7

State 2.7: training result screen displayed
Event 2.7.1: user presses "OK button
Action 2.7.1: dismiss training result screen, bring LLM main menu to front, -> END

LLMCMSS.06.02.03 - Alarm Setting

Environment: local terminal

Variant: A: CMS controls PTC device freeze for accidents, <= PREFERRABLE

Variant: B: ILC controls PTC device freeze on accident, CMS performs dialogue with user for recovery

Variant: C: ILC controls PTC device freeze and performs dialogue with user for recovery

Start State: 3.1 or 3.2 (for variant A) or 3.3 (for variant B) or 3.4 (for variant C) or 3.8 (for any non-training related distress)

State 3.1: (equivalent state 2.5) senior performing CTC training session
Event 3.1.1: accident notification event from ILC on CMS API with local terminal ID
Action 3.1.1: CMS: notify CTC for freezing training session, notify LLF-instance about distress situation (via event), LUI: re-acquire screen control and display recovery response dialogue, -> state 3.5

State 3.2: (equivalent state 2.4) senior performing PTC training session
Event 3.2.1: accident notification event from ILC on CMS API with local terminal ID
Action 3.2.1: CMS: find PTC device associated with local terminal (from LLM-DB) and notify PTC of

emergency-freeze for respective PTC device, notify LLF-instance about distress situation (via event), LUI: display recovery response dialogue, -> state 3.6

State 3.3: (equivalent state 2.4) senior performing PTC training session
Event 3.3.1: accident notification event from ILC on CMS API with PTC device ID, ILC already has frozen PTC device
Action 3.3.1: display recovery response dialogue, -> state 3.7

State 3.4: senior performing PTC training session (ILC to handle distress situation alone)
No events for this flow, ILC handles it all on its own.
Drawback 1: recovery management out of control of CMS
Drawback 2: ILC will have to know that user performs PTC training
Drawback 3: CMS will have to care for an accident during CTC training anyway, => 2 accident flows!

State 3.5: recovery response dialogue displayed (during CMS-frozen CTC session)
Event 3.5.1: User responds "no help needed"
Action 3.5.1: notify CTC to unfreeze, dismiss dialogue, flow ends, ->state 3.1

State 3.6: recovery response dalogue displayed (during CMS-frozen PTC session)
Event 3.6.1: User responds "no help needed"
Action 3.6.1: LUI notifies CMS of cancelled distress, CMS notify PTC to unfreeze, flow ends, -> state 3.2

State 3.7: recovery response dialogue displayed (during ILC-frozen PTC session)
Event 3.7.1: User responds "no help needed"
Action 3.7.1: LUI notify ILC to unfreeze PTC session, flow ends, -> state 3.3
Event 3.9.2: user presses "Yes, need help" button or response dialogue times out (down counter reaches 0)
Action 3.9.2: LUI changes to ILC VoIP call screen and automatically calls assigned support person's number and informs CMS of confirmed distress situation

ENVREQ: (for variant A) PTC service component (RefArch "PTC") must support event API to receive notifications and/or emergency events

ENVREQ: (for variant B) ILC service component must support event API to receive notifications

ENVREQ: (for variant C) ILC service component must handle accident including recovery response dialogue

State 3.8: any state where there is a "need help/emergeny/distress" button shown on the LUI screen but not the PTC active states already mentioned above
Event 3.8.1: user presses distress button
Action 3.8.1: display recovery response dialogue, -> state 3.9

State 3.9: recovery response dialogue displayed (arbitrary other LUI state)
On-Entry-of 3.9: query CMS for contact (tel.number) of currently assigned support person, memorize it and start downcounter for "/sysconfig/recover_response_timeout" seconds
Event 3.9.1: user presses "don't need help" button
Action 3.9.1: LUI returns to previous state in history, notify CMS of cancelled distress event (for statistics and logging)
Event 3.9.2: user presses "Yes, need help" button or response dialogue times out (down counter reaches

0)
Action 3.9.2: LUI changes to ILC VoIP call screen and automatically calls assigned support person's number

LLMCMSS.06.02.04 - Check senior's progress

Environment: remote terminal, user is either relative or therapist

Start State: 4.1

State 4.1: therapist or relative main menu screen displayed
Event 4.1.1: user presses "view training results"
Action 4.1.1: select all applicable (according to user role relative or therapist) training results for selected senior and display it in a new window in a training result screen, -> state 4.2

State 4.2: training result screen displayed
Event 4.2.1: user presses "previous" or "next" (buttons available only if there is more than 1 result available
Action 4.2.1: display previous/next result in result set in training result screen, -> state 2
Event 4.2.2: user presses "OK"
Action 4.2.2: close training result screen window and bring therapist/relative main window to the front, -> end of flow

LLMCMSS.06.03 - LLM Use Case Model - LLMTOS.05.03

This section maps the subsystems of the use case model laid out in LLMTOS.05.03 to functionalities in the LLM-system as far as the scope of this specification (LLMCMSS) is concerned.

Similar to LLMCMSS.06.01, not all the scenarios described in this section are not detailed in their steps and sequence. The functionality required by each requirements item in this section is

- either covered by other part(s) of the LLM system or
- further detailed in section LLMCMSS.06.02.
- If neither of the above covers the use case in detail, it is done here.
- Items not being specific use cases of their own formulate a (functional or architectural) requirement for a component defined elsewhere. An example for this would be the requirement of classifying training programs to determine their applicability for certain groups of seniors.

LLMCMSS.06.03.01 Use Case Subsystem "Training Program"

1. The "senior" role shall have a "Perform Training" LUI main menu item
2. Training programs shall be of different main types: physical, cognitive, ...

LLMCMSS.06.03.01.03 - Training program creation

1. Training programs shall be createable and assignable by the therapist role
2. Training programs shall be stored in the PROPERTIES section of the LLM-DB under "/sysconfig/training-programs"
3. The "therapist" role shall have CREATE access to the tree of training programs

LLMCMSS.06.03.02 Use Case Subsystem "Daily Schedule"

1. There shall be scheduleable tasks of type "training program", defined by the training program and personal trainee parameters
2. Scheduled training programs shall notify the trainee and allow for easy switching into the training program, see LLMCMSS.06.02.02

LLMCMSS.06.03.03 Use Case Subsystem "Alerts"

1. Alerts about the condition of the senior shall be handled by ILC
2. Additionally, Danger Alerts shall be integrated into CTC and PTC activity according to LLMCMSS.06.02.03
3. Additionally, Distress Alerts shall be enabled by equipping each display screen with an "Alert/Help/Unwell" Button to directly generate an alert into CMS and (optionally) into ILC

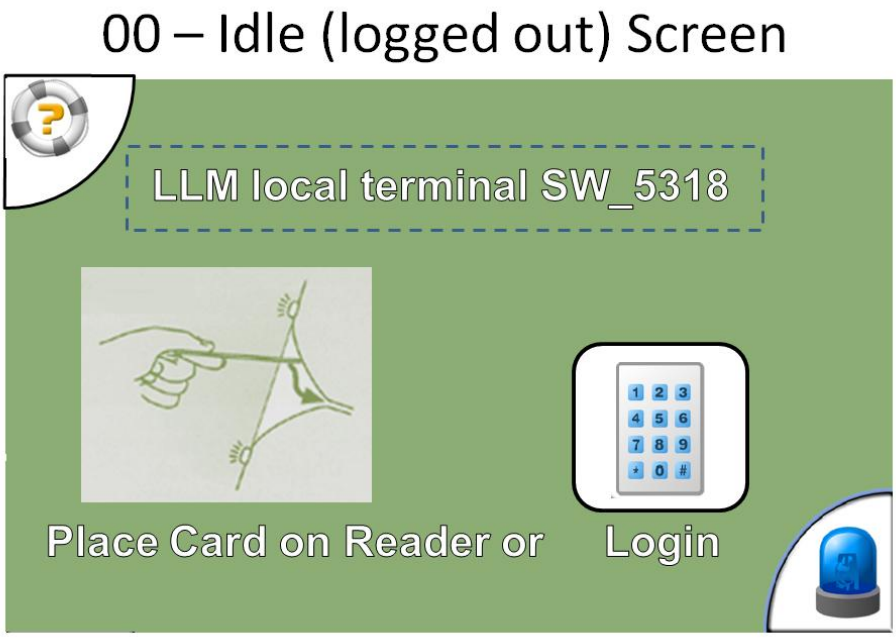
LLMCMSS.06.03.04 Use Case Subsystem "Reporting"

1. Monitoring seniors progress by relatives and therapists shall be handled according to LLMCMSS.06.02.04

LLMCMSS.06.04 - LUI Screens

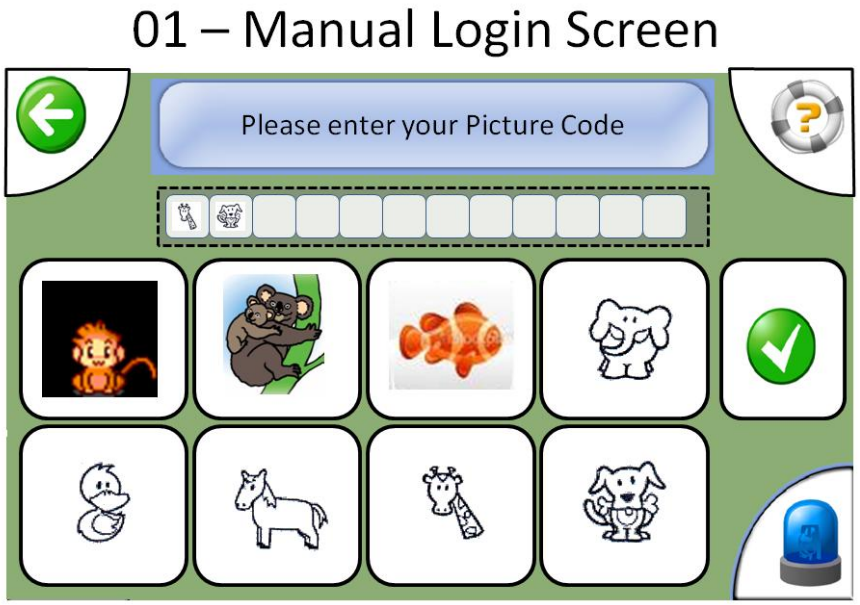
1. The screens shown in this section are intended to be only functional samples, not to be taken as-is in the design and layout way. Common LLM design guide and eHome usability principles shall be applied to the design and layout.
2. All buttons on the LUI screen shall have text captions in addition to their icons.

LLMCMSS.06.04.00 - Idle Screen



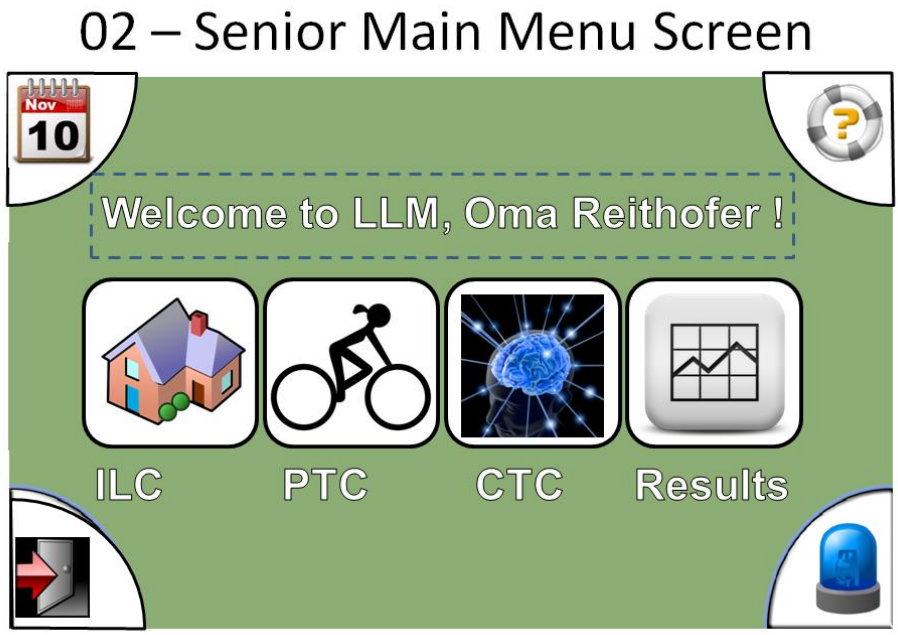
- Title: System Configurable local terminal identification
- Picture + Text Label: "present login token / smart card (if configured!)
- Button: "manual login"
- Button: "Help"
- Button: "Distress"

LLMCMSS.06.04.01 - Manual Login screen



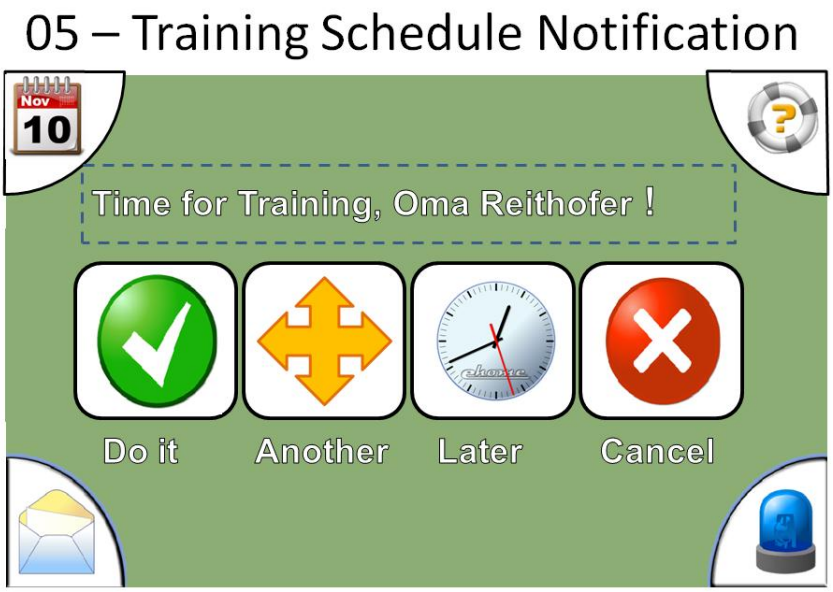
- Title: (Enter Picture Code) System-Configurable title
- 8 Buttons for 8 available picture glyphs
- display area to show already entered picture code part
- Button: "Enter/Submit"
- Button: "Back"

LLMCMSS.06.04.02 - LLM senior main menu screen



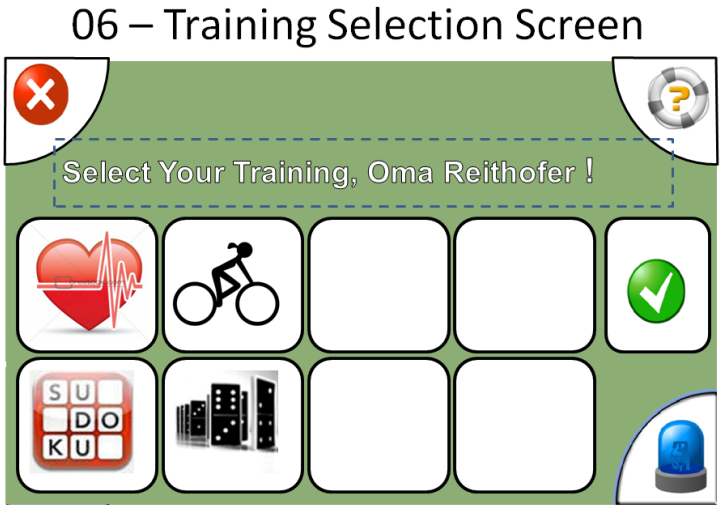
- Title: User-Configurable main menu banner (text/logo/graphics)
- Button: "Manage Schedules"
- Button: "View Results"
- Button: "Physical Training" - if user has rights
- Button: "Cognitive Training" - if user has rights
- Button: "Switch to ILC" - if user has rights
- Button: "log out"
- Button: "Emergency/Distress"
- Button: "Help"

LLMCMSS.06.04.05 - training program reminder notification



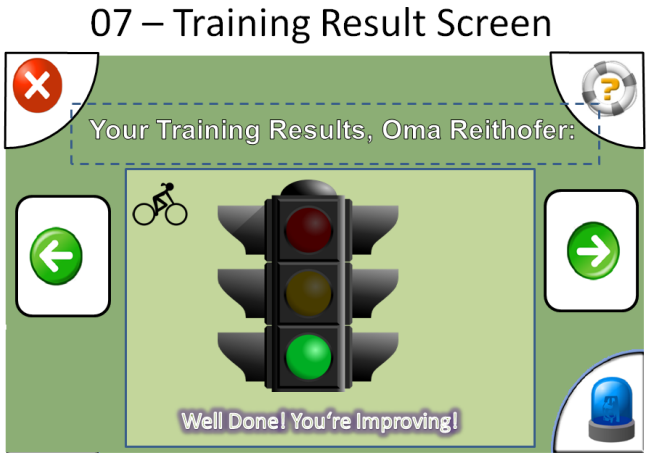
- Title: "it's time for your training" or other configrable text/logo/graphics + scheduled time
- Text: name of senior
- Text: type of training (cognitive/physical)
- Text: name of training program
- Button: "start training"
- Button: "start another training" - if senior has another active traning program of this type
- Button: "remind me later"
- Button: "Cancel"

LLMCMSS.06.04.06 - physical / cognitive training selection screen



- Title: "Select yout training program" or other configrable text/logo/graphics
- Text: name of senior
- Button set:
  for each of the active training programs for the senior
  one button with text or picture determined by the
  training type (physical/cognitive) class(cardio/fat-burn/... or sudoku/memorize/...); maximum of 2 rows with 4 buttons each,
  to not exceed window real estate, have reasonable size and produce no overload
- Button: "Cancel"

LLMCMSS.06.04.07 - training result screen

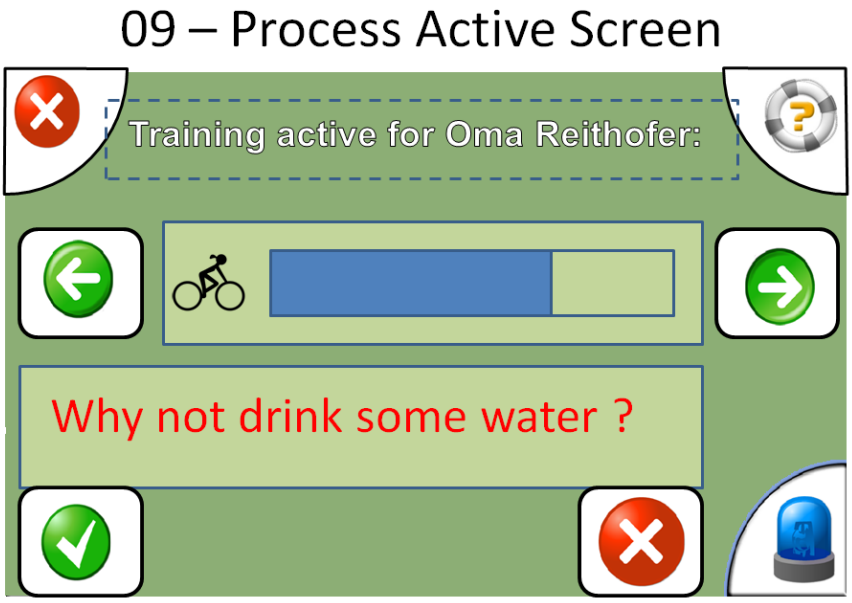


- Title: "Training results ..."
- Text: name of senior
- display frame (HTML): feedback data from last event
- button "previous" - active if more results available matching the request
- button "next" - active if more results available matching the request
- Button: "OK"

LLMCMSS.06.04.08 - recovery response dialogue

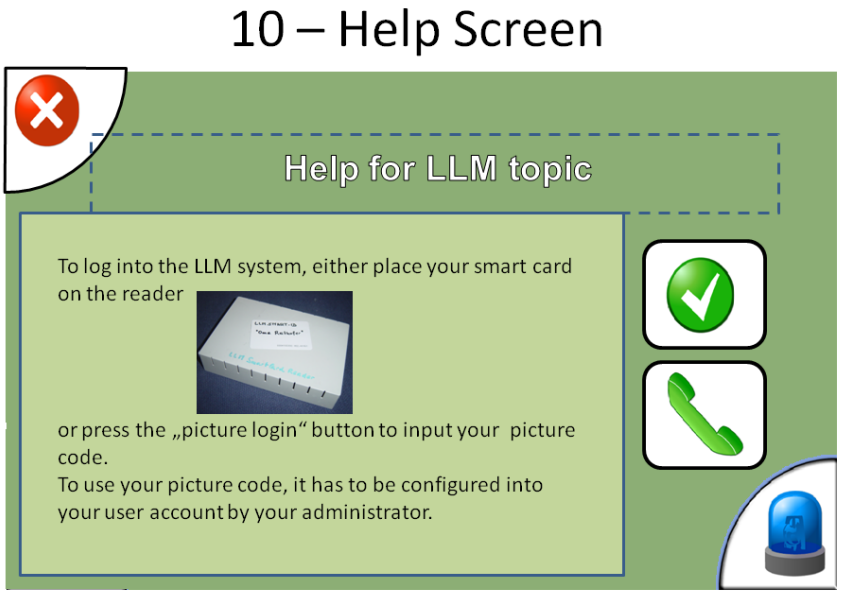
- Title: "do you need help?"
- Text: name of senior, description of local terminal site (with associated PTC)
- Button: "I'm OK"
- countdown display: seconds until response timeout
- Button: "I'm unwell"
- Behaviour:
  - opening dialogue starts response timeout (configurable RECOVERRESPONSETIMEOUT), timeout generates timeout event to CMS
  - Button "OK" or "unwell" stops timeout and dismisses dialogue
  - countdown display updates every second and show seconds until timeout

LLMCMSS.06.04.09 - process active display screen



- Title: "Training Active" (shall be configurable to be re-useable for other activity status displays)
- HTML status display area with 2 buttons: "previous status message", "next status message"
- HTML instruction display area with 2 buttons: "OK / did it", "not understood / could not do it / did not do it"

LLMCMSS.06.04.10 - help screen



- Title: "Help for ... " (Configurable per help topic)
- HTML help display area
- button: "OK"
- button: "voice/video call for more help"

LLMCMSS.06.04.11 - schedule list

- Title: Schedules for <username"
- list with 4 components to each scheduled event:
  - time and repetition
  - event type icon
  - event details
  - comment or remark
- if list longer than 4 entries:
  - NEXT button to scroll down
  - PREVIOUS button to scroll up

LLMCMSS.06.05 - administration and management use cases

LLMCMSS.06.05.01 - Logging in at RUI

Environment: RUI

Start State: 501.1

State 501.1: home URL of LLM RUI openend, welcome screen displayed
Event 501.1.1: user enters valid username and password
Action 501.1.1: RUI processes input, CMS verifies credentials, creates session, associates with web session, copies user rights into session, RUI creates main menu items, display main menu, -> State 502.1
Event 501.1.2: user enters invalid credentials in login form, submits form
Action 501.1.2: RUI processes input, CMS verifies credentials, rejects with error message, RUI displays error message, -> State 501.1

LLMCMSS.06.05.02 Configuration / Property object editing

This use case with associated dialogues shall be used for access to all manageable and configurable parts of the LLM-DB. Specific structures (senior/member/user/...) shall be mapped as part of a virtually connected "tree" that can be traversed and edited with this dialogue. The API to provide this virtual view is specifid in LLMCMSS.04.02.14.

If the respective controls are available in the employed web application component framework, the configuration editor shall as close as possible resemble the windows registry editor or the respective parts of the windows file explorer to allow for intuitive handling by the administrators.

Environment: RUI

Start State: 502.1

State 502.1: RUI main menu displayed
Event 502.1: user selects menu item "property object editor"
Action 502.1: initialize current context to "/", current page-offset to 0, enter state 502.2

State 502.2: list of properties displayed
On_Entry_Of_State 502.2: display "N" properties of the current context at the current page-offset, 1 row per property with 3 columns (name, type, value), plus 1 empty row for adding a new property, all fields editable text fields, each row with "submit change" and "delete" button, new-property row with "add" button
Event 502.2.1: user presses "Next" or "previous" button
Action 502.2.1: set page-offset to + or - "N", re-enter state 502.2
Event 502.2.2: user enters new values in a row and presses adjacent "submit change" button
Action 502.2.2: RUI writes back changed property, re-enter state 502.2
Event 502.2.3: user enters new values in empty "add" row and presses "add" button
Action 502.2.3: RUI adds new property, re-enters state 502.2
Event 502.2.4: user presses "delete" button next to a property entry
Action 502.2.4: display delete confirmation dialogue, -> state 502.4
Event 502.2.5: user presses "EXPORT" button of form
Action: 502.2.5:RUI generates XML file output document, browser pops up "save/open" dialogue, stay in state 502.2
Action: 502.2.6: user presses "IMPORT" button of form
Action 502.2.6: RUI pops up Import dialogue LLMCMSS.06.06.06.02, execute import, refresh actual displayed page, but revert to page 1, back to state 502.2
Event 502.7: user enters new path and presses "GO TO" button of context path area
Action 502.7: accept new path, re-enter state 502.2
Event 502.8: user presses "one level upward" button of context path area
Action 502.7: go back one level upwards in path, re-enter state 502.2

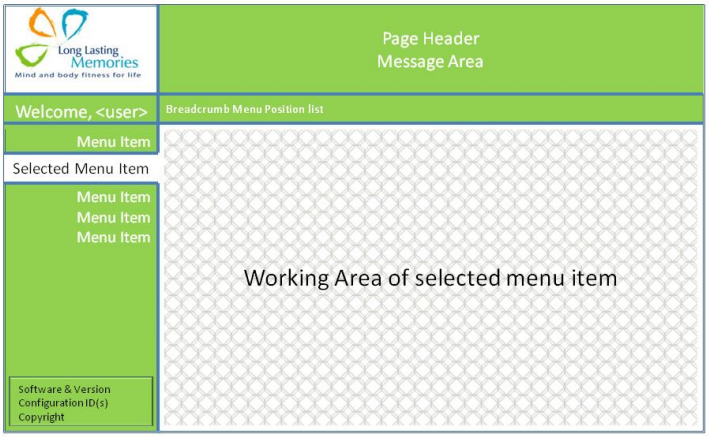
State 502.4: delete confirmation dialogue displayed
Event 502.4.1: user presses "OK"
Action 502.4.1: RUI calls "Delete_Property()" with CMS, CMS with LLM-DB, refresh list view, -> state 502.2
Event 502.4.2: user presses "Cancel"
Action 502.4.2: return to property list view, -> State 502.2

LLMCMSS.06.06 - RUI screens

LLMCMSS.06.06.00 - General LLM RUI screen layout

1. There shall be a constant layout of all RUI screens and dialogues according to picture 06.06.00.

06.06.00 - LLM RUI Screen layout

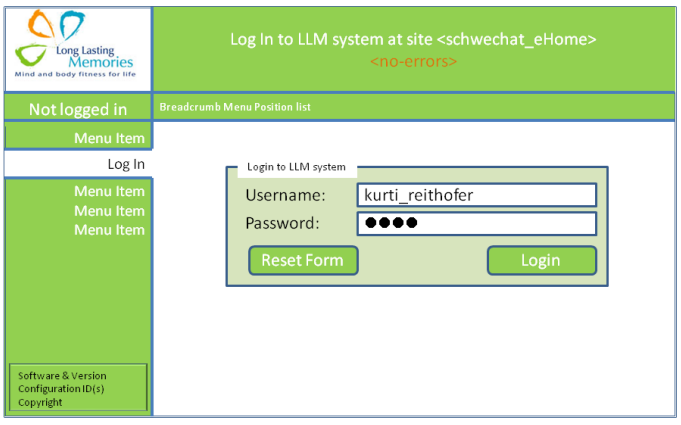


1. All Attributes of the following screen definitions apply to the working area, except for:
    1. Title - shown in "Page header"
    2. navig-pos: shown in "breadcrumb menu position list"
    3. Message - shown in "Message Area"

1. Whenever a menu is entered, there shall be a default entry selected
2. Whenever a menu item is selected (also if it is selected by default when the menu is entered) its working area shall already be displayed.
3. Depending on the nature of a menu item, the default displayed work area at selection of the menu item shall already show an initial data set, e.g. the last training result of the currently selected senior.

LLMCMSS.06.06.01 - Idle/Welcome/Login screen

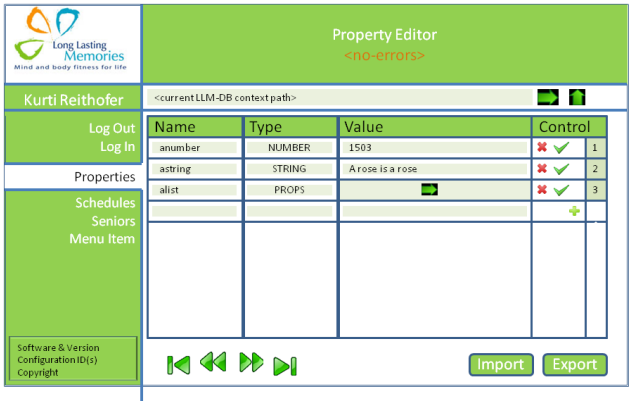
06.06.01 - LLM RUI Login Screen



- **Title:** Configurable: Welcome to LLM instance ...
- Form: login to LLM RUI
    - Username: text entry with lab
    - Password: text entry with labe input echo (password field)
    - Reset Button "Reset Form"
    - Submit Button: "Login to LL
    - **Message:** status message, init empty
- Disclaimer(s) and Privacy Caveats: configurable HTML paragraph
- Advertisments and Sponsor messages configurable HTML paragraph

LLMCMSS.06.06.06 - configuration object / property list display

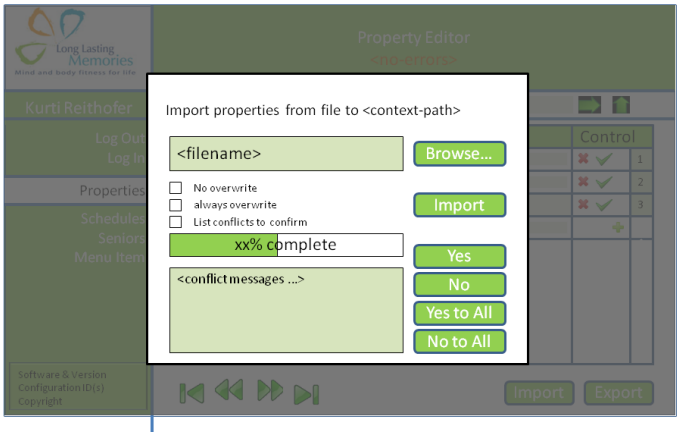
06.06.06 -RUI Property List



- Title: "Object List <path-or-identifier>" editable
- Button "go to" - to accept changed path and go to there
- "Page <n> of <k>", "n" bein editable
- Button "Go to page" - to change to entered page
- Button "Up" to go up in hierarchy, if applicable
- Table with headers:
  - ordinal number in current result set
  - for object list:
    - unique ID of object
    - value of 1st object attribute
    - value of 2nd object attribute
  - for single object:
    - name of object attribute
    - type of object attribute
    - value of object attribute
  - for property list:
    - name of property
    - type of property
    - value of property
  - if applicable: DELETE button
  - if applicable: SUBMIT CHANGES button
- button: "FIRST page"
- button: "NEXT page"
- button: "PREVIOUS page"
- button: "LAST page"
- button: "IMPORT"
- button: "EXPORT"

LLMCMSS.06.06.06.02 - Import property pop-up dialogue

06.06.06.02 -RUI Import Dialogue



- title including path where properties are to be imported
- file name selection for import file including BROWSE button
- radio button set for handling of already existing property names:
  - overwrite always
  - overwrite never generate list of import conflicts
- button: "IMPORT"
- progress bar
- message field for issues to be confirmed
- response for conflict issues:
  - button YES
  - button YES to ALL
  - button NO
  - button NO to ALL

LLMCMSS.07 - LLM CMS Aspects: Configuration, Security, Modularity, Performance, Volumetrics

This section of LLMCMSS specifies all non-functional aspects of the LLMCMS subsystem:

- Modularity
- Implementation issues
- Deployment Architecture
- Configuration
- Security
- Performance
- Volumetrics
- Safety and Runtime Environment

LLMCMSS.07.00 - Document Control

LLMCMSS.07 document properties	
Item	**Value**
document	LLMCMSS.07
Issue	0.8
History - 0.1	Created, list of aspects
History - 0.2	added document control
History - 0.3	added volumetrics
History - 0.4	added modularity (tbc), performance, security, configuration
History - 0.5	- re-structured subsection 7.2.1 into 7.2.1,7.2.2,7.2.3, renumbered following subsections - added implementation sequence - added DB-replication - added Implementation / deployment architecture
History - 0.6	- added table of contents - added module listing and responsibilities in 7.2.2.3
History - 0.7	- added reusability to 7.2.1, plus special example 7.2.1.2 in xTC exchangeable parts
History - 0.8	- clarify in LLMCMSS.07.02.07.1 that all combinations have to be supported

LLMCMSS.07.02 - Aspects Details

LLMCMSS.07.02.01 - Modularity

LLMCMSS.07.02.01.01 - General terms and concepts

1. The LLM system shall consist of small installable entities called ***Packages.*** A package is the smallest installable entity.
2. A number of packages can be installed and work together in an ***Instance.***
3. One or more Instances are created on one ***Node;*** a Node is one hardware box (1 PC, 1 server) or one partition in a multi-partition platform or one virtual box.

LLMCMSS.07.02.01.02 - modularity for reusability - special example: PTC and CTC integration

In an environment with so many variable parameters when it comes to system and device configurations as well as use cases and work flows, special attention shall be laid on building system components that can be re-used in other similar places.

To illustrate this requirement and underlying concept, one example which may well become reality shall be worked out here:

The common concepts in the integration of different sorts of PTC and CTC devices and software:

- As of LLMSYS V 1, the CTC is a software running on the local terminal and needing to be integrated into the display management done by the LLF.
- The PTC, on the other hand is assumed to be a "headless" device software with no own user interface. Therefore, the LLM system design uses a "dummy" activity display using a "process active screen" to show the approximate status of the PTC training session to the user.
- This results in the CTC integration and PTC integration being "blueprints" for two types of external systems being integrated into the LLM system.
- If now, what seems possible, the PTC gets its own user interface software, most probably running on the local terminal too, no further development should be necessary except for configuring the PTC integration , the respective screens and the PTC-related workflows and evtent processing, because the same model now necessary for the PTC integration has already been developed for the local terminsl based CTC integration

The above scenario shall be kept in mind in system design to make such a re-use of components feasible. The detailed system design shall be tested specifically for such use cases.

LLMCMSS.07.02.02 - Implementation Issues

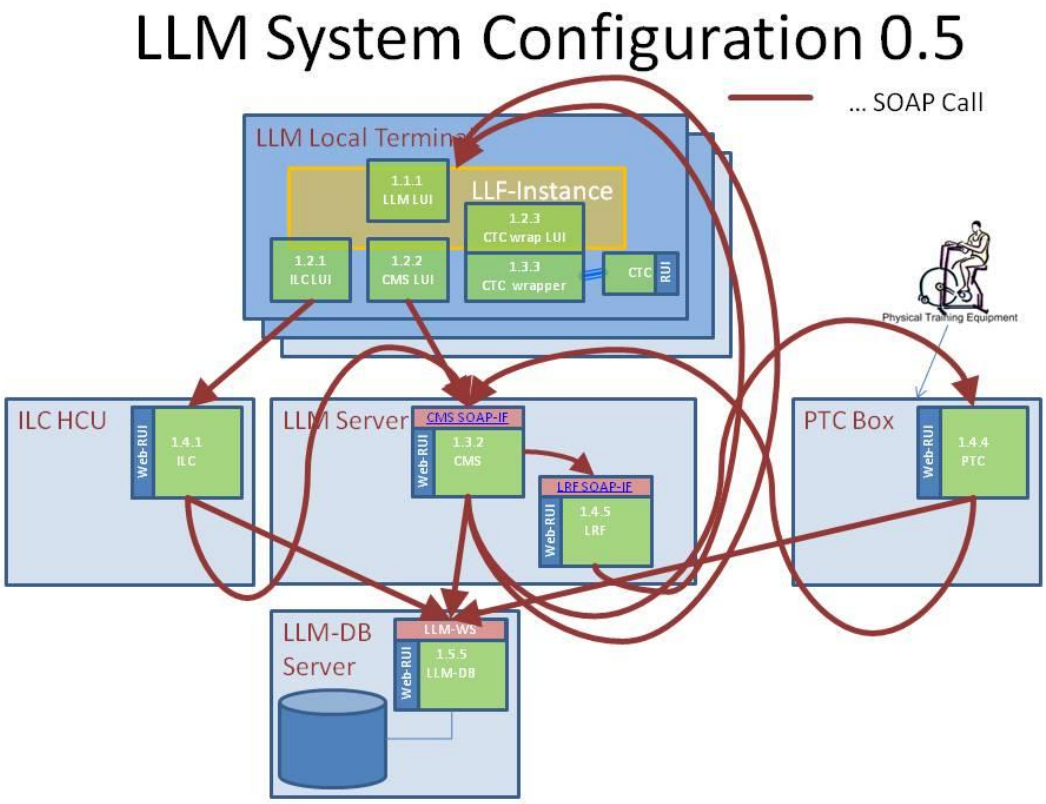
1. Implementation shall be broken down in modular work packages as devised in Modularity and Deployment Archictecture subsections.
2. Implementation shall follow a rapid develop-test-refine iterative cycle in order to achieve early experiences with structures, user interface and configurability.
3. As of the current modularity structure and project status,there are the following modules to be developed in the whole LLM-system:
   Note: italic bold printed items are considered part of the LLM CMS activities
   Note: to the right of each item, responsible teams are noted: AUTH, ISTU, CEIT
   1. ***LLF*** - ISTU
   2. LLM-DB - AUTH
   3. CTC integration - AUTH
   4. CTC LUI integration - ISTU
   5. PTC integration - AUTH
   6. ILC integration - CEIT
   7. ***RUI Framework + Application Server*** - CEIT
   8. ***LLM-LUI screens*** - ISTU
   9. ***CMS-LUI screens*** - ISTU
   10. ***Workflow LUI screens*** - ISTU
   11. ***Flow specific CMS service code*** - CEIT
   12. ***Specific RUI object editors*** - CEIT
   13. ***LRF*** - CEIT
   14. LRF plugins with specific training result processing algorithm code - AUTH with domain experts
   15. ***non-editor CMS RUI code*** - CEIT

16. ***generic CMS RUI DB-object editor*** - CEIT
17. DB local replication - AUTH
18. DB global replication (if different from local) - AUTH
19. LUI and LRF localization concept - ISTU
20. LUI und LRF localization and local texts - local project partners

4. As of the current modularity structure, the impolementation sequence could be as follows:
    1. first iteration: LLF development, LLM-DB development, in parallel with CTC integration and PTC integration, basic RUI functions (AppServer)
    2. second iteration: Version 1 screens and workflows, screen/flow-specific code libraries, specific object-editors, LRF development, DB local replication
    3. third iteration: LRF algorithms, RUI completion (all modules), generic DB-object-editor, DB global replication, RUI consolidation per-box

LLMCMSS.07.02.03 - Deployment Architecture

1. Deployment of LLM shall be on a combination of up to 3 node-types:
    1. local **terminal node**
    2. local/regional **server node**
    3. local/regional/global **database node**
2. Assignment of the functional modules to the node types shall be as shown in the "System Configuration" picture below:



LLMCMSS.07.02.04 - Configuration

This subsection deals with the way the LLM-System shall be configured reliably and efficiently.

1. For each LLM system ***Instance***, there shall be a central instance of LLM-DB containing all node/package specific configuration. The address of the Web-Service providing the LLM-DB interface may be configured in an instance-wide configuration file or Web-Application-Serer configuration data set. It also can be held by an web service discovery repository. In this case it has to be determined how the repository is found in the network and where the repository is installed an running.
2. All configuration data has to be held in the LLM-DB instance associated with the respective local terminal instance or server instance.

LLMCMSS.07.02.05 - Security

1. All accesses to LLM-system components must be protected by LLM-managed credentials
2. Platform hardening measures shall be applied to prevent circumvention of LLM-system internal security.
3. No LLM-system component shall require system admin privileges for installation or execution
4. Network ports used for access to web-services shall be clearly documented in system documentation and firewall measures be taken to prevent access to any non-authorized port.

LLMCMSS.07.02.06 - Performance

1. All performance figures are to be taken with respect to the volumetrics limits. If volumetrics limits are exceeded, performance may not be guaranteed
2. All performance values and system function shall be guaranteed only when network performance is at least:
    1. The equivalent of 2Mbit/second between local terminal and server-node
    2. The equivalent of 2 Mbit/second between all client-nodes and the database-server node
3. Response times to all local-terminal (LUI) actions shall be less than 100ms average and must not exceed 300ms in worst cases
4. Response times to all remote-terminal (RUI) actions shall be less than 1 sec. average and must not exceed 3 seconds in worst case
5. In order to achieve reliability and performance, LLM-DB-instances shall be as close as possible to the local terminal and server nodes.
6. To meet database concentration and local distribution requirements alike, a database replication scheme shall be folowed:
    1. local database shall be replicated into a regional instance
    2. regional instances shall be replicated into a global instance
    3. change propagation, propagation latency and direction as well as data-mastership (e.g. local overwrites regional, ...) shall be flexibly configurable inside the DB itself.

LLMCMSS.07.02.07 - volumetrics

1. One local terminal shall support one user at a time under normal circumstances. To cater for headroom, one local terminal support all of the combinations below:
    1. 4 interactive user sessions (LLM-LUI, CTC) at a time or
    2. 2 interactive users (LLM-LUI, CTC) and services (CMS,LRF) and database (LLM-DB) or
    3. 3 interactive users (LLM-LUI, CTC) and services (CMS,LRF)
2. One site-local server shall support:
    1. 60 user sessions AND services for them (CMS,LRF) OR
    2. 30 user sessions AND services for them (CMS,LRF) AND database for them (LLM-DB) OR
    3. database for 240 user sessions

3. One regional database server shall support
    1. active database for 1200 users OR
    2. backup/replica database for 10000 users
4. One global database server shall support
    1. active database for 10000 users OR
    2. backup/replica database for 50000 users

LLMCMSS.07.02.08 - Safety and Runtime Environment

1. All software shall be able to run under a configurable operating system user account.
2. Neither for installation nor for execution administrator privileges shall be necessary. If there are admin provileges required for certain preparations, these actions shall be described in detail in the technicak documentation and be separately executable form the software installation and execution.
3. All installed and configured software components shall automatically start when the system platform is started and shall be orderly shut down at system shutdown
4. In case of components crashed due to fatal software errors, an automatic monitoring component shall restart the crashed components after collecting evidence about the possible cause and status of the crash

LLMCMSS.08 - LRF functional specification

This is section 8 of the LLM CMS specification. It covers the area of how reporting and user feedback shall be handled and presented.

Reporting and Feedback are performed by the LRF (LLM Reporting and Feedback) component in the LLM system.

LLMCMSS.08.00 - Document Control

LLMCMSS.08 document properties	
Item	**Value**
document	LLMCMSS.08
Issue	0.3
History - 0.1	Created due to RefArch 0.3
History - 0.2	added document control
History - 0.3	- added sample alogrithms (LLMCMSS.08.03) - added LLMCMSS.08.02.15 (serve LLM-DB tree view as web pages) - added LLMCMSS.08.02.16 (static content area for web server in file system)

LLMCMSS.08.01 LRF Motivation and driving factors

This subsection does not contain requirements, all requirements that are derviced from the deliberations in this subsection are explicitely stated in the following subsections.

There are a number of factors influencing the way reporting shall be done and the software providing this information shall be structured:

1. Training components (PTC, CTC) deliver isolated, non-weighted and un-interpreted data points and samples collected during single or multiple training exercises and sessions.
2. There are different types of consumers for the information to be distilled from all this training data:
    1. The users (seniors) doing the training themselves
    2. The trainers and coaches
    3. The medical responsible personnel
    4. The relatives of the users
3. All consumers need different compilations of the information
4. Information needs to be weighted and compared against different benchmarks and thresholds:
    1. Absolute Values
    2. Relative to peer / reference group
    3. relative to own history
    4. in the context of medical status
    5. in environmental context
        1. weather
        2. living environment
    6. synchronized with other flows of events:
        1. medication

        2. personal events
        3. time of day/year/season
5. The way this performance information has to be presented optimally also may vary due to several factors:
    1. type of senior personality
    2. psychological stability
6. The nature and format of presentation and also the information extraction algorithms may vary also amongst installation sites and employed training equipment
7. All above parameters will most probably also vary at least during the course of the trial phase
8. There may be the need for testing the reception of various forms of performance feedback and collecting this type of "meta-feedback" information (feedback about the type of feedback)
9. As a matter of system and software stability as well as a consequence of the above factors plus the fact that not all (if any at all!) types of feedback presentation and algorithms will be known at the first deployment of the LLM system, there needs to be a high flexibility and modularity in the împlementation and configuration of the preparation and presentation of training feedback information
10. For Version 1.0 of the LLM system the following limitations in the feedback generations may apply:
    1. It is assumed that the training components already deliver weighted and interpreted results (good/fair/bad, etc.). So the LRF only has to do presentation, no decision-making.
    2. No correlation across the results of different training components and/or ILC. Also, no correlation with external data.

LLMCMSS.08.02 LRF Functional parameters

These requirements are partly derived from the factors found in LLMCMS.08.01, partly from LLMTOS, partly from other sections or external documents.

1. There shall be a reporting and feedback mechanism built as an autonomous component in the LLM system. This component shall be called LRF for brevity in the context of this specification
2. LRF shall work on measurement data deposited in the LLM-DB by training components and other sensors (Also ILC, as an example, shall be able to produce and deposit measurements data, for instance physical mobility detected by movement sensors)
3. There shall be 2 types of information calculated in the LRF, Reports and Feedback. The difference between Reports and Feedback shall not be technical, but rather conceptual and possibly in the selection of presentation methods:
    1. Reports:
       This shall be calculated values, statistics, charts, etc ...
       There may be more detailed information in a report and it may usually be shown in a RUI / Web Browser Environment
    2. Feedback:
       This shall be personalized information for the target audience
       This type of information will most of the time be presented on a local terminal in a LUI environment
4. Each generated information entity shall be qualified with the intended target audience and privacy, plus possibly additional qualification tags, plus security control.
5. Input data shall be taken from the LLM-DB
6. Generated information shall be stored in the LLM-DB, into the same structure as the inpout is taken from in order to generate feedback and/or reports based on provious feedback/reports. Alternatively the LRF shall be configurable in the way from where to take the input data and to where to put the results.

7. Generation of information shall be able to be triggered by external events from other LLMSYS components and by internal schedules
8. Completion of information generation shall be able to generate events to trigger other activity like:
    1. user interface presentation and subsequent information processing. The generated trigger event shall be able to carry properties passed on from the event triggering the information generation. These properties shall include local terminal ID, session ID and all properties to bring the report data into foreground on the respective local terminal.
    2. Workflow event(s). For this purpose the triggering event shall be able to carry flow, step, event, session and user information necessary to identify the intended action in the workflow to be triggered by feedback generation.
9. There shall be "pluggable" statistics and report generation algorithms. The Algorithm as the "loadable entity" shall be embedded in a JAVA class library that is dynamically found and loaded either at web application startup or, if allowed by the JEE application container, on demand when required the first time.
10. Selection of applicable generation algorithm shall be able to be based on any data item available in the LLM-DB
11. There shall be multiple LRF-instances per LLM-System instance, each with their own configuration and loaded algorithm sets
12. Reporting results stored in LLM-DB shall be any combination of:
    1. Graphic Images: charts, traffic-light indicators, bar-graphs, ...
    2. single text or numerical values
    3. lists and tables
    4. general HTML
13. In order to leverage already existing developments and their evolution, existing processing libraries and components shall be employed, like:
    1. GNUplot, JFreeCharts, etc .. in the presentation level
    2. decision support frameworks and rules languages for the knowledge bases
14. All configurability shall be deposited also in the LLM-DB using configuration tables and properties
15. The LLF can display an arbitrary HTML document in any LUI on any local terminal with the only restriction, that a regular URL must be provided to access the document. In order to let report results be stored at any logically sensible place, any item in the LLM-DB shall be accessable with a unique URL. This shall be handled by the LRF web-service using the unified tree API defined in LLMCMSS.04.02.14 by appending the tree path to the property containing the HTML to an URL served by the LRF service component.
16. Additionally there shall be a filesystem tree on the platform where LRF is deployed to store HTML, picture files and assistive files to be served by the RUI-webserver as static content. This directory tree will be used by report algorithms to deposit automatically generated picture images like charts, icons etc.
    The root of this tree shall be stored in the configuration area of the LLM-DB in 2 forms:
    1. the filesystem path on the OS platform
    2. the URL-path where the web server will serve it

LLMCMSS.08.03 Report and Feedback Algorithm Examples

This subsection provides a number of report and feedback algorithm examples in orer to show how such an algorithm shall be constructed and to give a hint about what it shall be capable of.

LLMCMSS.08.03.01 - Sample Report 01 - single training session feedback

Input parameters:

- Training session result ID (to directly access the result of the respective training session in the LLM-DB tree)
- LLF-instance ID (a pointer into the LLM-DB to the device control structure of the local terminal LUI session where the result shall be displayed)

Algorithm result:

- Result shall be an HTML document like the example in LLMCMSS.06.04.07 showing:
    - A symbol for the training type performed
    - a 3-state traffic light showing RED, YELLOW, GREEN depending on the improvement level of the senior in this session
    - a short text also reflecting the improvement level

Algorithm implementation:

- take the training result from the LLM-DB with the tree API (LLMCMSS.04.02.14)
- take the user's last result in this training type from the user property "LAST_TRAINING_RESULT_<training-type>"
- compare the values to detect if user has improved
- use link to RED light picture if dis-improved, YELLOW, if constant and GREEN if user has improved, choose text accordingly
- deposit the resultant HTML document in the static content file area (LLMCMSS.08.02.16)
- send the URL of the result in a "Feedback_Ready" event to the LLF-instance passed at the input

LLMCMSS.08.03.02 - Sample Report 02 - Therapist overview for 1 month and 1 senior and 1 training type

Input parameters:

- month and year for which to report
- senior ID for whom to report
- training type ID for which to report
- LLM-DB tree path to the property in which the HTTP-URL of the result shall be deposited

Algorithm Result:

- bar chart with one column for eachavailable result, auto-scaled in X- and Y-axis

Implementation:

- use LLM-DB tree API (LLMCMSS.04.02.14) to select all training results for the given senior and training type
- use JFreeCharts (or similar) library to create a PNG graphics document and store it in static web file space (see LLMCMSS.08.02.16)

LLMCMSS.09 - CMS requirements in LLM-Database

This section summarizes all the data structures and access mechanisms used by the LLM-CMS in the LLM-Database.

As the LLF is considered as an external component to the CMS subsystem, it can be considered as if had an "ENVREQ:" in front of it

All CMS subcomponents are clients for the LLM-DB components and store their persistent data there in structures described in the section.

LLMCMSS.09.00 - Document Control

LLMCMSS.09 Document properties	
Item	**Content**
Document (part)	LLMCMSS.09 - LLM-DB requirements by CMS subsystem components
Issue	0.4
History - 0.1	created as seperate section out of LLMCMSS.04.03.02, LLM-DB requirements.
History - 0.2	formatting, headers
History - 0.3	- mark whole LLMCMSS.09 as an ENVREQ - require unique object IDs for all LLM-DB objects in LLMCMSS.09.03.02.11
History - 0.4	- add structures for unified access, access protection and security with ref. to Q.16/17/18/19 in LLMCMSS.09.03.03

LLMCMSS.09.03 LLM-DB interface required by CMS function

LLMCMSS.09.03.01 General PROPERTIES structure in LLM-DB

As some, if not most of the specific properties and attributes required by most of the components in the LLM-system CMS subsection are not yet known, expandability shall be built into the LLM-DB in order to keep the database structure stable and open for expansion.

1. There shall be a PROPERTIES structure in the LLM-DB with a hierarchical structure to be used for an enhanced version of the Name/Value pattern
2. properties shall be able to be grouped in structures like sections (as known from .ini files) or directories (analogy with file systems)
3. sections also should be properties having names, the value being the sub-structure, producing tree-like organisation
4. Each property entry item shall have a unique ID. Uniqueness shall apply inside one instance of LLM-DB. No uniqueness may be guaranteed when one item is deleted and another different one is created. In this case the unique ID may be re-assigned. Applications should not rely on this uniqueness or, if required, take care of it on the application level. However, the unique ID shall be able to be used when traversing structures. If an entry has been deleted and is requested again by its unique ID, it is permissable to return a NULL value or to throw an exception (on the client API library side).
   Additionally, unique IDs may change when exporting a sub-tree and re-importing it. This ensures

the possibiity to import the subtree at another place in the higher hierarchy tree or importing it more than once, for instance in order to implement one type of templates.

5. A property name shall be a reference to a place in a certain property tree, and shall be able to be either of:
    1. absolute path in LLM-DB property tree, sections separated by "/" (preferable in the client library using path-separator variables)
    2. relative path to the "current" context the client is (using a context reference in the client library keeping track of the path context
6. A property value shall be one of:
    1. number - shall be able to hold 64-bit-FLOAT and 64-bit-INTEGER with auto-conversion if necessary
    2. string (unicode) - may be limited in size but no shorter than 1024 characters
    3. Document / Large String (CLOB)
    4. collection (=> strong link) of more properties
    5. reference (=> weak link) to one of: - recommended to be used only for certain cases as templates and shared properties
        1. collection in same DB-instance
        2. relative pathname to any type of item value
        3. absolute pathnameto any type of item value
7. properties references shall be usable in all other LLM-DB structure, however, LLM-DB may not need to be aware that some string is a property path
8. There shall be a client library in JAVA providing collection, properties and iterator interfaces to the properties structure
9. Methods in the client library shall keep a context to support buffering and maintaining a "current" place in the property tree
10. There should be methods in the client API to handle absolute and relative paths

LLMCMSS.09.03.02 Classes / Collections of items required and processed by CMS in LLM-DB

1. Seniors, enhanced by property handle for each entry
2. Users, enhanced by property handle for each entry
3. User rights, roles and permissions managed in property subtree named "auth". Master is account management in CMS components, all other clients must only read with robustness for changed structures between traversal reads.
4. Sessions, enhanced by property handle for each entry if already existing. If not yet existing, implemented as property subtree named "sessions". Sessions shall only be changed by the CMS component session management API, all other clients shall use this subtree as read-only with all the caution required by the fact, that one client may traverse the session tree for viewing purposes and the session management may delete the whole subtree or at least part of it.
5. Equipment Data: local terminals, Server Boxes, PTC devices, enhanced by property handle, if already existing and used by other components. If not yet existing, implemented as property subtree named "devices"
6. System Configuration Data, a property subtree named "sysconfig"
7. General Config-Items / Properties / Attributes in a property subtree named "config"
8. Activity-Information / Notification Data / Event Data in property tree(s)
9. Workflows in property tree(s) managed on a higher structural level inside the workflow processor, located in properties subtree named "workflows"
    1. Templates
    2. Instances
    3. Steps
    4. Agents, with relation to users

10. Schedules, Appointments (vs. ICalendar, ...), attached to users or devices or system general ("sysconfig") as properties subtrees named "calendar"
11. All objects processed by LLM-DB clients shall have globally unique immutable object IDs. These shall NOT be re-used if an object is deleted and a new one is created. This may lead to a new object ID if an object is deleted and created with the same set of attributes, but this is design intent.

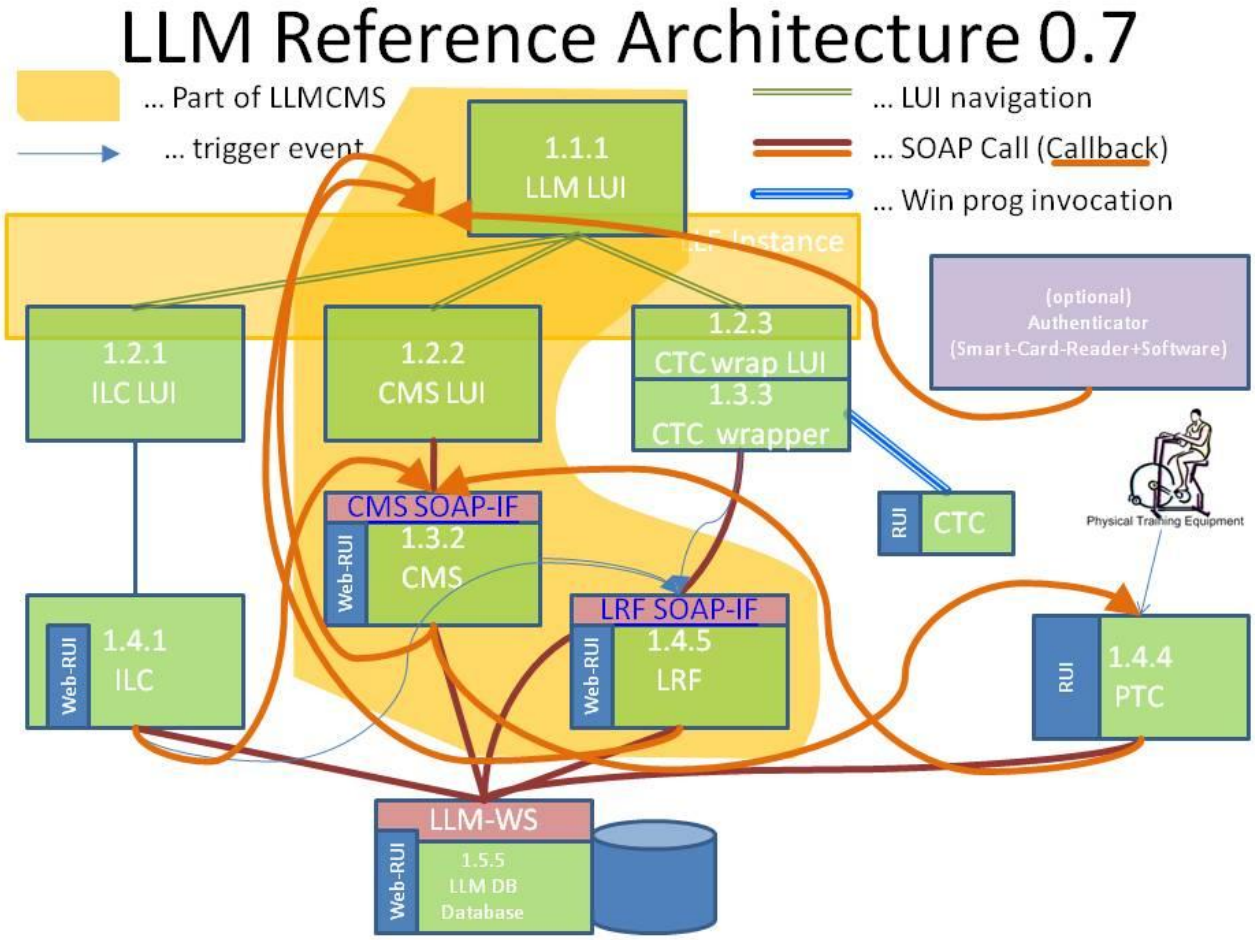
LLMCMSS.09.03.03 Methods for unified and secure access to LLM-DB items

1. A unified access method to all objects in the LLM-DB shall exist as specified in LLMCMSS.04.02.14. (See discussion in LLMCMSS.B, Questions 14 and 15)
2. The unified access API shall incorporate the access protection in order to avoid sending unauthorized data across machine boundaries.
3. Access security shall allow for a fine grained system of controlling access to any node and/or subtree of the data model defined by LLMCMSS.04.02.14.
4. Access security shall be able to allow/restrict operations ADD,READ,ENUMERATE,CHANGE,DELETE on any subtree by a system of access control lists (ACL), consisting of access control entries (ACE).
5. Any ACE shall allow any combination of the mentioned operations to one node to the holder of a unique rights-identifier.
6. A rights-identifier shall be able to be held by any user, role and group.
7. Any user, role or group shall be able to hold an arbitrary number of rights identifiers.
8. A rights identifier shall be stored as a NUMBER of min. 64 bit size and shall be able to be unique not only in one LLM-DB instance but globally across all thinkable LLM-DB instances, if necessary by a system of prefixes and classes of rights-identifiers clearly marked as "local" or "private"
9. ACLs shall be able to be attached to any node in the unified LLM-DB tree.
10. When checking access rights for a specific operation to a specific node, in the case of the checked node having no ACL, all the ancesters of the node in the tree shall be checked for an applicable ACL.
11. rights identifiers shall be able to have names, stored in a seperate section of the LLM-DB unified tree and carrying the numerical value of the rights identifier
12. See also the discussion in LLMCMSS.B, Questions 16 thru 19, for implementation hints and concepts

LLMCMSS.10 - LLM CMS reference architecture

In order to refer to functional entities in the CMS specification, a reference architecture of the LLM system is assumed. The LLM CMS specification will refer to parts of this reference architecture when it comes to specify attibutes or behaviour or interfaces between entities.

LLMCMSS.10 - LLM CMS Ref Arch document properties	
Item	Value
Document	LLM CMS reference architecture
Issue	0.9
History - 0.1	Created with reference architecture V 0.1
History - 0.2	Updated reference architecture V 0.2
History - 0.3	updated reference architecture 0.3
History - 0.4	added document control
History - 0.5	- updated reference architecture V 0.4 -- point out callbacks from services to LUI to display unsolicited information -- removed PTC LUI as PTC shall have no LUI
History - 0.6	- clarify CTC integration - clarify PTC integration - updated reference architecture V 0.5
History - 0.7	- updated reference architecture V 0.7 - include EXTAUTH external authentication device integration LLMCMSS.10.03.10 - clarify "LLF instance" and its web-service invocation API
History - 0.8	- refer to LLMCMSS.07.02.01.02 in LLMCMSS.10.02.02.03 - add PTC-LUI to LLMCMSS.10.03.01 with reference to LLMCMSS.07.02.01.02 - let LUI components explicitely refer to LLF/LLMCMSS.05 in LLMCMSS.10.03.01.06, LLMCMSS.10.03.02.03, LLMCMSS.10.03.03.03 and LLMCMSS.10.03.04.02
History - 0.9	- correct functionality provided by CMS-LUI, in LLMCMSS.10.03.04.01



The elements in the yellow shaded area in the picture are objects of the LLM CMS specification. All other objects are used as LLM CMS have interfaces to them or their interfaces are used by LLM CMS too.

LLMCMSS.10.02 - Concepts in the reference architecture

This reference architecture has been chosen to reflect the most probable functional component landscape the LLM system (the LLM system is assumed to be the technical system providing the LLM service) will be composed of.

LLMCMSS.10.02.01 - Seperated and not separated components: LUI and RUI

There are components that look similar but have completely seperate behaviour and implementation. Examples of such component "pairs" are the LUI and RUI components.

Each component has a LUI - Local User interface - and a RUI - Remote User Interface. The LUI is the user interface for the local user and performs user-centric tasks. The RUI is the user interface for the remote access user and the administrator and performs user-centric tasks as well as administrative, management, configuration and troubleshooting tasks. To a certain extent (detailed with each component having LUI and RUI) the LUI is a (rather small and simple) subset of the RUI.

The RUI is an built-in functionality of each component and as of LLM system architecture V 1.0 an HTML-based web user interface. It provides all user interaction the respective component is capable of.

As a result, the RUI is tightly bound the the component itself and uses only loose coupling to other RUIs and therefore assumed as being part of the component itself.

The LUI is more simply and user-centered. All LUIs of a LLM-system are much more tightly and seamlessly integrated to provide maximum usability and comfort. Therefore, the LUIs share a lot of common functionality and are more tightly interwoven than the RUIs. Moreover, the LUIs are in some places "wrappers" around the "original" user interface of a component. As a result, the LUI of a component makes sense to be separated from the component.

LLMCMSS.10.02.02 - Integration of already existing components

1. Moreover, already existing components, as CTC, ILC and (to become) PTC, shall be able to take part in this architecture as-is.
2. However, ILC, CTC and PTC are here referred to as placeholders for ***any*** ILC, CTC or PTC that fulfils the behaviour required for taking part in a functional LLM system. How this can be accomplished is detailed below in the respective section(s) covering these components.
3. Special attention shall be given to re-usability by well-chosen implementation of integration components, detailed especially for CTC and PTC in different variants set forth in **LLMCMSS.07.02.01.02**.

LLMCMSS.10.02.03 - LUI - Local User Interface and LLF - Local user interface framework

There are a number of LUI - Local User Interface - Components in the LLM reference architecture. The LUI is a user interface intended for the local, not technically skilled and, most probably, physically challenged, user.

For the purpose of the LLM CMS the assumption will be made that the eHome LUI already has proven to be able to meet the requirements for a LLM LUI. Furthermore, it is already existing and can be easily adapted to the functions required in LLM LUI.

Therefore, the eHome LUI software component and platform shall be adopted for all LUI components in the LLM architecture. The framework extracted from the eHome software package shall be called LLF - LLM LUI framework - for the sake of brevity and identification.

The LLF is a windows application running on a specific set of touch-enabled net-top intel x86 hardware and operating system platforms. It is specified in the LLF specificationpart of this specification.

LLMCMSS.10.03 - Components of the reference architecture

The reference architecture is divided up into components each fulfilling a specific functionality.

LLMCMSS.10.03.01 - 1.1.1 - The LLM-LUI

1. The LLM -LUI is the central local user interface to one LLM-system. It is the central entry point to all components of an LLM instance. The LLM-LUI shall allow access to:
    1. ILC-LUI - The local user interface of the independent living component
    2. CTC-LUI - the user interface of the cognitive training component
    3. PTC-LUI - the user interface of the physical training component, see also variants considered in LLMCMSS.07.02.01.02

4. CMS-LUI - the administration user interface for the whole LLM system

2. The LLM-LUI shall also present the single point of authentication for a user. After login into the LLM-LUI, all user credentials shall be carried on to the connected subsystems and their LUIs. A user shall only need to login once and then be known to all subsystems with all their data and access rights.
3. The process of making one LUI visible, most probably implemented by bringing its window to the front of the display screen, is named switching.
4. The LLM-LUI shall switch between the respective user interfaces as seamless as possible (depending on the ways viable in integrating the user interfaces of already existing components).
5. Switching shall not only be initiated by the human user, but also by a subsystem needing attention shall be able to bring its LUI to the front and attention of the user.
6. The LLM-LUI shall be implemented using the LLF, specified in LLMCMSS.05, in order to be integrated in one single LLF-instance with all other LUI-components of the LLM-system.

LLMCMSS.10.03.02 - 1.2.1 - The ILC-LUI

1. The ILC-LUI is the user interface of the independent living component. It is integrated into the LLM-LUI.
2. As of the time of releasing this specification, the only available ILC implementation used in LLM is eHome . It is assumed that the LLM-LUI will use the same base technical implementation as the eHome-LUI. Therefore, integration shall be as tight as possible and the ILC-integration into LLM-LUI shall serve as the blueprint of an "Integrated Application" in terms of LUI-interaction.
3. The ILC-LUI shall be implemented using the LLF, specified in LLMCMSS.05, in order to be integrated in one single LLF-instance with all other LUI-components of the LLM-system.

LLMCMSS.10.03.03 - 1.2.3 - The CTC-wrapper-LUI

1. The CTC-LUI is a wrapper component around the real user interface of the cognitive training component. It is assumed to be integrated into the LLM-LUI.
2. The Integration of CTC is not part of the CMS development. However, the following assumptions are made to enable CMS development:
    1. CTC is running on the local terminal platform and is a native windows application.
    2. CTC can be started like any other windows application, the PID can be got from the invocation system call.
    3. The completion of CTC can be monitored like any other windows application by watching the PID until disappearance.
    4. Before start, there may be a preparation sequence setting the environment by querying the LLM-DB.
    5. After completion there may be a result gathering sequence writing tot he LLM-DB results store.
    6. All the CTC preparation, start, completion and result gathering shall be encapsulated in a DLL function call "CTC-invocation".
    7. The DLL containing the CTC-invocation shall comply to the LLF function API standard and shall be delivered by the CTC-integration development team (at AUTH).
    8. Whether the CTC-invocation function is synchronous or asynchronous shall be agreed bilaterally between the CTC integration development team (at AUTH) and the LLF development team (at ISTU). Per default, CTC supervision and error handling (timeout, program hangup, etc.) shall be handled within the CTC invocation function call without emburdening the LLF with such supervisory tasks.

3. The CTC-LUI-integration shall be implemented using the LLF, specified in LLMCMSS.05, in order to be integrated in one single LLF-instance with all other LUI-components of the LLM-system.

LLMCMSS.10.03.04 - 1.2.2 - The CMS-LUI

1. The CMS-LUI is the user interface of LLM-CMS, the Central Management System. The CMS-LUI covers a subset of all possible administrative activity of the CMS, namely the part important for the local user, like viewing statistics, schedules and results~~, changing password or other user parameters.~~ More detailed administrative and management activties are handled by the CMS-RUI, specified in LLMCMSS.04.05, built into the CMS, specified in LLMCMSS.04.
2. The CMS-LUI shall be implemented using the LLF, specified in LLMCMSS.05, in order to be integrated in one single LLF-instance with all other LUI-components of the LLM-system.

LLMCMSS.10.03.05 - 1.3.2 - The CMS

The CMS is a web-service providing the "business logic" of the management functionality. It offers an Web-Service-API for a number of functional groups:

- session management (for LUIs and other up-stream components
- user data administration
- general configuration management
- workflow processing
- notification processing
- schedule administration and handling

There can be any number of CMS instances per LLM-DB instance but one CMS instance only has access to one LLM-DB instance.

LLMCMSS.10.03.06 - 1.5.5 - The LLM-Database / LLM-DB

The LLM-DB is the central data storage component of one LLM-system. It holds all persistent data centrally for one LLM-system. All other components are required to refer to the LLM-DB to store and retrieve persistent data.

The LLM-DB exposes the LLM-Web-Service-Interface built on SOAP for all data manipulation and retrieval.

LLMCMSS.10.03.07 - 1.4.5 - The LRF - LLM reporting and feedback functionality

The LRF generates feedback and reports for all types of users when triggered by another application.

Any application activity, when it has data to be brought to the attention of users, can trigger a report (or feedback, when it comes to informing the user about the performance of a training session) to be generated. This triggering is performed by invoking a trigger method on the LRFs web API. The LRF then produces the appropriate representation of the data requested.

The data the LRF works on is taken from the LLM-DB. The result, any type of appropriate information, usually (but not limited to) human-readable and displayable on a LUI or RUI, is also deposited in the

LLM-DB. Input and output of the reporting/feedback process are held in the same area of the LLM-DBs storage, in order to make the result of one report available as input for another report.

LLMCMSS.10.03.08 - 1.4.4 - The PTC - Physical training component

1. All specific PTC-Integration development shall be done by the PTC-integration development team (at AUTH). PTC is assumed to have no LLM-LUI-integratable user and operational interface. The complete interaction between CMS and PTC is done via configuration data and result data in the LLM-DB.
2. There are 2 Exceptions:
    1. There shall be a relation structure in the LLM-DB between local terminal and training program and the PTC-device. The PTC device may have an entry in its configuration structure about a control Web-API. If this entry is present and the control web-API supports emergency-freezing (powering down, etc.) of the device, the "Alarm Setting" Workflow LLMCMSS.06.02.03 shall use this functionality (variant A) to shut down the PTC device in Action 21.
    2. The PTC shall notify the CMS about the termination of a training session via the CMS Event Web-API. If the PTC is not capable of this notification (needs to be detectable by the CMS by looking up the PTC device configuration properties), the CMS will change Workflow LLMCMSS.06.02.02 - "senior performing training session" to not display "PTC Session active" but return to senior main menu or to place a button on the PTC active screen of rthe user to inform the LLM system that the PTC training session is complete.

LLMCMSS.10.03.09 - The LLF instance

The LLF-instance is the running program on the local terminal platform handling all LUI interactions and displaying all the screens.

The LLF-instance is the program that is started either autoimatically of by the user or an administrator and then occupies the complete local terminal screen and is the only component to interact with the user.

The LLF-instance handles 2 types of events:

- user actions by pressing on-screen buttons or entering texts onscreen.
- events "injected" via a web-service interface.

These events can leat to display updates, change of screens or more complex activcity involving interaction with application service web-API (such as CMS or LRF)

LLMCMSS.10.03.10 EXTAUTH - The optional external authenticator

EXTAUTH is an optional software component running hiddedn in parallel with the local terminal LLF instance. Its purpose is to monitor for instance a smart card reader and to detect the presence of a smart card authentication token. It then takes the user credentials (code, ID, username/password or whatever is provided by the card and issues an event to the LLF-instance's web-service API whcih has the same effect like a user logging in.

Optionally, the removal of the smart card can lead to logout or terminal locking, depending on future requirements, but this is NOT part of the current LLM service implementation scope.

LLM Abbreviations

This page tries to list all important abbreviations created and/or used in the LLM project. Please everybody feel free to add your own knowledge, corrections and hyperlinks to places with even more information for a specific abbreviation

- **ACE**
  An Access Control Entry, part of an Access Control List (ACL). An access control entry describes the rights of one specific rights identification on the target object the ACL it belongs to is associated with. ACLs and ACEs are implementation concepts of security regulating access to data object and functions and are discussed in LLMCMSS.B.02.16/17/18/19.
- **ACL**
  An Access Control List, a sequence of Access Control Entries (ACE), associated with a target object it regulates access rights for. ACLs and ACEs are implementation concepts of security regulating access to data object and functions and are discussed in LLMCMSS.B.02.16/17/18/19.
- **CTC**
  The Cognitive Training Component of the LLM System. It consists of a software component interacting with the user via the local terminal of the LLM system, which is integrated with the LLMCMS into the LLM System to provide assistance in cognitive training including scheduling of training sessions, reminders for sessions, controlling the training session according to a training program devices by a therapist and keeping track of the results.The CTC presents the senior with cognitive exercises ranging from associative tests to memorizing to puzzles to "brain games", parameterized for the actual needs of each participating senior.
- **EXTAUTH**
  An EXTernal AUTHentication Device, such as a smart card or other token reader. The external authentication device must be integrated with a software running either on the local terminal or a separate computing node. This software may generate LOGIN or LOGOUT events into the LLF-instances SOAP-API which are then handled like manually entered credential information. The event has to carry credential information and information about the nature of the credential information, as it shall not be limited to username and/or password as it is not wise to store such things on a portable device.
- **GOE** - General Object Editor
  High-Level Semantics-Aware General RUI web-UI function to edit general data structures in the LLM-DB. Specified in LLMCMSS.04.05.03.
- **GPE** - Generic Property Editor
  Low-Level generic RUI web-UI function to edit any data inside the LLM-DB including XML import and export. Specified in LLMCMSS.04.05.04.

- **LLF**
  The LLM LUI Framework. A software base which can be instantiated, configured with screen definitions and enhanced by DLLs to build a LUI (Local User Interface).
  The LLF is specified in LLMCMSS.05
- **LLM**
  Long Lasting Memories. This very project, delivering a unified solution for cognitive and physical health and autonomous living for senior citizens
  - A system (The "LLM system") providing the LLM service
  - The LLM service supporting elder citizens to keep their physical and cognitive abilities healthy.

- **LLM-DB**
  The LLM-Database. LLM-DB is the central persistent storage location. No other persistent dependable storage exists in the LLM system.
- **LLMCMS**
  The LLM Central Management System. The part of the LLM System keeping the parts providing the service (CTC, ILC, PTC) together and providing means for user interaction and administration. The LLMCMS is specified by the LLMCMSS (LLM CMS Specification) and implemented by the LLMSDV (LLM System DeVelopment) activity inside the WP3 - Service Adaption and Customization
- **LLMCMSS**
  The LLM CMS (Central Management System) Specification. This structured document specifies functional, non-functional and implementation aspects of the LLM CMS.
  The LLMCMSS is separated in 10 parts, called LLMCMSS.01 thru LLMCMSS.10, plus a Q&A-Section/Commentary, called LLMCMSS.B
- **LLM system**
  The technical system, comprised by a wide area network of computers (local terminals, ILC components, LLM servers, database servers) and a software configuration, providing the LLM service
- **LLMTOS**
  The LLM Technical and Operational Specification, LLM project document D3.1 in work package 3. THE LLMTOS is the main document on whh the LLM CMS specification (LLMCMSS) is based.
- **LRF**
  The LLM Reporting and Feedback functionality. A flexible software component that can contain numerous algorithms to generate reports and feedback for seniors, their relatives and therapists about the physical and cognitive performance and other parameters measured and stored inside the LLM system.
  The LRF is specified in LLMCMSS.08.
- **LUI**
  The Local User Interface. A simple, easy to use, touch-screen based user interface for seniors to interact with the LLM system.
- **PTC**
  The Physical Training Component of the LLM System. It consists of a physical exercise machine like an ergometer, treadmill or Nintendo-Wii-based fitness training component, which is integrated with the LLMCMS into the LLM System to provide assistance in physical training including scheduling of training sessions, reminers for sessions, controlling the training session according to a training program devices by a therapist and keepting track of the results.
- **WFMC**
  The WorkFlow Management Coalition, an association of companies and people working on standardization and tools around workflow management .
- **XML**
  XML (Extensible Markup Language) is a set of rules for encoding documents electronically.
- **XPDL**
  The XML Process Definition Language, a language for modeling workflows and business processes, based on XML

LLM CMS Specification Appendix B - Questions and Answers

This part of LLMCMSS summarizes questions being posed over time and their answers.

LLMCMSS.B.00 - Document Control

Document Item	Value
Issue	0.4
History - 0.1	2010-03-08 - created with initial set of already asked questions
History - 0.2	- completing all answers with references to specification part up to Question 14
History - 0.3	- adding more detailed Implementation Hints for security and unified tree API with questions 15,16,17,18,19
History - 0.4	- add more examples for LLM-DB as viewed by API as of LLMCMSS.04.02.14 in question 20

LLMCMSS.B.01 - Overview and Motivation

In principle, all questions should be answerable by the contents of the specification itself. Indeed, every question that is not covered by a part of the specification, results in updating the specification, if only to exclude certain aspects from the scope of the specification. However, some answers are not so easy to find by only reading the specification text. Moreover, some questions will be asked moe often as they come to mind frequently. So, in order to provide one more aspect to the matter, this part answers the questions and provides sort of a commentary to help in understanding some of the not so obvious parts of the formal specification text.

LLMCMSS.B.02 - Questions and answers in Detail

1. Q: I went through the “LLM CMS Use Cases” and as I see the GUI plan of RUI part is not complete yet. Can I rely on a final version soon or is it just meant to be a guideline to see how it should look like? In the latter case I would figure out the required RUI functionality through the WSDL and documentation.
   A: As of LLMCMSS V 0.8, there are only few, but most versatile functionalities in the RUI.
   A: The RUI functionality is explicitely specified in LLMCMSS.04.05
   1. Senior training result viewer
   2. Property editor.
      The property editor can be usede to manage senior data, equipment data, schedules and appointments and even training programmes
   3. General Object Editor
   4. Schedule Editor
   5. User Manager (add/remove/modify)

2. Q: I checked the Web Service API documentation and I would like to suggest a few enhancements if you don't mind. -> Evdokimos ? (WH)
   Seniors are identified by ID and the delete, get, edit works by using this ID only. Thus if the admin e.g. wants to edit a user's profile the system has to get all seniors to get the ID. I would suggest e.g. `find_seniors("fname", "lname", "country","city","active", "username", "password")` method with optional parameters so the query can be targeted to specific users. Or is it expected from the admin/therapist to remember IDs?
   In case of Users there can be `find_users("fname", "lname", "Role", "username", "password")` method as well for the same reasons described in the previous section.
   It would be helpful to have an API method (e.g. `get_seniors_of_user("user_id", "username", "password")`) which returns all the seniors associated with a user. Or the get_seniors_relations_from_user method could do that if the senior_id parameter is zero and the user has permission to get his/her associated seniors.
   Is it possible to extend the Web Service API with above mentioned functionalities?
   A(WS): will forward this to AUTH/EK along with the suggestion to include the tree-API also into the LLM-WS
   I completely agree with your proposal for a "xxx_find" functionality in the LLM-WS, I would like to ask the guys from AUTH to provide the tree-API I suggested in LLMCMSS.04.02.14 already on the LLM-WS. By using this API and a "list" or enumeration functionality, all objects in a certain context could then easily be found.
   Additionally, I enhanced LLMCMSS.09 be the requirement, that every database object shall have its unique, immutable ID.
3. I saw in the "LLM Reference Architecture 0.7" figure the "RUI blue box" in the CTC and PCT and the Web-RUI in the ILC box which I don't fully understand. As I understood the RUI is a web-based interface for remote access via the Internet (for care takers, trainers, admin) and not for directly managing e.g. a PTC equipment.
   A: the RUI in the PTC box is completely conceptual, has nothing to do with CMS RUI and is assumed to have no integration into CMS. It's out of scope. All PTC management interaction is intended to be by LLM-DB properties entries.
4. Q: Shall we take care of the logging functionality to log all the actions done through RUI? Or is it done at Web Service side?
   A: LLMCMSS.04 specifies that all CMS-Web-Service activity shall be logged. If the RUI accesses functionality that the web-service also provides, this shall also be logged, although the CMS RUI web-application will most probably not go through the SOAM-mechanism to manage things like users or schedules. I'm not sure if it makes sense to log anything (except for the usual debug and troubleshooting logs) inside the RUI.
   However, things like session setups, logins, LLM-DB changes etc. shall be logged (as they are offered on the CMS-WS).
5. Q: Is there test data in the LLM-DB to test the LLM Web Service API? If so can we have access to it? -> Evdokimos ? (WH)
   A: WS already asked AUTH/Evdokimos to provide a LLM-DB test-installation with the already existing web service and including test data. PLease dont expect too much here, as these data is either completely synthetical and therefor not very realistical or very confidential and therefore incomplete at best!
   WS to ask Josef to further keep track of that matter.
6. Q: As I see the passwords are not encrypted at the moment. Will this change in the future or how will the security take place?
   A: expect passwords not to be stored in cleartext, especially passwords of user accounts providing

broader access. I suggest to hash passwords, like in the Unix-OS and to store only the hash value. Entered passwords when authenticating will then also be hashed and can thenm be compared.

7. Q: this training result viewer. what kind of features does it have?
   A: The training result view on the RUI is specified in LLMCMSS.06.02.04, but not specifically for the RUI. In essence it shall be able to view training result reports produced by the LRF. The LRF shall produce training results for each target audience: seniors (to be displayed at LUI), therapists and relatives (displayed at RUI). All these reports are HTML documents and are marked with the senior, the training programme, the session (if applicable) and the target audience. There are already raw training results in the LLM-DB in the "CTC/PTC seniors activity" objects. I'd expect the reports to stored in the generic "properties" or in the user-specific "properties" sub tree. The report itself, being an HTML document can in the essence be stored in a single STRING-type property. The pictures to be included and CSS-stylesheets and other formatting elements shall be managed in a filesystem tree on the platform the LRF and the RUI are running, more detailed specified in LLMCMSS.08.02.16 and LLMCMSS.08.02.15.
   A: RUI based training result viewer is explicitely specified in LLMCMSS.04.05.05
8. Q: can a trainer get notification?
   A: A trainer is not forseen to receive any notification, only (video) calls for supporting seniors, but this is not part of LLM system functionality. LLM system need not foresee such a thing, except it can be covered by a LUI functionality/workflow.
9. Q: can notification appear in RUI?
   A: For the time being there is no notification to appear in RUI. RUI, for the time being, has no callback-mechanism foreseen. However, there is hope that wicket or another component based framework would support such a functionality.
10. Q: everybody can send notification to anybody?
    A: Notifications are intended to be used by the system, not by users. There are currently 2 sources of notifications:
    1) the ILC danger or stress detection, producing a notification with the "distress confirmation screen"
    2) The appointment management system, producing a notification when an appointment is due.
    Each notification may evoke its own user dialogue: an distress alert notification would evoke the "distress confirmation screen", a training schedule would evoke a dialogue with a button to immediately start the training.
11. Q: ad LLMCMSS.04: Lock/Unlock User Account is ok, but what do you mean by Lock/Unlock Session
    A: Lock/Unlock session would mean that the user leaves the terminal and the terminal and session stays locked unless the user again provides password or other credentials. No specific user procedure and use case for now.
12. Q: what did you mean by "change terminal" here at LLMCMSS.04.02.5
    A: I thought, that there may be a scenario where a user would move from one terminal to the other but keep her/his session with all status and credentials. From the session point of view it would only mean changing the reference to the active associated terminal. No defined user procedure and use case do exist now.
13. Q: Sessions as in LLMCMSS.04.02.5; do we have to keep track of users?
    A: A session should be existing as long a user is logged in. It shall be used to store session-wide values, e.g. the position where the property editor has its current editing point, the current user, the local terminal the user has logged in, etc, ...
14. Editing various things: the general object editor, a discussion:
    Q: About editing various things: trainer has to have a place at RUI where he can create trainings, appointments, schedlure.
    A: This is correct, there are already some sketches for a LUI-based schedule editor. Woiuld need to complete them for RUI purposes.

On the other hand, there is a thought that comes to mind:

Could there be a generic object editor?

There is already the general property editor, but this one is not aware of the specific semantics properties ob objects have, not even time- and date-values.

A general object editor will need some meta-information about classes of objects: what properties of what type and how to enter them in what kind of user interface and how to check them to be consistent.

If we could think up such ageneral object editor, we had a great component for all types of information managed in the LLM-DB. The metadata about classes and their properties could be stored in the LLM-DB Tree itself.

Q: could you please write an example of a generic object and a general property?

A: An object would be an appointment. The appointment class would have a few properties: a date/time, a task, a user for whom the apointment is and some repetition properties

A genral editor editing an appointment would neeed to know that an appointment had these properties and what value ranges these properteis would have to have. It could take this meta-information for the meta-information property tree.

Q: is the metadata for "connecting" properties to objects?

A: The editor could then e.g. offer a date/time chooser for the date/time values, offer a dropdown-list for a limited-value range or complain if somebody entered the 31-FEB.

Sorry about terminology confusion:

An object would be anything having a name and any number of properties.

In the LLM-DB-tree-API objects would be in the nodes of the tree, having properties being primitive values or objects with arbitrary properties themselves.

Adding an object would be accomplished by adding a subtree to the total tree, thus adding an object with a speficic name and unique ID into a specific place in the tree hirarchy.

An example using the appointment again:

A "senior"-object has a number of fixed properties, like defined in the LLM-WS and, additionally, a set of generic properties by having a "properties" tree attached to it. In the "properties" tree, an "appointments"-subtree can be created. Each item in this "appointments" tree being one appointment with a number of properties, as defined for an appointment.

The only fixed meaning given to positions in the tree and enforcing specific names of properties is in the software processing the appointments, in this case the CMS managing the appointments and notifying users if an appointment is due.

The difference between the generic property editor and a general editor using metadata to enforce the intended meaning of properties is like having a file-system:

you could edit each file with a hex-editor, but you have to know what you're doing, and it's dangerous, not to mention boring.

Therefore there are different tools: text editors, word processors, outlook, excel, ...

Q: how can we imagine the metadata in the tree?

A: I can imagine a subtree called "CLASSES", each entry being a class definition carrying the name of the class. Ech class definition would have a "MEMBERS" subtree, each entry being the definition of a member variable carrying its name and having properties defining its type and value ranges etc, ...

Q: as I see, objects, are members, users, Components.. and unlimited properties can connect to these objects.

A: Thats right. The metadata can then also contain information about how to present such an object on the RUI, how to enter the value and many more.

A: The General Object Editor, abbreviated GOE, is explicitely specified in LLMCMSS.04.05.03

15. Q: How may a generic node in the LLM-DB tree look like and what attributes shall it have?
    A:
    1. It shall be able to hold:

1. a NUMBER value
2. a STRING value
3. a NAME
4. a PARENT reference
5. an ACL reference
6. a bit-mapped FLAGS mask

2. The reference fields shall be numerical references uniquely referrring to another tree node.
3. The NUMBER value shall have the same numerical range as the reference fields in order to be able to re-use the NUMBER also as a reference.

16. Q: How can a implementation model for access control security on the LLM-DB tree look like and what features shall it have in detail?
    A(1): The security shall consist of an access control scheme allowing to grant separate rights for READ, CHANGE, ADD, DELETE on each single node in the tree. The API to access the tree shall check the relevant access permissions before performing the respective operation (Read, Enumerate, Change, Add, Delete). The tree API shall be enhanced by security checking utility methods like "check_read_access(node_id)". If an API call enumerate nodes in a sub-tree, nodes not allowing READ for the security principal invoking the function shall not be enumerated. For a client principal having no READ access right for a certain node, this node shall not be visible, as if it were not even existing. There may be special cases to be considered in this case if a enumeration method is used to check the uniqueness of a name or picture code. Most probably, adding nodes with to-be-unique properties or changing to-be-unique values will have to be executed with higher privileges capable of always READing the complete data set involved. Another concept would be to integrate such operations in the tree API and to allow complete enumeration of the subtree in case of ADD or CHANGE permitted for one sub-node in the tree, but this complete enumeration would only be carried out in the tree API and would not comprise a security leak.
    A(2): The rights shall be assigned to entities identified by so called rights identifiers. Rights identifiers shall be entities able to be owned by users, groups or roles. Assignment of rights shall be stored in access control entries (ACE). One access control entry (ACE) shall assign any combination of READ,CHANGE, ADD, DELETE to one rights identifier. To each node in the LLM-DB tree there can be an access control list (ACL), consisting of access control entries (ACE). If a specific tree node does not have an ACL attached to it, access control may be specified by any of its ancestors in the tree (parent, parent of parent, etc.)
17. Q: How can we implement access control to API methods, menu items, GUI operations and other objects not explicitely in the LLM-DB tree?
    A(1): Access control must, as with the LLM-DB tree, be checked by the relevant methods, functions or procedures executing the function behind the API or GUI control.
    A(2): In order to store access control information to non-tree objects, there shall be proxy-objects in the tree carrying the ACLs for the external entities.
18. Q: Where shall the ACL for any tree node be stored.
    A: One implementation my be to equip each node with an ACL-reference. The ACL would then by built up also by nodes in the tree, only assigned a special meaning to represent an ACE. The parent-reference may be used as a linking reference between all ACEs of an ACL.
19. Q: How shall a tree-node be used as an ACE?
    A: The FLAGS mask shall be used to store flags for READ, ADD, CHANGE and DELETE rights granted. The NUMBER field shall hold the rights identifier

20. Q: How will the unified tree for a sample set of users look like

A: An example of the unified tree as seen "thru the eyes of" LLMCMSS.04.02.14, including the possible mixture of already existing fixed structures and flexible add-ons by properties may look like the following:

```
"/" - type PROPS (root)
"/USERS" - type PROPS (the fixed part already provided by the LLM-WS "users" structure,
mapped to the virtual tree as seen by unified tree API
```

"/USERS/klaus_schwarz" - type PROPS
"/USERS/klaus_schwarz/lname" - type STRING (the fixed "lname" field of the "user" structure)
"/USERS/klaus_schwarz/fname" - type STRING (the fixed "fname" field of the "user" structure)
"/USERS/klaus_schwarz/properties/user_birthday" - type STRING (the optional property named "user_birthday" added to the properties attached to the user)
"/USERS/klaus_schwarz/properties/children" - type PROPS
"/USERS/klaus_schwarz/properties/children/Frank" - type PROPS
"/USERS/klaus_schwarz/properties/children/Frank/age" - type NUMBER
"/USERS/klaus_schwarz/properties/children/Thomas" - type PROPS
"/USERS/klaus_schwarz/properties/children/Thomas/age" - type NUMBER
"/CTCS" - type PROPS (the fixed structure collecting all LLM-WS objects of class "CTC")
"/CTCS/COMPONENTS" - type PROPS (collection of all CTC components)
"/CTCS/COMPONENTS/treadmill_11/" - type PROPS
"/CTCS/COMPONENTS/Wii_23" - type PROPS
"/CTCS/ACTIVITIES" - type PROPS (collection of all activities of class CTCactivity)
"/PROPERTIES" - type PROPS (the system-wide free space for managing arbitrary configuration and other properties)
"/PROPERTIES/sysconfig/DistressResponseTimeout" - type NUMBER (the system-wide lenght of the reponse timeout in seconds)
"/PROPERTIES/CLASSES" - type PROPS (the subtree containing all higher level class definitions)
"/PROPERTIES/CLASSES/USER" - type PROPS (defining the higher level class USER)
"/PROPERTIES/CLASSES/USER/fields/Age" - type PROPS (defining class field "Age")
"/PROPERTIES/CLASSES/USER/fields/Age/range_min" - type NUMBER (defining minimum allowed value for Age)
"/PROPERTIES/CLASSES/USER/fields/Age/range_max" - type NUMBER (defining maximum allowed value for Age)
"/PROPERTIES/CLASSES/USER/fields/Reference_ID/type" - type STRING (defining the type of the field "Reference_ID", may be STRING or NUMBER)
"/PROPERTIES/CLASSES/SENIOR" - type PROPS (defining the class SENIOR)
... etc ...