

D3.2.2

Smart Lighting System Design

Project number:	257243
Project acronym:	TClouds
Project title:	Trustworthy Clouds - Privacy and Resilience for Internet-scale Critical Infrastructure
Start date of the project:	1 st October, 2010
Duration:	36 months
Programme:	FP7 IP

Deliverable type:	Report
Deliverable reference number:	ICT-257243 / D3.2.2 PU / 1.0
Activity and Work package contributing to the deliverable:	Activity 3 / WP 3.2
Due date:	September 2011 – M12
Actual submission date:	3 rd October, 2011

Responsible organisation:	EFA
Editor:	Paulo Santos
Dissemination level:	Public
Revision:	1.0

Abstract:	In this document the Smart Lighting System Design will be discussed. The purpose of this document is to serve as the basis for an actual implementation of the system.
Keywords:	Smart, Grid, Public, Lighting

Editor

Paulo Santos (EFA)

Contributors

Paulo Viegas, Paulo Santos (EFA)

Nuno Pereira, Miguel Areias (EDP)

Disclaimer

This work was partially supported by the European Commission through the FP7-ICT program under project TClouds, number 257243.

The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose.

The user thereof uses the information at its sole risk and liability. The opinions expressed in this deliverable are those of the authors. They do not necessarily represent the views of all TClouds partners.

Executive Summary

This document presents the design of a Web Application to manage the public lighting in Smart Grids. The chosen architecture consists on the integration of existing low-level management tools interconnected with the Smart Lighting Server, a new component introduced in the following paragraphs. This SL Server will run inside the Trustworthy Cloud and communicates with the existing tools that are located outside in a LAN environment. This communication will be achieved through a gateway (the Smart Lighting Gateway) that will do all necessary translations and will provide data relay functionalities. The SL Gateway and the SL Server will be developed in the scope of this project.

In order to assure a fault and intrusion-tolerant system the data storage will be entirely implemented using the State Machine Replication middleware developed by FCUL [5].

An in-depth look on the overall SL System architecture is given in Chapter 3. Then, Chapter 4 provides a drill-down to each sub-system and a general description of the behaviour of the components and their interactions. Chapter 5 shows the Web Application classes diagrams and explains how the application will be built on top of the Wicket framework. On Chapter 6, the Web Application data model is discussed using an entity relationship model. Chapter 7 describes the interactions between the different components for each Web Application use case. It also describes the services that will be implemented by each component. The user interface mock-ups that serve as a basis for implementing the actual Web Application web pages are presented on Chapter 8. Finally Chapter 9 proposes some functional validation scenarios.

Contents

Chapter 1	Introduction	1
Chapter 2	Scope	2
Chapter 3	Smart Lighting System architecture	3
3.1	EDP Systems	3
3.2	SL Gateway.....	4
3.3	SL Server	4
Chapter 4	Component Model	5
4.1	Web Application	6
4.1.1	Presentation Logic Layer	6
4.1.2	Wicket.....	6
4.1.3	Business Logic Layer.....	6
4.1.4	SMR Proxy	7
4.2	DTC Management.....	7
4.3	CC Bridge.....	7
4.4	SMR - State Machine Replication	7
4.5	EDPSys Gateway.....	7
4.5.1	SMR Gateway.....	8
Chapter 5	Web Application class Diagrams	9
5.1	Smart Lighting unrestricted web pages.....	9
5.2	Smart Lighting restricted web pages.....	10
Chapter 6	Data Model	11
Chapter 7	Dynamic Model	12
7.1	User logs in the SL System	12
7.2	Page Requests.....	13
7.3	User changes SL System configuration	14
7.4	DTC Management poles DTC to synchronize (UC175)	15
7.5	User requests real-time DTC information.....	16
7.6	User manually changes DTC operation	17
7.7	Command Centre sends a new DTC set	18
7.8	Command Centre sends DTC events	19
Chapter 8	User Interaction Model	20
8.1	Login	21
8.2	DTC Schedule Page.....	22
Chapter 9	Validation Scenarios	23
Chapter 10	List of Abbreviations	24
	References	25

List of tables

Table 1: List of Abbreviations24

List of figures

- Figure 1: Smart Lighting System architecture 3
- Figure 2: DTC architecture 4
- Figure 3: Smart Lighting System components 5
- Figure 4: Business Logic Diagram 6
- Figure 5: Smart Lighting unrestricted web pages class diagram 9
- Figure 6: Smart Lighting restricted web page class diagram.....10
- Figure 7: User Login sequence diagram12
- Figure 8: Page request sequence diagram13
- Figure 9: Page change request sequence diagram14
- Figure 10: DTC synchronization sequence diagram15
- Figure 11: User requests real-time DTC information sequence diagram.....16
- Figure 12: User requests manual DTC state/mode change17
- Figure 13: Command Centre sends new DTC set18
- Figure 14: DTC sends an event.....19
- Figure 15: Web page navigation diagram20
- Figure 16: Login page.....21
- Figure 17: DTC Schedule page22

Chapter 1

Introduction

This document describes the design of the Smart Lighting System, introduced in the document D3.2.1 [1]. It also discusses the actual implementation of that system.

Chapter 2

Scope

This document discusses the implementation of the Smart Lighting System Server. This document will not go into detail on the EDP System components that have already been developed; therefore, it will refer only their interface. All data persistency is guaranteed by the State Machine Replication [5], developed by FCUL.

Chapter 3

Smart Lighting System architecture

The Smart Lighting System is comprised by several sub-systems. It will bring together the already existing EDP systems that physically control the public lighting and a new component, the SL Server, which will act as an easy to use interface between the User and those systems. Different kinds of Users will connect via the World Wide Web to the SL Server which, in turn, communicates with EDP systems via the SL Gateway.

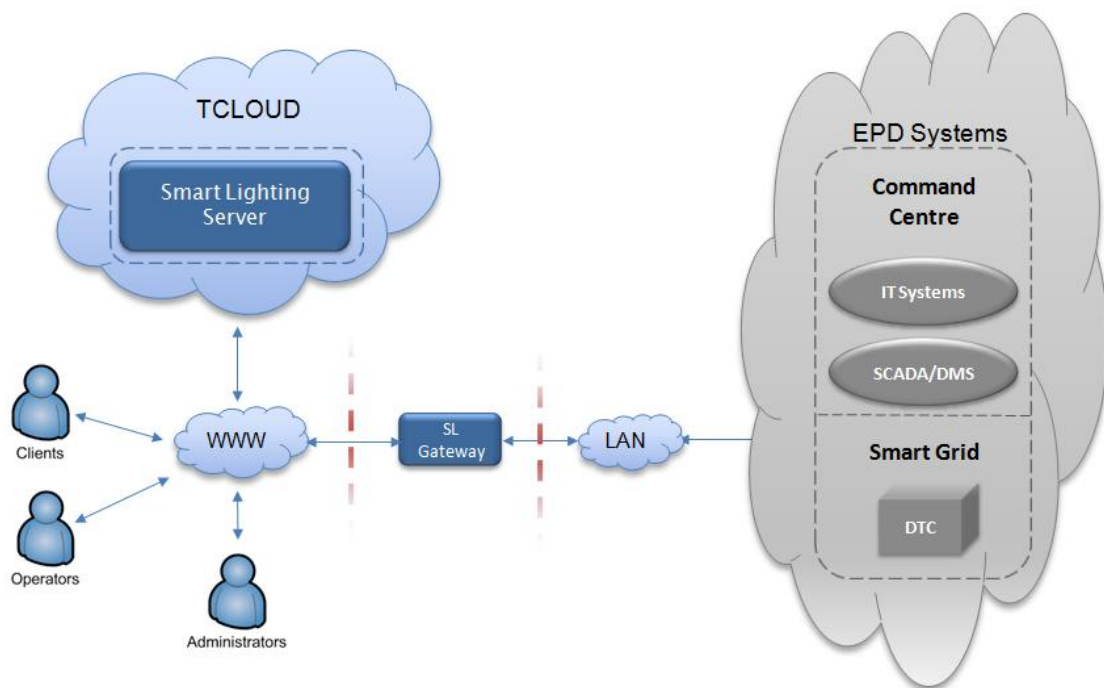


Figure 1: Smart Lighting System architecture

3.1 EDP Systems

These consist of a set of existing tools and components at EDP, which perform the low level management of the Smart Grid components. These tools include:

- IT Systems: systems and applications for management and central data processing, such as applications for the energy metering management and commercial systems);
- SCADA/DMS: systems and applications for the supervision, the control, the optimization and the management of power distribution networks;
- DTC: local control equipment to be installed at switching stations, including modules for measuring, actuation, processing, interface, communication, etc.

Figure 2 depicts the basic DTC architecture.

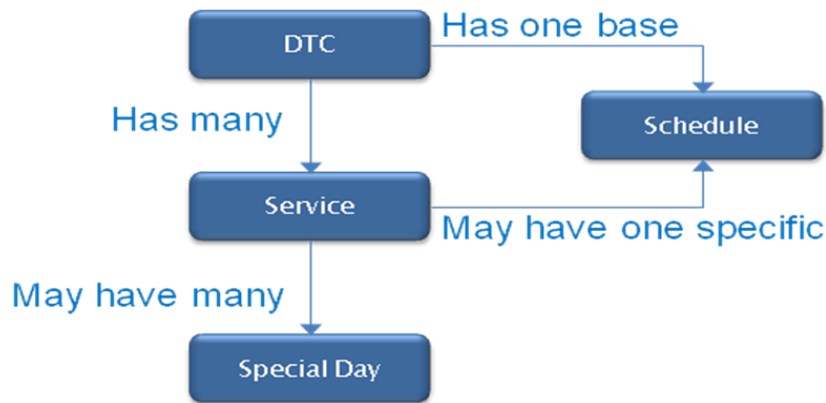


Figure 2: DTC architecture

A DTC has always a base Schedule that is used when no other specification exists. There is at least one Service in each DTC that corresponds to an electrical wiring system, such as street lighting or monuments. Each of these Services may have one specific Schedule that will override the DTC base Schedule. Each Service may have many Special Days. These are days in which the configuration differs from the standard one, such as holidays, or festivities. On these days the setting of the Special Day overrides all other settings.

For more detail on DTC please refer to [1].

3.2 SL Gateway

The SL Gateway will be the bridge between EDP Systems and the SL Server. Indeed, while EDP Systems are designed to operate in a LAN environment and use the proprietary EFACEC BUS communication protocol [9], the SL Server will use the XML-RPC protocol [6]. Thus, the SL Gateway will translate all requests, and consequent replies from the SL Server (in XML-RPC) to the EDP Systems (in EFACEC BUS) and vice-versa.

3.3 SL Server

The Smart Lighting Server is the module of the Smart Lighting System responsible for the control, the management and the configuration of the SL System. It has been designed to run inside the Cloud.

It will provide public lighting management functionalities, like on/off commands, real time status, energy consumption and schedules update, in a user friendly way. Some concepts will be introduced in the SL System in order to ease schedule management and reporting operations. For more details please refer to [1].

Chapter 4

Component Model

In the next figure, it will be presented a view of the Smart Lighting System components.

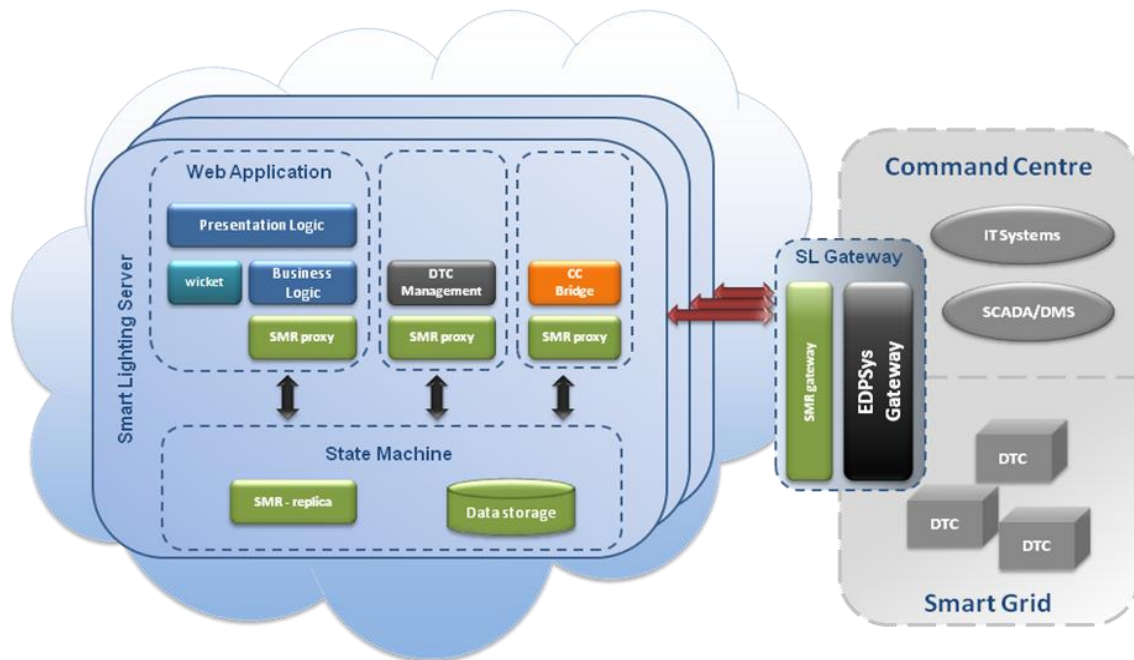


Figure 3: Smart Lighting System components

All data persistency will be assured by the FCUL's State Machine Replication. This will ensure data integrity and availability despite a large spectrum of faults.

The Smart Lighting Server will have 4 main processes:

- State Machine Replica (Keeps the server state)
- Web Application (Stateless)
- DTC Management (Stateless)
- Command Centre Bridge (Stateless)

By keeping the server state only in the SMR, we can assure that the other processes are stateless. Since the SMR ensures that the state is the same in all replicas, no other concerns arise when shutting down instances due to a fault condition or resource crunching.

To facilitate the implementation of the stateless processes, the communication with the SMR will be done using the JPA interface [3] implemented on the SMR proxy.

Outside the Smart Lighting Server there will be a SL Gateway. The Server communicates with the SL Gateway through the SMR gateway. The latter will keep track of the number of SMR replicas that are active at any time, will be responsible for dispatching events from EDP Systems to the SL Server and collecting requests from the SL Server so that replies are sent to the correct instance.

Next, we provide a short description of the SL Server components.

4.1 Web Application

4.1.1 Presentation Logic Layer

This layer of the Web Application is responsible for interacting with the end-User. It implements a Web interface that contains only user interface artefacts (HTML pages, Cascading Style Sheets, Images, JavaScript). Further, it is built on top of the Wicket framework [7] and aims to provide an appealing and intuitive user experience.

4.1.2 Wicket

Wicket [7] is a Java development framework for web-based applications provided by Apache Software Foundation. It is based on XHTML templates for the Presentation Layer which are bounded to Java components at a lower layer. Its extensible architecture and the shipped out-of-the box components with AJAX capabilities allow developing a product in a reasonable time. Since this framework uses XHTML templates, it allows full control over produced HTML.

By default, Wicket saves the state of the session. In order to use stateless operations, sessions must be stored in the SMR.

4.1.3 Business Logic Layer

This layer implements the SL System features. It uses the JPA framework [3] for transparent data storage access and some components are implemented using MDA models [4]. The data model is described in XML files. The MDA engine generates the entity classes (POJO's), the services (the Java classes with methods to interact with the data storage, the Persistent Layer) and the data storage creation scripts.

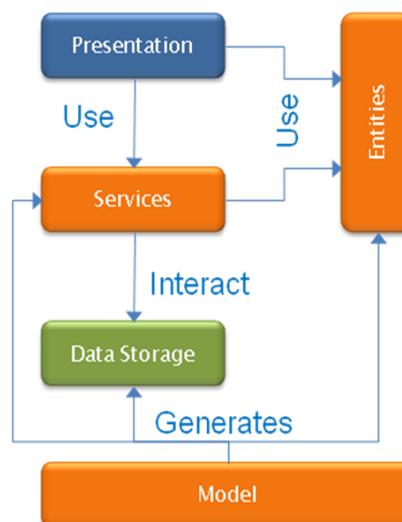


Figure 4: Business Logic Diagram

For each table in the data model the MDA generates an entity that represents each table record and a service that implements every operation possible on that entity. Then, the Presentation Layer uses the generated entities and services to interact with the data storage in a secure way, since only the set of operations defined in the services are allowed.

Looking deeply into Figure 4, the Presentation Layer uses the data entities with read only access, it only shows the entity data. The Services uses the data entities with read and write access. Since the Presentation Layer and the Services implementation reside in the same Java application no Proxy is required to the Presentation Layer use the Services.

4.1.4 SMR Proxy

The SMR Proxy serves as a frontend to interact with the State Machine (backend). It supplies a JPA-like open interface, so that higher-level components abstract from the actual physical implementation. It uses an Object-Relational-Mapping (ORM [8]) approach for interacting with the data storage system. It is also responsible for all SMR communication required, for instance, to keep track of all SMR replicas for broadcasting requests and voting replies.

4.2 DTC Management

DTC Management is the component responsible for DTC communication. It constantly polls the data storage via the SMR proxy and, whenever new data is available, relays them to the DTC through the SL Gateway using the XML-RPC protocol. After closing the communication with each DTC, this component stores the outcome in the data storage using the SMR Proxy.

It is responsible for querying and updating the state of individual DTCs, more specifically the DTC service state (on/off) and the mode (automatic/manual).

It has algorithms for determining if a DTC needs to be synchronized.

4.3 CC Bridge

The Command Centre Bridge deals with data collected from the Command Centre, such as an anomaly status, state transition events and DTC set changes. It receives data from the SL Gateway through the XML-RPC protocol and stores them on the State Machine.

4.4 SMR - State Machine Replication

SMR is a tool to build fault and intrusion-tolerant systems. It is a Byzantine Fault-Tolerant component (BFT). It is safe since all servers serve the requests in the same order and because it ensures that all the correct client requests are executed. It also uses reliable and authenticated point-to-point channels in communication, e.g. through SSL or IPsec. With n instances ($n \geq 3f + 1$), it tolerates at most f faults.

4.5 EDPSys Gateway

EDPSys Gateway is the interface used to route queries from the Smart Lighting Server to the correct EDP system. On the EDP System side the communications are based on EFACEC proprietary RMI/IIOP/SOAP communication framework that uses a publish/subscribe paradigm and has a distributed systems modular architecture which supports Java & C++ components as well as multi-platform (Linux; Windows; HP-UX).

4.5.1 SMR Gateway

The SL Gateway uses the SMR Gateway to keep track of all SMR replicas and to route queries and replies to all of them.

Chapter 5

Web Application class Diagrams

In this chapter, we describe the class diagrams of the Web Application, which is used for creating a typical Smart Lighting web page on top of the Wicket framework [7].

With the Wicket framework, every web page is represented by a Java class that holds all the presentation logic associated with that page and is built using basic components (labels, images, forms, etc...) and/or Smart Lighting complex components, both of which can be reused in other web pages.

There are basically two different kinds of accesses:

- Unrestricted access: they do not require the accessing User to be logged in the SL System – login and error pages;
- Restricted access: the accessing User must be logged in the SL System to display them – pages that show the system internal data or allow the User to interact with the public lighting.

5.1 Smart Lighting unrestricted web pages

This kind of web pages does not require that the accessing User provides the credentials. These pages are extended from the base page that only provides methods for showing internationalized messages to the User. A generic class diagram for such web pages is shown in Figure 5. Each web page will add the necessary attributes and methods to accomplish its goal.

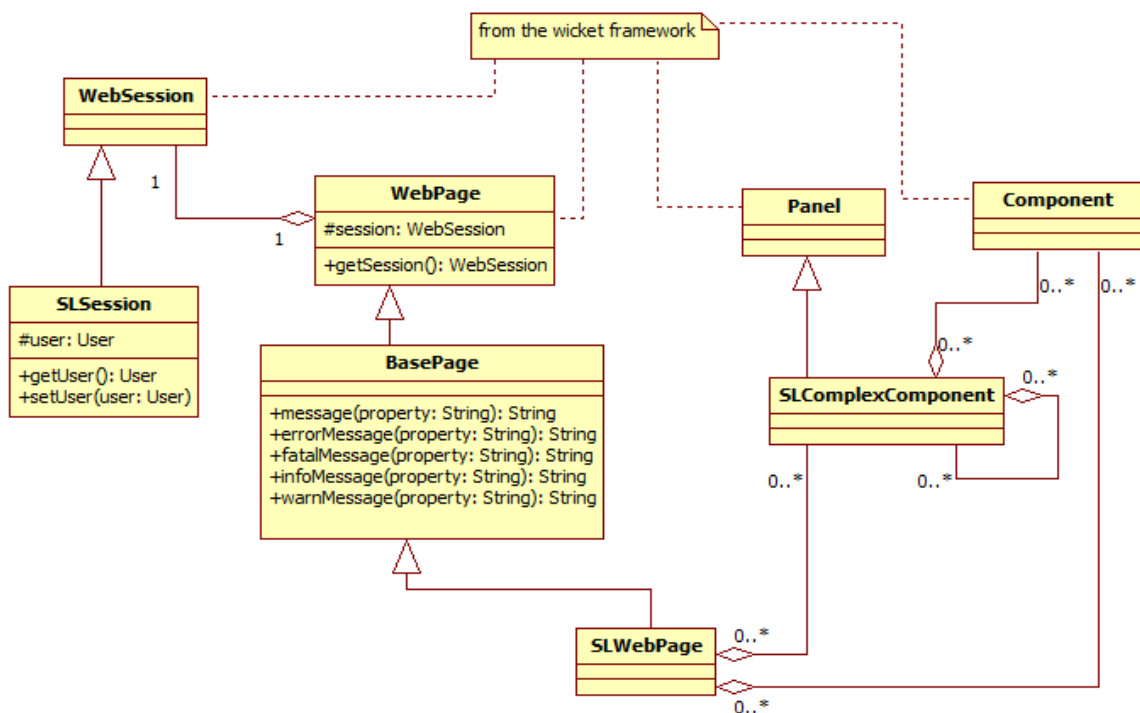


Figure 5: Smart Lighting unrestricted web pages class diagram

5.2 Smart Lighting restricted web pages

These pages require that the accessing User logs in the SL System with valid credentials. All pages of this kind extend the “SLBaseFrame” class. This class assures the User is correctly logged in. In case there are further restrictions, they should be implemented using provided methods to determine the user type. A generic class diagram for such web pages is shown in Figure 6. Each web page will add the necessary attributes and methods to accomplish its goal.

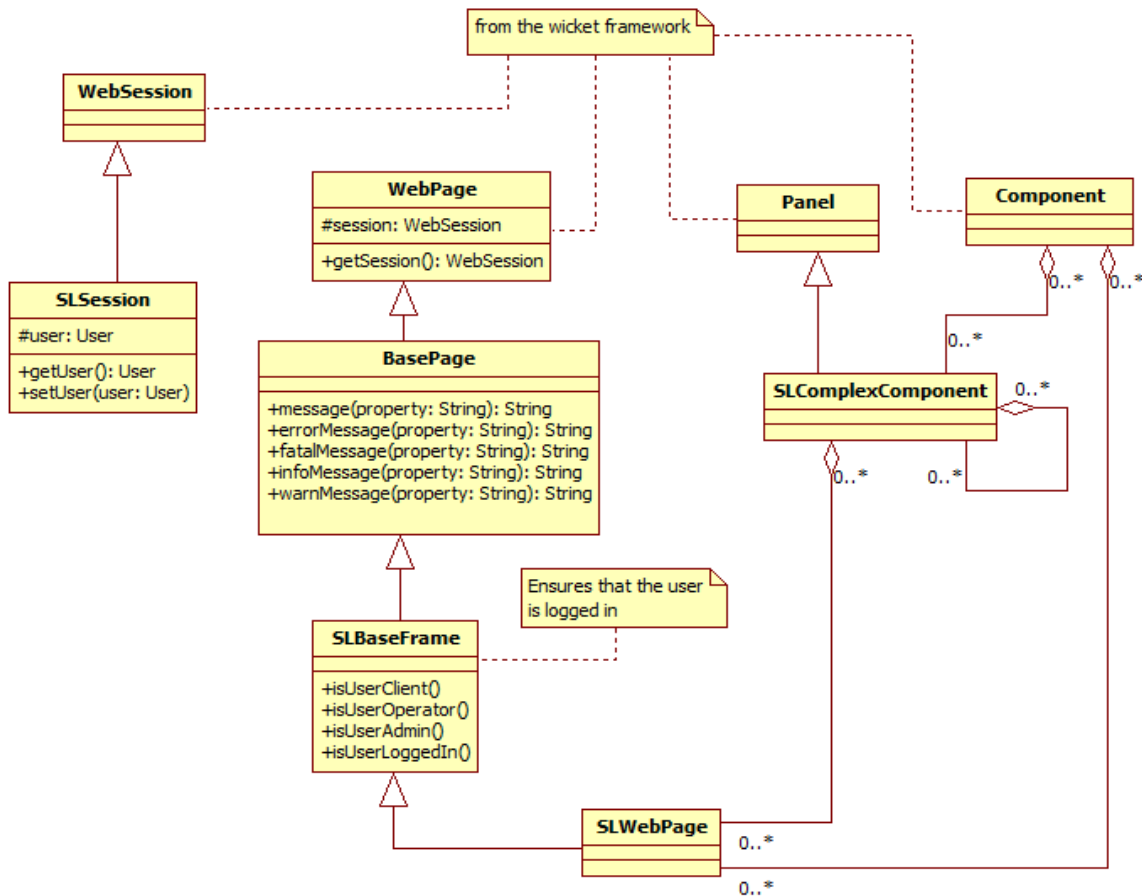


Figure 6: Smart Lighting restricted web page class diagram

Chapter 6

Data Model

The SL Server data model will enable data persistency for the whole SL System. Such data includes business specific data (DTC available features, services and configurations, timetables and alarms) and SL application data (User information, auditing and application configurations).

The detailed description of the data model, with each object definition, can be found at chapter 2 of document D3.2.2 CO Annex [2].

Chapter 7

Dynamic Model

This chapter describes the dynamic model of the Smart Lighting System. All state requests and changes are done through the SMR proxy that abstracts from the number of the running instances, requests order multicast and State Machine responses voting operations [5]. The following diagrams do not show it for sake of simplicity. All state readings are assumed to be non-certified unless otherwise specified.

Whenever appropriate, the sequence and the activity diagrams will refer to the use cases already defined in [1].

In the sequence diagrams, the “get” method is assumed to be a read-only operation, while the “set” method is defined as a write operation returning the entity passed as parameter with its fields updated, for instance the “VERSION” and the “ID” fields.

7.1 User logs in the SL System

Before the User access the Web Application, he must log in the SL System through the login page.

This activity begins when the login button is pressed on the login page. In this page, the User’s credentials are requested via the “username” and “password” text boxes. Both data are mandatory and if they are not provided in the login form, the request must be considered invalid. Once the form is validated, the User’s credentials are checked against the User table in the data storage. If they are not valid, then, the User will be redirected to a page that will inform the login was unsuccessful. If the credentials are valid, instead, the User is redirected to his home page.

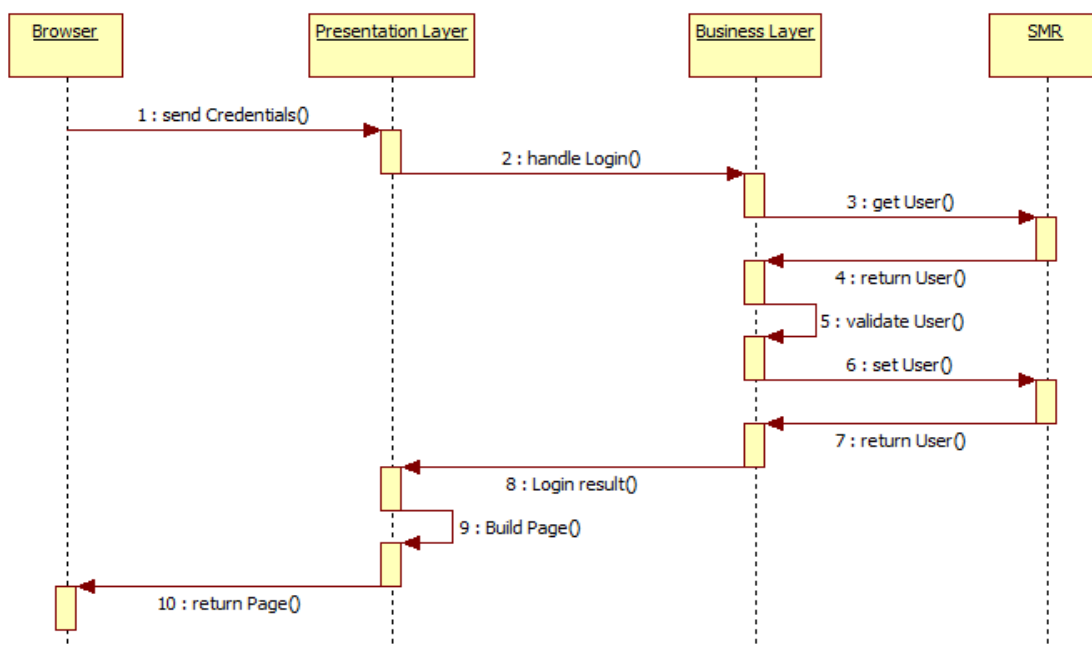


Figure 7: User Login sequence diagram

7.2 Page Requests

The sequence diagram for this activity is shown.

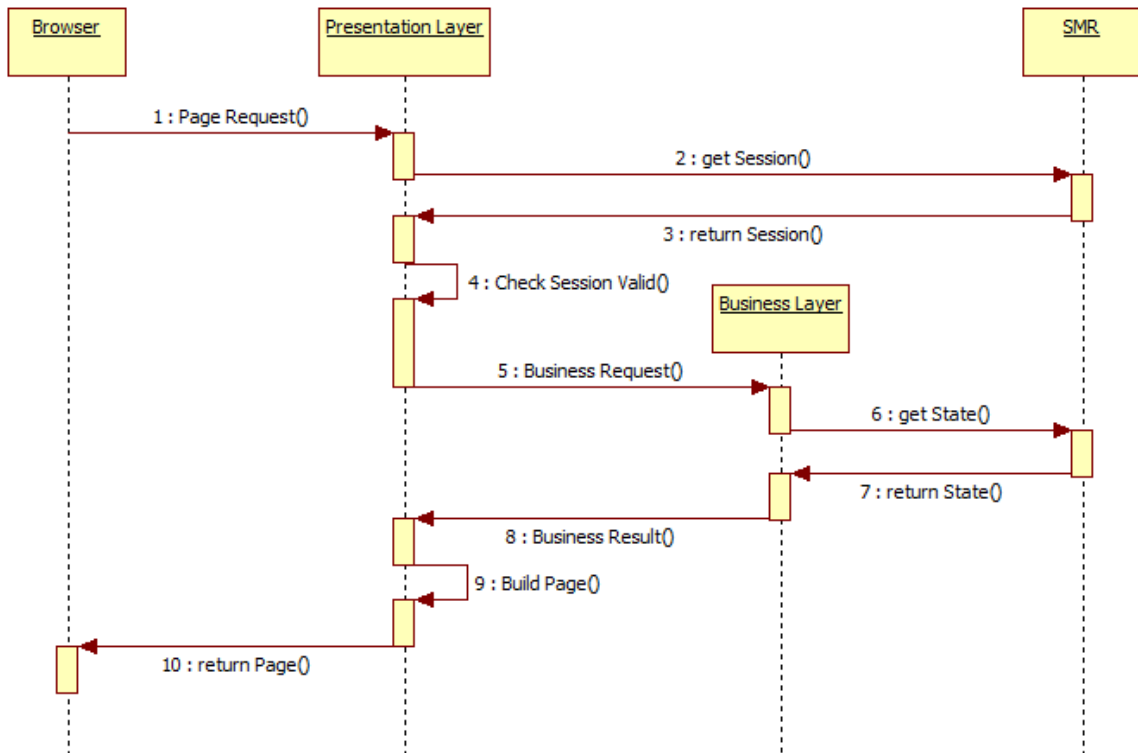


Figure 8: Page request sequence diagram

All business requests are methods provided by a service. Information about the business request for each use case is provided in 3.2 of document D3.2.2 CO Annex [2].

For more information on the checks performed to validate the session please refer to 3.2 of document D3.2.2 CO Annex [2].

7.3 User changes SL System configuration

In case the Presentation Layer receives a request to change the SL System state, the following sequence applies, whether it is a DTC configuration or an application configuration, such as passwords.

Together with every page request the browser sends the session ID. With this ID the Presentation Layer obtains the Session Object (state) from the SMR. The Presentation Layer then checks if the session is valid (for details on this step please refer to 7.2). After validating the session, it forwards the request to the Business Layer. The new entity state is then stored in the SMR and an updated version of that entity is returned. With this result, the Presentation Layer builds the new page that is presented to the User in its browser.

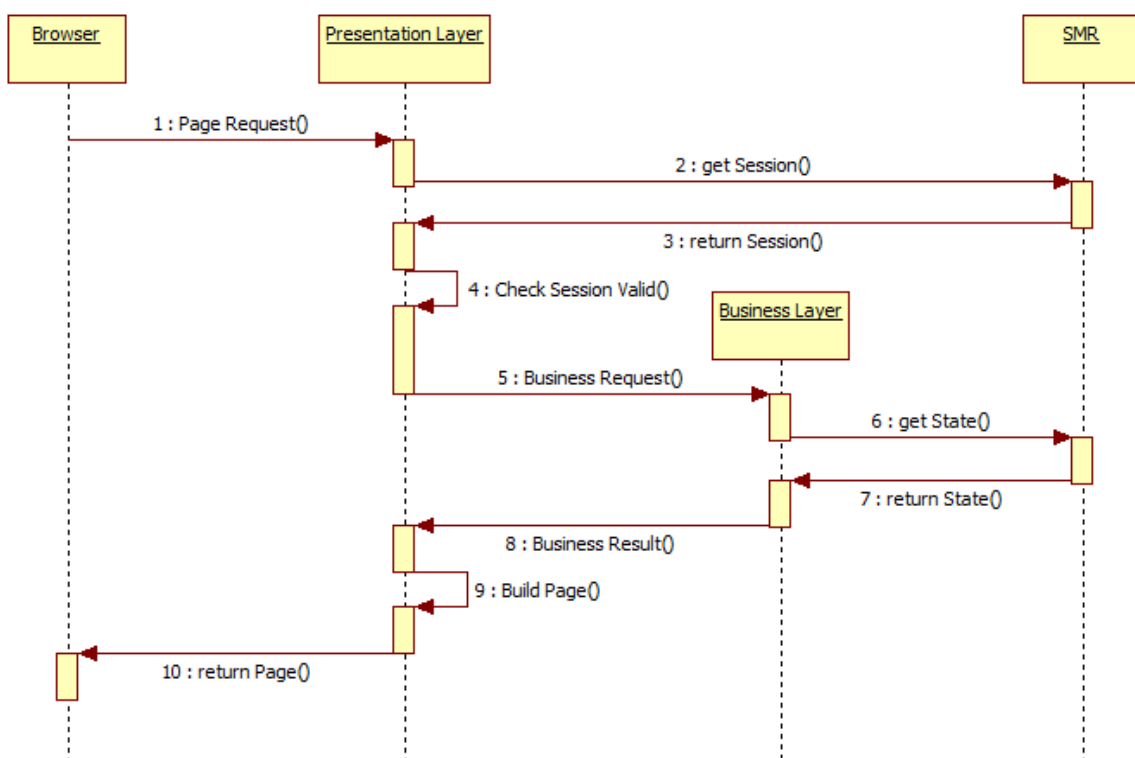


Figure 9: Page change request sequence diagram

For detailed information on the use cases implemented by this sequence diagram please refer to 3.3 of document D3.2.2 CO Annex [2].

7.4 DTC Management poles DTC to synchronize (UC175)

The following sequence takes place periodically at a configurable interval. Step 3 and the following, only take place if the DTC List returned in step 2 was not empty.

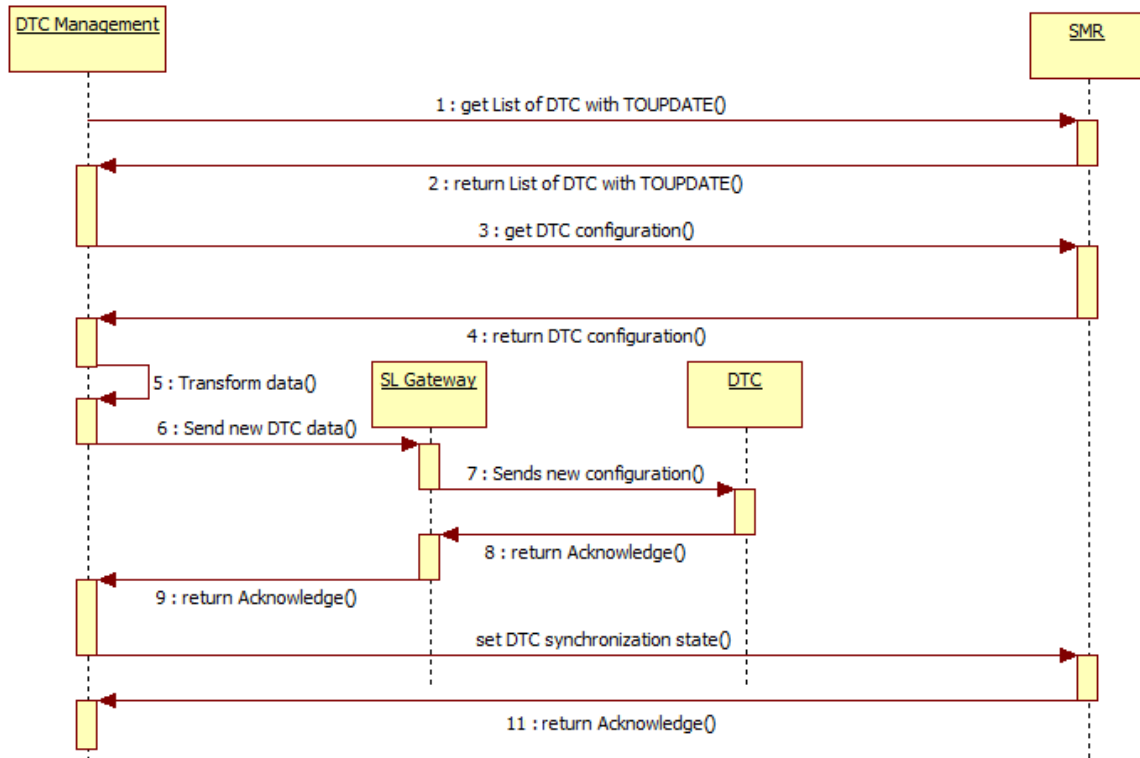


Figure 10: DTC synchronization sequence diagram

After getting a list with the DTCs which require an update, the DTC Management gets the DTCs configurations from the SMR. Then it performs the necessary transformations on these data before sending them to the DTCs through the SL Gateway. After receiving the acknowledgement from a DTC, the DTC Management stores its synchronization status in the SMR. Last step is done for all updated DTCs.

7.5 User requests real-time DTC information

Whenever the User requests near real-time information from a DTC, the following sequence takes place.

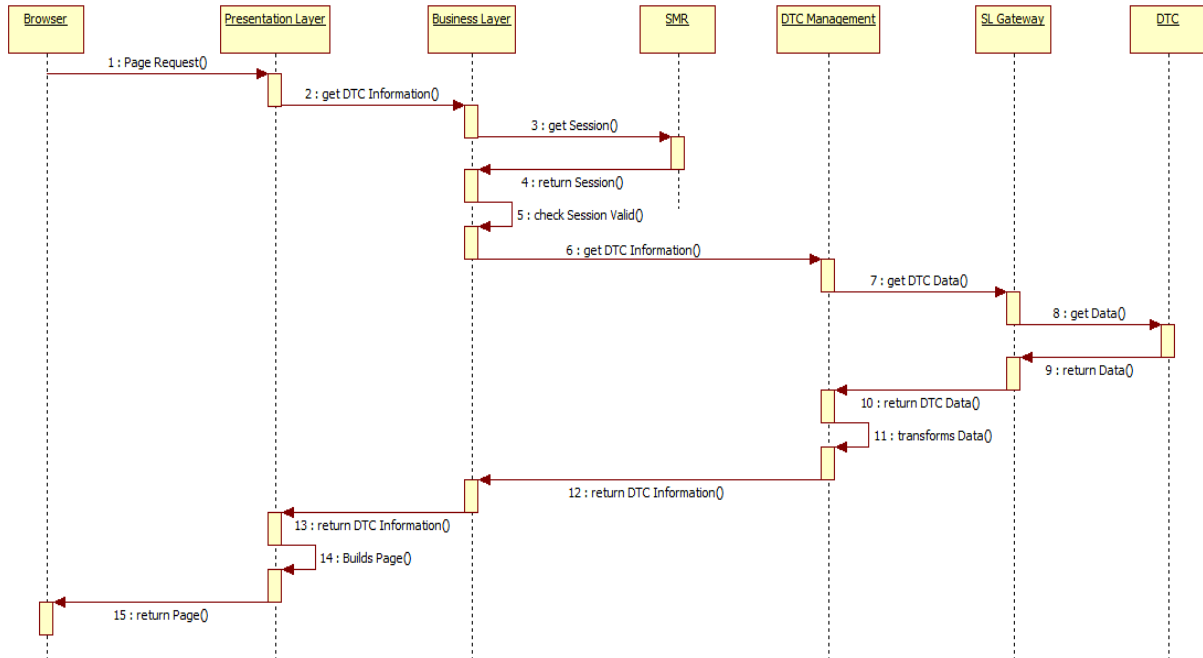


Figure 11: User requests real-time DTC information sequence diagram

For details on the use cases implemented by this sequence diagram please refer to 3.4 of document D3.2.2 CO Annex [2].

7.6 User manually changes DTC operation

When the User requests a manual change in the DTC operation, the following sequence takes place.

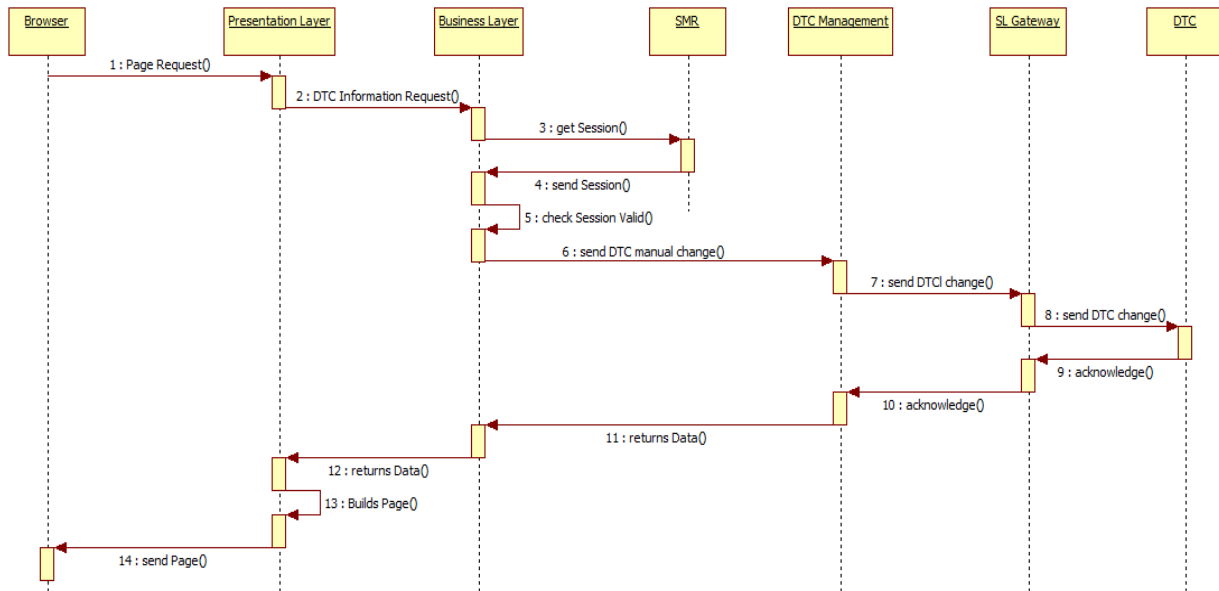


Figure 12: User requests manual DTC state/mode change

This sequence is similar to the one described in 7.5 except that the state is sent to modify a DTC instead of just being requested. It should be also noted that, after the DTC receives its new requested state, in step 8, it performs the change and then, in step 9, returns the current state. The latter may differ from the requested one, for instance when an invalid request was performed. After receiving the DTC state, the Presentation Layer builds the page and returns it to the browser.

For detailed information on the use cases implemented by this sequence diagram please refer to 3.5 of document D3.2.2 CO Annex [2].

7.7 Command Centre sends a new DTC set

Each time the DTC settings are changed, because a DTC is added, changed or removed in the Command Centre, the event must be sent to the Smart Lighting Server via the SL Gateway. This ensures that the whole DTC system is always synchronized with the Command Centre.

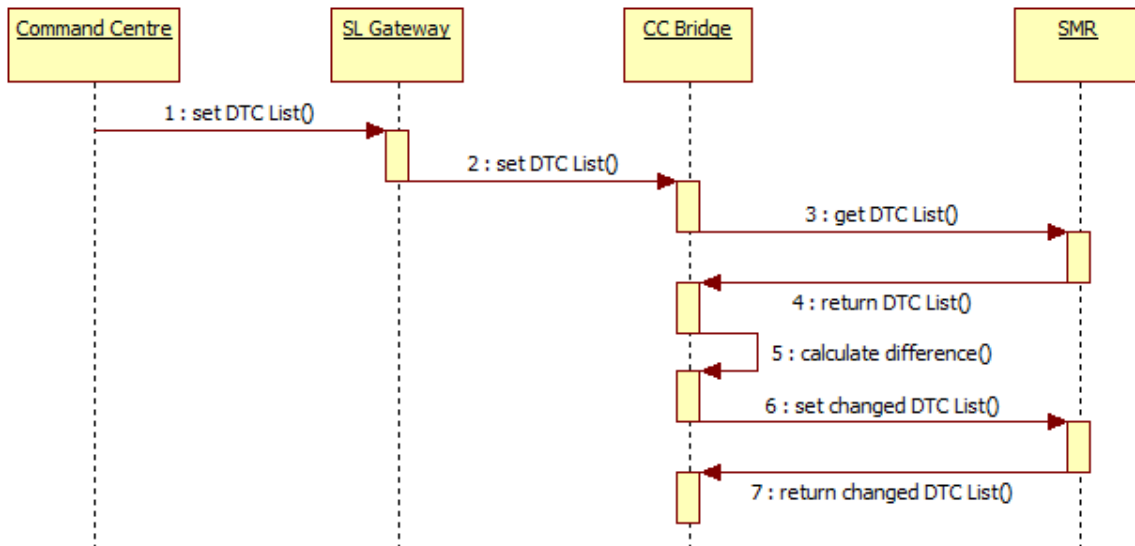


Figure 13: Command Centre sends new DTC set

The Command Centre sends an event with a new DTC List that is relayed to the CC Bridge by the SL Gateway. When the CC Bridge gets the event, it retrieves the current DTC List from the SMR and then calculates the differences in order to minimize the number of write operations on the SMR. After calculating all differences, the CC Bridge stores the modifications in the SMR.

Command Centre events are asynchronous, and do not require a response.

7.8 Command Centre sends DTC events

When a public lighting event is generated in the Command Centre, the following sequence takes place.

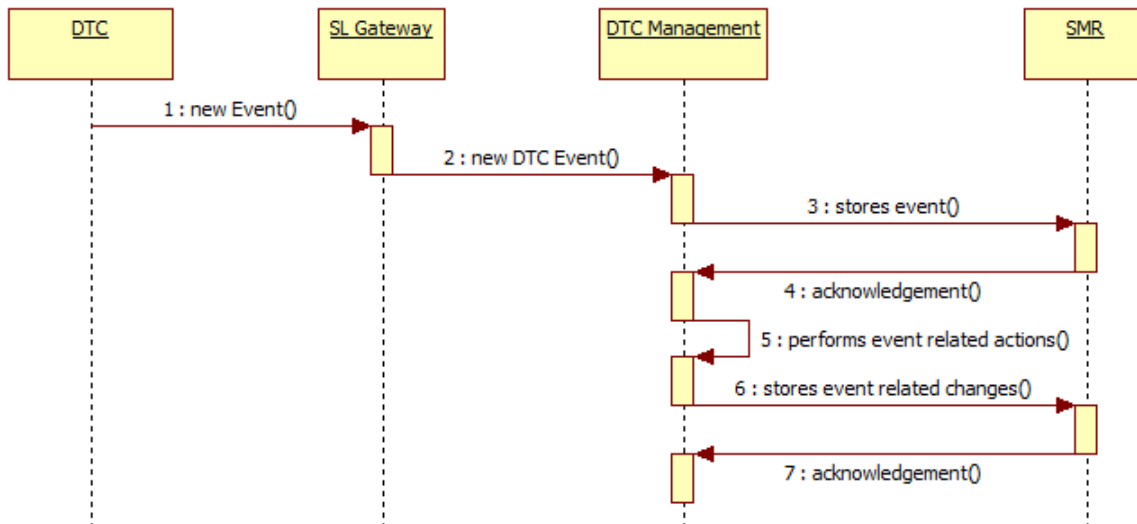


Figure 14: DTC sends an event


When the CC Bridge receives a public lighting event through the SL Gateway, it stores the event in the SMR. Then it processes the event and stores the necessary changes to the SL data model in the SMR. For instance, when a DTC state changed event is received by the CC Bridge, besides storing the event, the CC Bridge also stores the current DTC state in the DTC_SERVICE table.

DTC events are asynchronous, and do not require a response.

Chapter 8

User Interaction Model

This chapter will focus on the user interface to the Web Application. Due to confidentiality issues, only two pages are described here. Instead the documentation of remaining pages is placed in chapter 4 of document D3.2.2 CO Annex [2].

Each page shown to the User is presented alongside the use cases. Several pages have certain areas that are restricted, depending on the user type. Each of the restricted components is identified by a yellow circle with a number inside .

The number key has the following meaning:

1. Only visible to Users with type Operator;
2. Only visible to Users with type Administrator;
3. Only editable when creating, disabled when editing;
4. Only editable by Users with type Operator;
5. Only visible to User with type Operator or Administrator.

The web page navigation diagram is shown in the next figure.

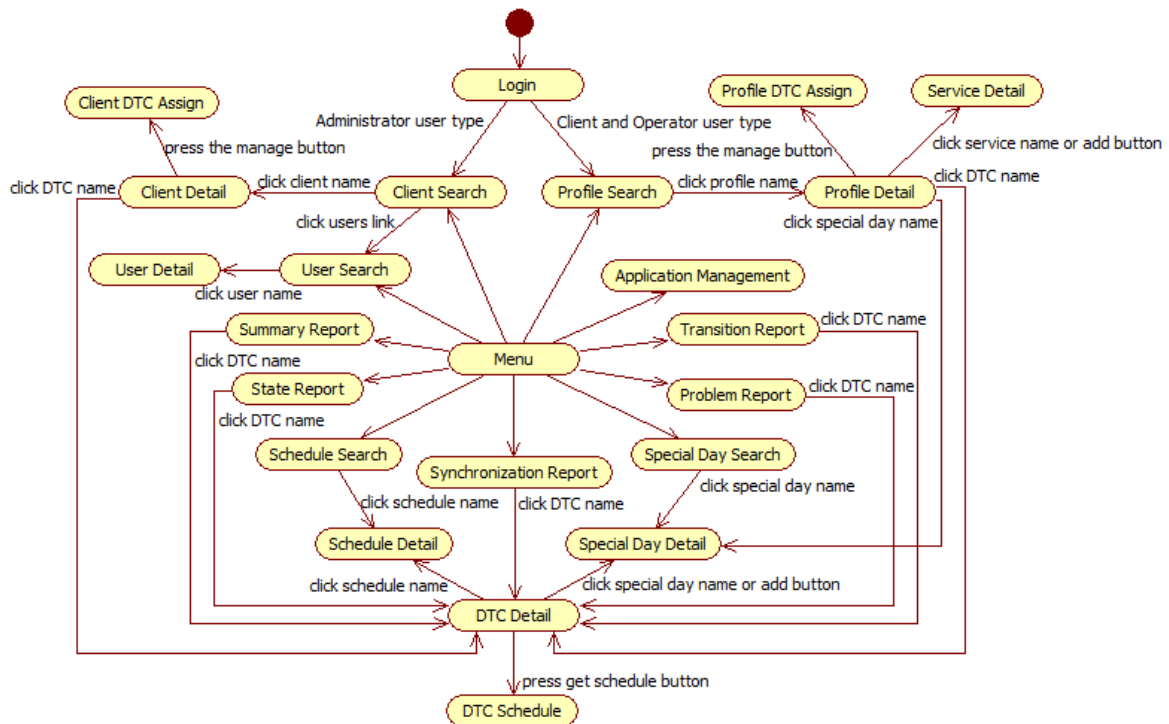


Figure 15: Web page navigation diagram

8.1 Login

This page is shown whenever an unauthenticated User tries to access the application.



Authentication

User:

Password:

[forgot password](#)



Smart Lighting Application version X.X (c) 20XX All rights reserved. Powered by



[contact support](#)

Figure 16: Login page

After the User successfully logs in SL System, he is redirected to his home page according the user type.

If the User has forgotten the password, he can press the “forgot password” link that will show a popup window asking for the User’s e-mail. The provided e-mail is validated and checked against all Users’ e-mails and if it belongs to a User, the password is reset and sent to the provided e-mail address.

8.2 DTC Schedule Page

When the User clicks on the “get schedule” button on the “DTC detail” page (4.6 of document D3.2.2 CO Annex [2]) he is redirected to the DTC Schedule Page. The latter has no editable items.

Figure 17: DTC Schedule page

It presents one tab for each Service. When the User selects a tab, the browser displays below in the page the information of the respective Service.

There is a list of Schedule types and a calendar fixed on the current year that allows the User to get a perspective on how the DTC different Services operate every day. There are three kinds of Schedule on a DTC: "Base", "Service Specific" and "Special Day". The fourth Schedule type ("Consolidated") represents the merging of the other Schedules, taking into account the DTC rules. On the calendar a tooltip is shown when the User hovers the mouse pointer on a day with some kind of configuration. This tooltip provides a textual description of the control of that day.

The legend of calendar colours is displayed on the bottom of the calendar.

This page implements UC180.

Chapter 9

Validation Scenarios

This section contains a set of test scenarios that will be performed with the purpose of validating the Smart Lighting Application. When applicable, the test scenario mentions the use cases that are tested.

For confidentiality reasons this entire section was moved to chapter 5 of document D3.2.2 CO Annex [2].

Chapter 10

List of Abbreviations

DTC	Distribution Transformer Controller
RMI	Remote Method Invocation
DB	Database
POJO	Plain Old Java Object
JPA	Java Persistence API
SMR	State Machine Replication
SL	Smart Lighting
SCADA/DMS	Supervisory Control and Data Acquisition / Distribution Management System
BFT	Byzantine Fault Tolerance

Table 1: List of Abbreviations

References

- [1] Areias, M., Grosinho, M., Viegas, P., Santos, P. & Rodrigues, A.
D3.2.1 - Smart Lighting System Specification.
- [2] Areias, M., Pereira, N., Santos P. & Viegas P.
D3.2.2 CO Annex - Smart Lighting System Design.
- [3] *Java Persistence API (wikipedia).*
http://en.wikipedia.org/wiki/Java_Persistence_API
- [4] *Model-driven Architecture (wikipedia).*
http://en.wikipedia.org/wiki/Model-driven_architecture
- [5] Pasin, M.
D2.2.1 - Preliminary Architecture of Middleware for Adaptive Resilience.
- [6] *XML-RPC.*
<http://www.xmlrpc.com/>
- [7] *Wicket Framework.*
<http://wicket.apache.org/>
- [8] *Object-relational mapping (wikipedia).*
http://en.wikipedia.org/wiki/Object-relational_mapping
- [9] *The BUS Architecture.*
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&arnumber=964941