Energy Management System

*This document contains information that is confidential and proprietary to Schneider Electric. This document including any excerpt hereof, may not be copied, transmitted, distributed or otherwise communicated to any third party without the express written consent of Schneider Electric.*

*This document has undergone extensive technical review before being released. While every care has been taken in preparing these documents in order to keep the information herein as accurate and up to date as possible, neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein, nor for errors or omissions or for damages resulting from the use of the information contained herein.*

*The content of this document is subject to change without prior notice.*

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Authors | Date | Comments |
| 1.0 | Marko Jelaca | 1-Dec-17 | *Inicijalna verzija* |

| Responsibility | Role | Name & Surname |
| --- | --- | --- |
| Chief of Architects |  |  |
| QA |  |  |
| Deployment |  |  |
| System/SW Architect |  |  |
| Release Manager |  |  |
| Test manager |  |  |
| Technical Leader |  |  |
| Solution Architect |  |  |

Final review and approval of the latest version:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Date | Name & Surname | Business Role | Signature |
| Reviewer |  |  |  |  |
| Approver |  |  |  |  |

Table of Contents

[1. Definitions and Abbreviations 2](#_Toc441752415)

[2. References 2](#_Toc441752416)

[3. Abstract 2](#_Toc441752417)

[4. Overview 2](#_Toc441752418)

[4.1. Functional Design 2](#_Toc441752419)

[4.2. Architectural Design 2](#_Toc441752420)

[5. Detailed Design 2](#_Toc441752421)

[5.1. Component Design 2](#_Toc441752422)

[5.2. Data Design 2](#_Toc441752423)

[5.3. UI Design 2](#_Toc441752424)

[6. Additional Considerations 2](#_Toc441752425)

[6.1. Quality and Testing 2](#_Toc441752426)

[6.2. Deployment and Configuration 2](#_Toc441752427)

[6.3. Upgradability 2](#_Toc441752428)

[6.4. Extensibility 2](#_Toc441752429)

[6.5. Security 2](#_Toc441752430)

[7. Appendix 2](#_Toc441752431)

Table of Figures

# Definitions and Abbreviations

|  |  |
| --- | --- |
| Definition/Abbreviation | Description |
|  |  |
|  |  |

# References

|  |  |  |
| --- | --- | --- |
| # | Title | Description |
| 1. |  |  |
|  |  |  |

# Abstract

<This section should be brief and clear. It should describe the scope of the document in no more than 1-2 paragraphs and motivation or reasoning why this document was written.>

Ovaj dokument predstavlja projektu dokumentaciju za projekat tima P4 EMS. Zadatak je implementacija sistema upravljanja energijom.

# Overview

<This should be no more than 1-2 pages where the high level requirements that this architectural document addresses are stated (with the link to the detailed requirements document/user story if exists) and where the high level design is introduced. One global diagram of the relevant architecture is almost a must in this section.>

## Functional Design

<High level description of the functionality – consult relevant Product Owner and/or Business Analyst.>

## Architectural Design

<High level design with main modules, interfaces between them and to the outside world plus main data flows.>

# Detailed Design

<This is where the design is explained in detailed. The text should be accompanied with block and UML diagrams where appropriate. Remember that design is best explained with diagrams not just pure text! This chapter should also explain how are requirements given as the starting point for the design and introduced in the Design Overview chapter addressed. Depending on the design scope several sub chapters can be introduced. Most of the following subchapters (but not limited to) should be addressed in every design document, depending on the subject or the scope of the design.>

## Component Design

## 

Slika ‑. UML dijagram komponenti

<Refinement of the architectural design – describe components (decompose modules from the architectural design), their scope and describe interactions between components. Each component should be described in following terms:

### NetworkModelService komponenta

***Definition***

Describe the specific purpose of the component.

***Responsibilities***

Network Model Service (skr. NMS) komponenta ima zadatak da skladisti staticke podatke modela.

***Constraints***

Give any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component.

***Composition***

NetworkModelService komponenta sadrzi implementaciju Generic Data Access (skr. GDA) interfejsa koji omogucava komunikaciju sa drugim komponentama. Pored toga sadrzi sam servis koji omogucava skladistenje podataka.

***Interactions***

NMS komponenta komunicira sa User Interface (skr. UI) komponentom i sa Calculation Engine (skr. CE) komponentom. Obe komponente (UI i CE) koriste NMS kako bi dobili staticke podatke modela.

***Interface/Exports***

NMS komponenta pruza GDA interfejs za komunikaciju sa ostalim komponentama. GDA pruza mogucnost rada sa podacima bez potrebe da se zna njihova interna struktura, nacin skladistenja i slicno.

Metode koje pruza GDA interfejs:

UpdateResult ApplyUpdate(Delta delta);

ResourceDescription GetValues(long resourceId, List<ModelCode> propIds);

int GetExtentValues(ModelCode entityType, List<ModelCode> propIds);

int GetRelatedValues(long source, List<ModelCode> propIds, Association association);

List<ResourceDescription> IteratorNext(int n, int id);

bool IteratorRewind(int id);

int IteratorResourcesTotal(int id);

int IteratorResourcesLeft(int id);

bool IteratorClose(int id);

**Detailed Subsystem Design**

Provide a detailed description of this software component. Complex diagrams showing the details of component structure, behavior, or information/control flow may be included.>

### Calculation Engine komponenta

**Definition**

Describe the specific purpose of the component.

**Responsibilities**

CE komponenta ima zadatak da izvrsi optimizaciju na osnovu definisanog algoritma.

**Constraints**

Give any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component.

**Composition**

Provide description of the use and meaning of the subcomponents that are a part of this component.

**Interactions**

CE komponenta komunicira sa SCADA komponentom, NMS komponentom i UI komponentom. Od SCADA komponente dobija trenutna merenja sa simulatora, a od NMS komponente podatke o modelu koje koristi u procesu optimizacije. Izracunate vrednosti optimizacije prosledjuje na UI, i vraca SCADI.

**Interface/Exports**

Pruza implementaciju ISubscribe interfejsa za subscribe UI komponente na topic (rezultat optimizacije) i IScadaCe interfejs za dobijanje podataka sa SCADA komponente. CE komponenta koristi IGenericDataAccess interfejs kako bi dobila staticke podatke modela od NMS komponente.

**Detailed Subsystem Design**

Provide a detailed description of this software component. Complex diagrams showing the details of component structure, behavior, or information/control flow may be included.>

### SCADA komponenta

**Definition**

Describe the specific purpose of the component.

**Responsibilities**

Describe the primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients?

**Constraints**

Give any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component.

**Composition**

Provide description of the use and meaning of the subcomponents that are a part of this component.

**Interactions**

Provide description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use?

**Interface/Exports**

Describe the set of services (resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing meanings of values, parameters, etc.

**Detailed Subsystem Design**

Provide a detailed description of this software component. Complex diagrams showing the details of component structure, behavior, or information/control flow may be included.>

### Simulator komponenta

**Definition**

Describe the specific purpose of the component.

**Responsibilities**

Describe the primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients?

**Constraints**

Give any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component.

**Composition**

Provide description of the use and meaning of the subcomponents that are a part of this component.

**Interactions**

Provide description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use?

**Interface/Exports**

Describe the set of services (resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing meanings of values, parameters, etc.

**Detailed Subsystem Design**

Provide a detailed description of this software component. Complex diagrams showing the details of component structure, behavior, or information/control flow may be included.>

### Adapter komponenta

**Definition**

Describe the specific purpose of the component.

**Responsibilities**

Adapter komponenta ima zadatak da kreira inicijalni delta objekat na osnovu CIM-XML fajla i prosledi ga NMS komponenti.

**Constraints**

Give any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component.

**Composition**

Provide description of the use and meaning of the subcomponents that are a part of this component.

**Interactions**

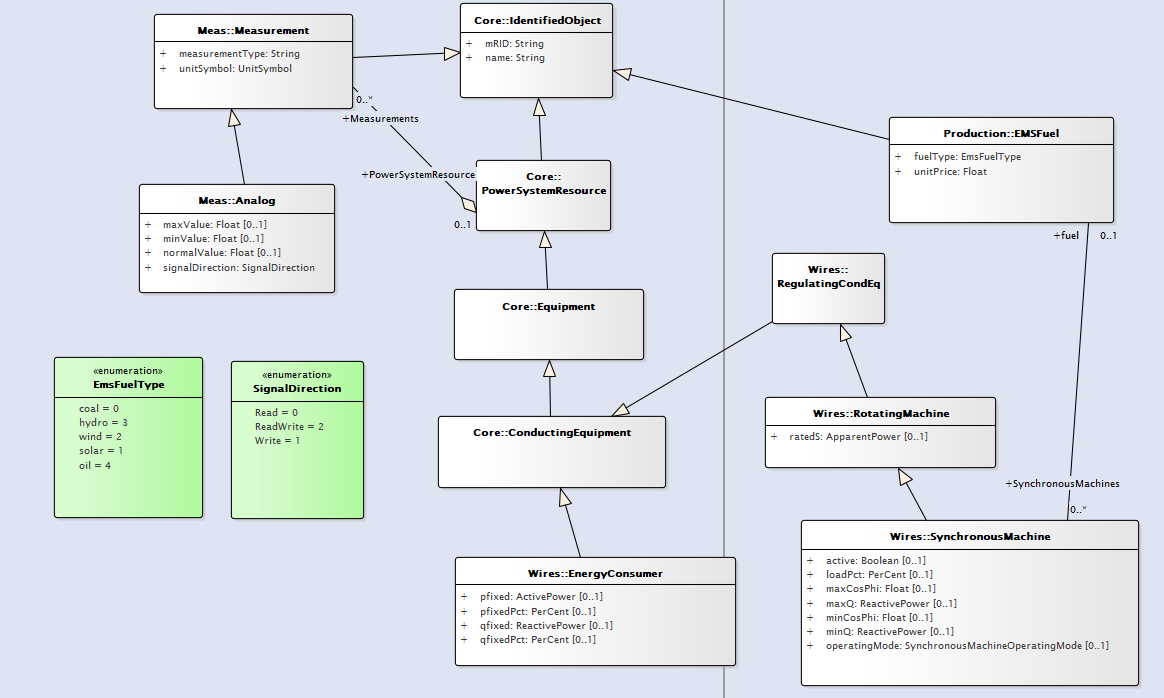
Komunicira sa NMS komponentom kako bi joj prosledila podatke na osnovu CIM-XML fajla.

**Interface/Exports**

Koristi IGenericDataAccess interfejs za slanje podataka na NMS.

## Data Design

Na slici 5.2.1. predstavljen je UML dijagram modela projekta. Klase modela su sastavni deo IEC61970 CIM modela. Klase *Analog*, *EnergyConsumer*, *EMSFuel* i *SynchronousMachine* poseduju svoju konkretizaciju, dok su ostale klase modela apstraktne. Nabrojane klase i njihovi atributi opisani su u daljem tekstu.



*Slika 5.2.1. UML dijagram modela*

Klasa *IdentifiedObject*, paket *Core*

* *mRID*
* *name*

Klasa *Measurement*, paket *Meas*

* measurementType
* unitSymbol

Klasa *Analog,* paket *Meas*

* maxValue
* minValue
* normalValue
* signalDirection

Klasa *EnergyConsumer*, paket *Wires*

* pFixed
* pFixedPct
* qFixed
* qFixedPct

Klasa *RotatingMachine*, paket *Wires*

* ratedS

Klasa *SynchronousMachine*, paket *Wires*

* Active
* loadPct
* maxCosPhi
* maxQ
* minCosPhi
* minQ
* operatingMode

Klasa *EMSFuel*, paket *Production*

* fuelType
* unitPrice

Enumeracija *EmsFuelType* opisuje tip goriva koje koristi određeni generator.

Enumeracija *SignalDirection* opisuje status pisanja ili čitanja određenog podatka.

## UI Design

<Describe the User Interface design. Provide mock-ups where appropriate (engage UI designers).>

# Additional Considerations

<Every design document should cover the following areas: Quality and testing, Deployment, Configuration, Upgradability, Extensibility and Security. Generalized statement/paragraph is expected in the body of this chapter (in this exact area) while the details on each subject are to be given in the sub chapters below.>

## Quality and Testing

<This sub chapter should give answers to the following questions:

What existing features are affected by changes (regression area)? What should the test engineer pay attention to when creating test suite and individual test cases? Are there any aspects where extra care is needed when devising tests?

What should be the integration testing strategy? What should be the test bed? Are there any existing tools that can aid in testing, for example to create inputs? Should some tools be developed for this purpose? Recommended minimal system for integration testing should be given.

Are there any special HW/SW/Environment requirements for integration and system testing (having in mind both types of deployment: MyADMS (MyAGMS) and Enterprise)? List those and provide alternatives if some are impossible or hard to attain.

What are the expected non-functional requirements (e.g. performance, availability and sizing requirements) for newly added and/or changed features/components? How are performances of existing features impacted?>

## Deployment and Configuration

<This sub chapter should give answers to the following questions:

Does this design imply changes in the configuration of ADMS (AGMS)? List those and if possible suggest default values.

Is installation packaging affected by this design? If yes, what needs to change?

Are there any post install steps that need to be documented and exercised after each installation? List those and provide explanation.

All questions should be considered and answered for both MyADMS (MyAGMS) and Enterprise editions of ADMS (AGMS).>

## Upgradability

<This sub chapter should assess the impact of this design on the ability to upgrade existing ADMS (AGMS) systems. Following questions should be answered:

Is there any impact on the upgradability of ADMS (AGMS)? If yes, explain what impacts are there and explain how they are addressed.

Please note that the upgradability is affected if one of the following is being done:

- Interfaces declared public are being changed (are we still supporting the old interfaces?)

- Data model which is exposed through public interfaces is changed (are we sill supporting old data model as input and are we providing it as output on the legacy interfaces?)

- Configuration changes have been made (are they addressed during the install/upgrade procedures?)

- Persistence store model (smart caches, SQL databases, flat files …) has changed (are we providing conversion methods? How?)>

## Extensibility

<This sub chapter should give answers to the following questions:

Have considerations been made to the extensibility of the software modules that are subject of this design? Public interfaces including data model have to be identified and then documented during the development process.

Connected to the upgradability sub chapter above, have existing interfaces been changed and are we keeping backwards compatibility?>

## Security

<This sub chapter should address security aspects of the design if any in coordination with software security engineer/team. Following questions should be answered accompanied with the threat model where appropriate:

What are the security risks identified with this feature’s requirements? How the security risk should be managed: reduced, transferred, avoided or accepted?

What threats have been identified? What are the mitigation techniques that should be applied? What impact the threats have on the overall security and what is the probability of those threats being realized?

How to deploy the feature or function in a secure fashion? What application, operating system and network configuration needs to be in place? E.g.:

• Is there a specific firewall rule that needs to be added?

Are there any security rights (group policies) that need to be associated with the user or computer running the feature?>

# Appendix