UPPSALA
UNIVERSITET

f14031

Examensarbete 30 hp
Augusti 2014

Power System Software Development

with Possible SCADA System Integration

Filiph Appelgren

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

A Power System Software Development with possible SCADA System Integration

Filiph Appelgren

In order for power system operators (such as TRANSCO) to maintain and operate the transmission grid in a safe, secure and effcient way, automatic tools such as SCADA systems is necessary to meet demand at all times. The main purpose of this thesis was to develop a software/prototype at the load despatch centre at TRANSCO with the ability to monitor and communicate with power plants. The software is supposed to work in a real-time electronic market. The power plant operators can declare their availability and capability parameters of their generating and producing units to TRANSCO and LDC operators can send load despatch instructions to the power plants (such as load changes and other ancillary instructions). The prototype also has a compliance monitoring application that validates unit outputs against despatched instructions. If the output is outside a specific interval, a transgression warning is sent to the power plant informing them that they should adjust their unit output against the target load. If further transgression is continued, the operator at LDC can re-declare the units availability on the power plants behalf and issue a new load despatch instruction to the unit. The re-declared availability level will be valid all the way back to when the unit issued the last availability declaration.

The software was successfully developed and could perform all tasks that it was supposed to in a satisfactory way. In order to make the development as sufficient and effective as possible, a "dummy" power plant was created and was used to simulate unit outputs and plant operator behaviour. As the time was too short, the SCADA integration was never investigated and was left to whom is taken over after this thesis has ended.

Handledare: Kamal Radi
Ämnesgranskare: Juan De Santiago
Examinator: Tomas Nyberg
ISSN: 1401-5757, UPTEC F14 031

Contents

List of Figures

Chapter 1

Introduction

TRANSCO is a subsidiary of ADWEA and is in charge of operating the high voltage power and bulk water transmission grid within the Emirate of Abu Dhabi. The LDC is a subdivision at TRANSCO that controls and monitors all power generation and water production units connected to the grid in order to anticipate and meet demand at all times.

In this document, both power generation and water production is considered but you may notice only power generation references. This is due to the fact that these two processes are almost equal and the despatch process is carried out the same. So unless stated otherwise, power generation includes water production.

1.1 Background

The research and development within the area of controlling and operating power systems is a very popular field and a significant progress has taken place in the last decades. A few years ago, TRANSCO launched a program for developing an automated high technology system that operates and controls the power system with a human interaction held to a minimum. This system should handle everything between calculating instruction schedules based on future demand and plant parameter declarations to creating invoices to distribution companies. This collection of various systems brought together by different companies was named the DSSF (Despatch and Settlement Special Facilities). The DSSF was a first of its nature and seemed very elegant and highly efficient in theory but as it turned out, the actual implementation was harder and far more complex than anticipated. To make different systems communicate and operate side by side is a highly difficult task and requires qualified personnel with enough funds set aside.

1.2 Aim Of Thesis

Due to some obstacles the DSSF is not working as anticipated. This raised a thought of a new DSSF system were the functionality of EDL, see section 2.1.3, could be integrated within the already existing and fully operating SCADA (Network Manager) system. If this could be accomplished, a lot of trouble would be minimized, e.g. the maintenance cost is reduced due to the fact that a whole

system is removed and a smaller risk for crashes and communication failures since no information is needed to be transferred between Network Manager and EDL. It is also a relief for the operators since there are fewer interfaces that needs to be handled and correctly understood. This new DSSF system would thus circumvent a major block and become more compact, user friendly, reliable and robust.

The aim of the thesis is hence to create a program, a prototype, which can perform some of the tasks that the EDL could perform. Some of these applications is

- Send load despatch instructions to power plants
- Send ancillary service requests (such as run on back-up fuel, put on AGC and frequency sensitive unit etc.)
- Validate these instructions against declared unit parameters
- Receive unit declarations e.g. availability declarations
- Keep track of all sent instructions and received declarations by logging all data in a database
- Perform compliance monitoring

The first aim of this thesis is to create a fully functioning prototype that operates in real-time with the above described functionalities except the compliance monitoring part and create a corresponding graphical user interface which allows an operator to interact with the program. The prototype should also comprise a power plant client program that acts as the counterpart of this main prototype (at this point there are so called client EDL software's installed at each power plant to whom the main EDL at LDC is communicating). The second aim is to incorporate the compliance monitoring application within this prototype. This part is perhaps the most important and sought tool since this will aid TRANSCO in settlement disputes with the generating companies. Although these two aims are difficult and time consuming to reach I will strive to achieve the third aim which is to investigate an integration of this prototype into the existing SCADA system and if possible, to make an actual integration.

Chapter 2

Theory

The DSSF system is a state-of-the-art technology in automation, monitoring, control and optimization. It goes beyond traditional despatch processes when applied to a constrained environment of power and water in deregulated privatized markets rather than a centrally planned environment. Since the demand for more green power such as solar and wind increases, another dimension of complexity will be added to the overall operating and control process since these types of power generation is inflexible and non-dispatchable. This mean that old processes with manual load instructions will become more and more cumbersome and non-efficient both in an economical and environmental aspect. A DSSF system would take care of all this. The DSSF system is a collection of a few smaller systems that together performs the overall despatch concept. In this section we will look through each of these systems individually to get a good understanding on how everything works. In figure 2.1 a picture of the overall DSSF system can be seen.

In order to build a prototype that could be integrated in the Network Manager and replace some applications, a good knowledge and understanding of all ingoing DSSF systems that communicates with EDL is essential. Hence a first step is to learn how the other systems operate, why and when they perform certain tasks, how they communicate and to whom. A term called ODC will be mentioned (the Overall Despatch Concept) which elaborate on how instructions are sent to units and their response back i.e. the daily planning and communication between LDC and power plants. Information about the EDL system was found in [1].

2.1 DSSF System

2.1.1 Network Manager

The network manager is the operating SCADA system and performs automated despatch instructions to units who are under AGC, controls and monitors the power system and provide other components in the DSSF system with grid feedback. The feedback include real-time power plant generation values among other data. This information is necessary for example EDL to perform compliance monitoring and for Unit Commitment (see section 2.1.2) to initiate the planning process and to calculate new estimated generation schedules.

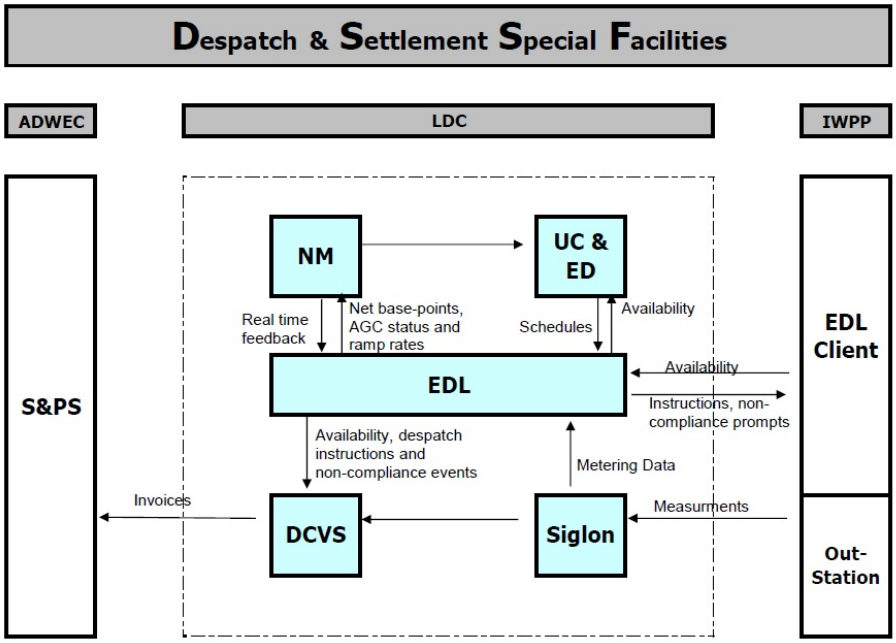


Figure 2.1: The DSSF system with all its containing sub-systems. The arrows show communication data flow between the systems and a small abbreviation of what the flow consists of. The EDL Client and the Out-Station is located at individual water and power producers (IWPP's) while NM, UC, EDL, DCVS and Siglon is located at LDC premises and S&PS at ADWEC premises.

2.1.2 Unit Commitment

The UC can operate in either a short term mode (48 hours ahead) or a very short term mode (4 hours ahead). In these modes, the UC calculates and optimizes every unit's power generation and water production outputs for the specified time frame. Hence the output from UC is an optimized power generation and water production schedule based on technical and economic factors. UC takes availability declarations and capability parameters from EDL as unit conditions as well as estimated power demand and other economic factors when performing a new optimization run. This schedule is then transmitted to EDL for validation and possible manual changes.

2.1.3 Electronic Despatch and Logging System

EDL is an electronic despatch and logging system and is the central hub that communicates with almost every system inside the DSSF. It is the main link between the despatch process and the settlement process (which is the base of the so called state-of-the-art system where instructions and declarations is automatically transferred between the main functional components of the DSSF). The main EDL is located at LDC premises while a smaller version, also called client EDL, is located at each power plant. These EDL applications can communicate with each other and hence form a medium for power plant operators

and LDC operators to exchange information in an efficient and easy way. This link makes it possible for power plant operators to declare their availability[1] and capability[2] of their generating units to LDC at TRANSCO. It also creates electronic messages generated either by manual instructions or inherited by load changes from the UC schedule and to which all units must comply. A very important task that EDL supports is the compliance monitoring which means that it monitors the output of each generating or producing unit and validates them against their load despatch instructions. The EDL transmits availability declarations to UC, non-compliance events to DCVS for settlement purposes and net base-points to Network Manager around which it performs AGC. Some of the applications that the main EDL system comprises are:

- Day-ahead pre-despatch schedule
- Generate and send instructions (either manually or based on pre-generated UC schedule)
- Validate instructions against units declared parameters
- Receive availability and capability declarations
- Compliance monitoring of all power generating and water producing units
- Commercial and operational event logging

2.1.4 Instation (Siglon)

This is the system that collects and processes metered data which is actual produced net electrical power output at each defined point of generation [2]. It utilizes the optical terminal equipment at LDC to distribute the information to relevant systems within the DSSF.

2.1.5 Data Collection and Verification System

The DCVS collects necessary data that is required for settlement and planning purposes as described by ADWEC. Some of the data that the DCVS is receiving is the metered data collected by the Instation (which is then verified and validated), sent instructions, received availability declarations and non-compliance events from the EDL. It also receives PASS data (plant accounting settlement system) which the DCVS does not validate or verifies but only passes through to S&PS. The DCVS also creates invoices for distribution companies.

2.1.6 Settlement and Planning System

This system which is not located at LDC premises but at ADWEC premises receives validated data from the DCVS system and based on these, it verifies IWPP invoices and creates new invoices for distribution companies to be used later on in claiming processes.

[1] Availability is the amount of output the unit can generate at a specific time instant. Several factors such as low temperature in the boiler due to the type of fuel the unit is currently running on can make the availability of a certain unit to be lower than the maximum stable generation output.

[2] Capability parameters could be maximum and minimum stable generation/production levels.

2.2 DSSF Interfaces

In this section we will get familiar with the interfaces that exist between EDL and other systems. We will exclude the interfaces that do not include the EDL since these do not play an important role in the outcome of the thesis. More precisely we will investigate the actual communication methods, mediums and frequency that lie under these interfaces. In figure 2.2 an overview of the interfaces is shown and all of these will be explained. You may note that the communication is one-sided for the DCVS and Instation system. This indicates that EDL does not import information from DCVS and does not export information to the Instation.

All communication is executed by sending, receiving and reading ASCII-files. As a rule of thumb, the recipient of such a file only have reading access while the sender has both reading and writing access. The layout inside the file will however differ depending on the system that created the file and to whom it was destined for.

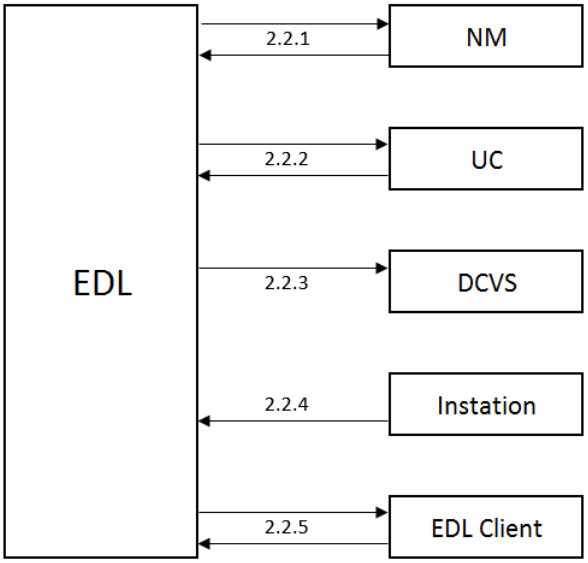


Figure 2.2: A compact overview of all the systems that the EDL interacts with. The arrows display data flow and the numbers is a reference to corresponding sub-section in the report.

2.2.1 Interfaces between EDL and NM

In this section the communication between EDL and Network Manager will be described. Both systems utilize ASCII-files with the file name build up as

$$sendingsystem\_recievingsystem\_Data\_YYYYMMDDhhmm.txt \quad (2.1)$$

where sendingsystem and recievingsystem is either EDL or NM depending on which way the file is transferred. The last part of the file name is made up by

the UTC[3] time stamp at which the file was created in the sending system. The actual transfer is done over LAN together with a shared folder using FTP (File Transfer Protocol). This name convention and file transfer method is similar for the communication between EDL, UC, DCVS and EDL clients.

EDL to NM

The communication link from EDL to the Network Manager should comprise of the following information:

- Load despatch instructions of every unit that is under AGC control (net MW value of the new base-point)
- AGC status (1 if the unit should be put under AGC control and 0 if it should be taken off)
- Operational ramp rates of all units (both up and down)

Since the file name is partly made up by the time stamp, in minutes, the granularity of these files is at one minute. To reduce redundant information exchange, the EDL will act on an ad-hoc basis and only issue a new file if some settings or new instructions has been despatched. If a new file is issued, only units that is related to the underlying changes is included in the file.

NM to EDL

The network manager transmits real time data from the transmission system and power plants to the EDL. The content should include system target frequency, system actual frequency (both averaged over a one minute period), current AGC status and the last sent AGC set-point for every unit. These communication is not ad-hoc and Network Manager thus sends a new file with updated values each minute. The values are only for units that are under AGC control. It's against these values that EDL performs compliance monitoring (remark that this is only true for units under AGC. For units not under AGC, the compliance monitoring is validated against the load despatched instructions created by UC or manually created inside the EDL) and since a new file is created each minute, a single file loss is not an emergency and can easily be parried. For every file that EDL receives it will send back an empty file to Network Manager with the same file name but with the extension .acc or .rej depending on how the reading went. By doing this, Network Manager knows that something is wrong with the communication if it receives a .rej file or if it does not receive an acknowledge file at all from EDL each minute. In this case, the Network Manager will create an alarm in the alarm event list. On the other hand the EDL can determine that something is wrong if it does not receive a new file each minute from NM (since NM is scheduled to send a new file to EDL each minute). By the same reason, EDL will inform the operator of communication failure.

[3]Coordinated Universal Time - A closely successor to Greenwich Mean Time (GMT). The use of UTC over GMT is preferable since GMT is no longer precisely defined by the scientific community.

2.2.2 Interfaces between EDL and UC

The communication between EDL and UC follow standard formats, namely ASCII-files with information written inside the file. It utilizes the same name convention and transfer methods as EDL to NM, see section 2.2.1.

In addition to the ordinary file that is sent, this communication protocol also has a reference file that is used to identify a particular piece of data in both EDL and UC. This reference file is of the form

$$sendingsystem\_recievingsystem\_reference.txt \tag{2.2}$$

and contains an equal amount of lines as data points that is subjected to interfacing (transferred to the other system). The content inside the file looks like

$$< ID - tag >; < ID - EDL >; < ID - UC >; \tag{2.3}$$

where the three tags are string identifiers for the particular piece of data, in EDL system and in UC system respectively.

EDL to UC

EDL transmits the availability declarations, capability parameters, operational ramp rates, and maximum/minimum de-rated capacity of every unit in order for UC to perform its optimization runs. These data has a granularity of one minute and the file is send on an ad-hoc basis to minimize redundant information.

UC to EDL

The UC provides EDL with optimized unit schedules based on technical and economic factors. More precisely, the information that is included in the file should be:

- Schedules for each despatching unit (both power and water units)
- System marginal price for both power generation and water production

The time horizon that the schedule should cover can be configured within the system but is usually one day ahead. The schedules has a granularity of 30 minutes and thus contains half-hourly set points for all units and a new schedule optimization run is performed with a one hour interval.

2.2.3 Interface between EDL and DCVS

This interface is a one-sided communication meaning that EDL exports data to DCVS but not the other way around. Since the DCVS deal with settlement processes, EDL will contribute by forwarding operational data such as:

- All kinds of despatched instructions (see section 3.1.1 for more details on different kinds of instructions that exist)
- Availability declarations
- Non-compliance events

The actual file transfer is through FTP and as usual event driven with the same name format as the other EDL exporting functions.

2.2.4 Interface between EDL and Instation

This is a simple one-sided communication link where the Instation transfer the actual produced net electrical power at each point of generation. The ASCII-file is sent each minute with a granularity of one minute.

2.2.5 Interfaces between EDL and EDL Clients

These two interfaces is the ones that this thesis will be mostly focused on. Some of the information that is transferred between the two system has already been described in section 2.1.3 but will be repeated her for consistency. The communication is thorugh ASCII-files with one minute granularity and on an ad-hoc basis.

EDL to EDL Client

EDL will validate the schedule received from UC and maybe add manual instructions to the schedule before sending it to the power plant. The file is supposed to be pushed to the target directory two minutes before the unit is supposed to change load. Information that is included in this communication direction is:

- Load despatch instructions
- Ancillary service requests
- Prompts such as non-compliance or non-acknowledged messages

It will receive and display to the LDC operator if the power plant accepted or rejected the instruction. In case an acknowledge file is not received, the instruction status is set to not acknowledged and the operator can take appropriate action (check if the communication link is down).

See section 2.3 of the overall despatch concept for a more through explanation of the communication process and what happens in different kind of scenarios.

EDL Client to EDL

When the Client EDL receives the load despatched instructions from the main EDL, the operator must acknowledge its receipt by either accept or reject the incoming instruction. In this interface the power plant operator can declare unit availability and capability parameters to LDC.

2.3 DSSF Overall Despatch Concept

This section will describe and elaborate on the overall despatch concept. It will show the process from estimating instruction schedules to non-compliance events. It will describe how certain events trigger other events and how instruction acknowledgement is handled. The ODC is separated into two parts, the DSF and SSF.

2.3.1 Despatch Special Facilities

Availability declarations, current measurements and load forecasts are collected by the UC system where a power and water schedule is derived and forwarded to the EDL. Well inside the EDL, the schedule is visually displayed and manual changes can be made by an operator (if the changes are small, there is no need for another UC schedule to run). This schedule (or manually modified schedule) is then the basis for the electronic load despatch instructions that are sent to various power plant units. An instruction is then automatically created and sent two minutes before the unit is expected to change its load. Before this instruction is send, a validation process starts that validates this instruction against current unit declared parameters. The unit operators must acknowledge the received load despatch instruction sent by LDC by manually either accept or reject it. If the instruction is rejected, a rejection reason must be entered. If the instruction is not acknowledged within a specific time period, (after several reminder messages has been sent) the LDC operator will be informed and can take appropriate action [3].

EDL performs compliance monitoring in parallel of units by comparing real-time metered values against the despatched instructions (if the unit is under AGC, the monitoring will be against the AGC instructions sent by the Network Manager). If a transgression is made by the unit (the difference between unit output and target load is larger than a specified threshold) the operator is informed and a non-compliance event is triggered. The allowed interval that the unit should lie inside depends on if the unit is under AGC or is frequency sensitive. In case of a non-compliance event, electronic messages can be sent to unit operators informing them to re-declare their availability or adjusting unit outputs. If unit availability is not re-declared within a specified time period and the output is still outside the allowed interval, the LDC operator is then allowed to re-declare the units availability on behalf of the unit operator.

The same procedure is done when a power plant operator is declaring its availability or other capability parameters to LDC. LDC is informed by a popup message that a new declaration has arrived. The LDC operator must then acknowledge this declaration by either accept or reject it. A message is then returned to sending power plant operators informing them if the declaration was accepted or rejected by LDC.

All collected EDL data is sent to DCVS for settlement purposes. The modified schedule is sent to NM and UC which then forms a basis for the next schedule optimization run.

2.3.2 Settlement Special Facilities

Settlement data is pulled from each power plant and transferred to the OTE at LDC. These settlement data is describing each of the power plants power and water units. By the use of a RTU, the information is collected by the DCVS where it is verified and validated. Data from EDL is also collected such as non-compliance events and LDI's etc. Together the DCVS sends the data to the S&PS system where it is transformed to invoices and reports. The S&PS also creates a plant summary report which is sent back to the power plant and stipulates the invoice amount and other important notifications such as errors in the validation process [4].

Chapter 3

Software Design

3.1 Low Level Design

The prototype consist of two separate programs, the main program (from now on called main prototype) that, if possible, should be integrated with SCADA and the smaller program (called client prototype) that acts as a power plant client program (a similar program as the EDL client which is installed at each power plant). These two programs have been developed on the same machine but when they are tested and simulated, they are installed on two separate computers and communicates by sharing a library (acts as a small server on the intranet) and then pushing ASCII-files to these shared folders. Since the programs are developed in the MATLAB environment and due to the fact that it's expensive to buy MATLAB licenses, a special amount of effort has been put down to investigate if MATLAB supports stand-alone applications. As it turns out, it does and this means that a program can be developed on a machine with MATLAB installed but then compiled to an executable (.exe) file that can run on any machine that has the same operating system as the compiling machine. The only thing that is needed is a MATLAB runtime compiler (MRC) on the computer with the same version as of the compiling computer MATLAB version. This MRC can be included in the compiled program (called the package), or downloaded from the internet. The MRC is of course free and the installation is only needed once.

Since these two programs are supposed to have some HMI (Human Machine Interaction) a corresponding GUI was necessary to be designed and developed. Much information about graphical user interface design was found in [5] and [6].

At the end, two final and fully functioning programs where developed with two corresponding interfaces which made it possible for operators to interact with the programs and communicate with each other in a satisfactory way.

3.1.1 Main Prototype

This program is the most extensive and complicated of the two developed programs. Its main functionality is to let operators send manually instructions to the unit operators. Supported instructions are:

- Load change, the most common instruction where the instruction gives the

unit a new target load to which it should reach at a specific time instant.

- Synchronize, instruction sent to units who is inactive but should be connected and synchronized to the grid.
- De-synchronize, sent to units who should be put off the grid and become inactive.
- Cancel synchronize, sent to units who should ignore the last received synchronize or de-synchronize instruction.
- Forced load change, this instruction cannot manually be selected by an operator but do play an important role in the program. This instruction is issued automatically by the system when it for example detects a non-compliance event. Since the operator is changing the availability of the unit, the program issues this instruction so that the target load is not higher than the new re-declared availability level. This instruction cannot be rejected and hence does not need to be acknowledged by the power plant operator.
- An AGC status is sent to a unit who is supposed to be put on or taken off AGC.
- A Back-up fuel instruction tells a unit to start or stop running on back-up fuel. Very important instruction due to the cost difference in the different kind of fuels.
- Frequency sensitive, tells a unit to be or not to be frequency sensitive. This means that the unit will change its load depending on how the frequency of the grid is at the moment. Depends on the speed droop and it will allow a larger tolerance band in the compliance monitoring.

All these instructions are accompanied by a time parameter which indicate at which time instant this instruction should be accomplished by the unit (as you can see in figure 3.1 next to the target time text).

In figure 3.1 the layout of the main prototype GUI is shown. The program operates in real-time and the black bar in the graph is hence the current time (which then moves when a new minute is reached). At the bottom you have three panels: an instruction panel, a summary panel and a multi-panel. In the multi-panel you can select which plant and unit you want to be displayed in the graph and to be summarized in the summary panel. There is a special feature in these two pop up-menus which allows you to select all plants, all power units or all desalination units, see figure 3.2. If you for example, in the unit pop up-menu, select the all power units item you can see in the graph and in the summary panel all the characteristics for the selected plant's power units. This allows the operator to get a good and swift overview of a particular plant's power units etc. At the bottom in the multi-panel you can see the connection to all the power plants that this prototype is linked to. If the connection is up and running, there will be a green checkmark otherwise there will be a red cross indicating a failing connection.

In the summary panel you can see the selected units current values and properties which continually updates. If the operator has sent an ancillary

instruction, for example that the unit should be put on AGC, a time stamp will be present next to the AGC Status text indicating the time the unit is supposed to be put on AGC.

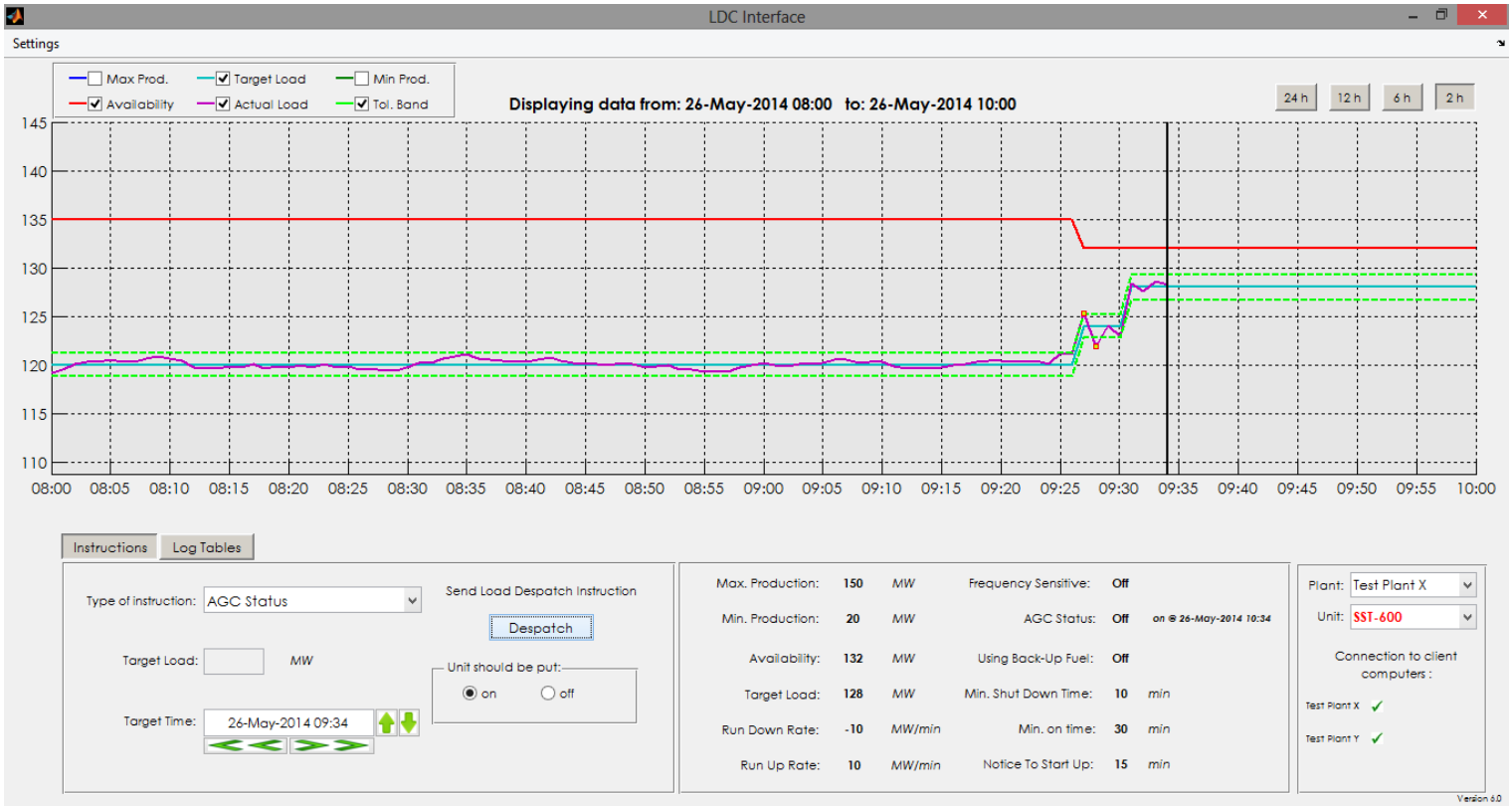
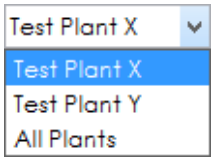


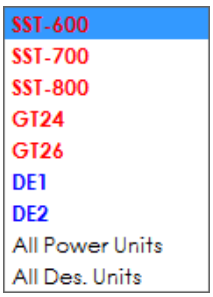
Figure 3.1: This is the main prototype interface. A graph displaying both past and future values takes up a large portion of the GUI. You can choose which line that should be included in the graph at the top left corner and which time span that should be used in the top right corner.

In the instruction panel the operator can send instructions to the unit, see figure 3.2 for all different kinds of instructions. Some instructions require a target load (for example the load change instruction) and most are accompanied by a target time. After the operator has selected an instruction, target load and a target time he can despatch the instruction by hitting the despatch push button. Remark that the instruction will be sent to the selected unit in the unit pop up-menu inside the multi-panel.

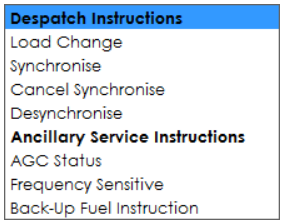
If the operator press the Log Tables push button next to the instruction push button, two other panels will switch places with the instruction panel and summary panel. These panels has the same size as the two shown in figure 3.1 and the graph is still present. In figure 3.3 there is a zoom in on the two panels. Here the operator can see the latest load despatch instructions and received declarations from the selected unit. Only accepted instructions is shown here. A pushbutton labelled full view is also visible which, if pressed, opens the full log view which is the collection of all send instructions and all received



(a) The plant popup menu.



(b) The unit popup menu. Red labels indicates power units and blue desalination units.



(c) The instruction popup menu.

Figure 3.2: The three pop-up menus that exist in the main prototype GUI. Remark that the unit pop-up menu 3.2b and the instruction pop-up menu 3.2c changes respectively depending on which plant and unit the operator has selected.

declarations. Figure 3.4 displays this log table which can be seen as a database where all information is stored. No records can be deleted from this database, only overwritten etc. if such action is needed. This will make a full trail of all instructions and an operator can see exactly what has happened since program started.

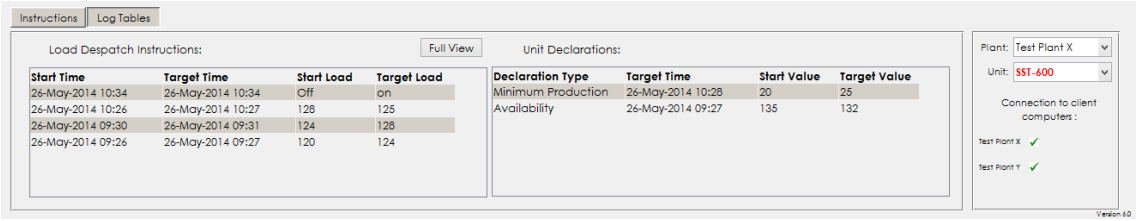


Figure 3.3: This is the log table view and displays the six latest despatched instructions and received declarations for the selected unit in the multi-panel.

In the top left corner of the main GUI interface you can see a Settings button. If pushed, the settings GUI will be displayed to the operator where he can: change folders in the shared library, change unit parameters (such as ramp rates or minimum shut down time etc.), add a new plant, add a new unit to existing plant, remove plant or unit. The settings GUI is shown in figure 3.5

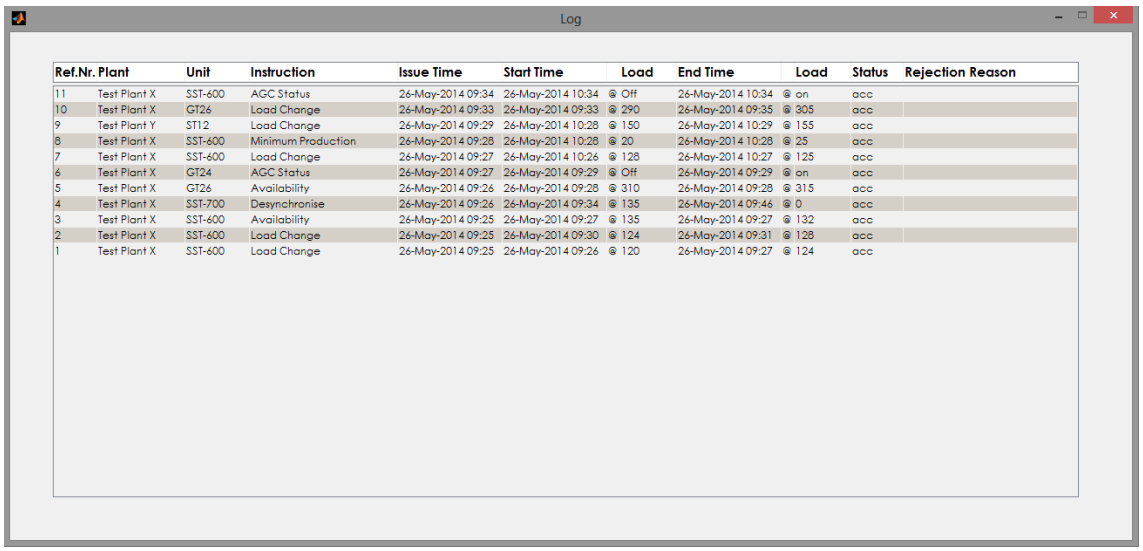


Figure 3.4: The log table that acts as a database where all records are stored and cannot be deleted.

3.1.2 Client Prototype

The client prototype act as a counterpart to the main prototype. It is much smaller and does not comprise the same functionality and freedom as the main prototype does (which it should not do since it is the LDC operators who should have most and complete control on what's happening). In figure 3.6 the client prototype interface is shown. The two tables on the left side of the GUI is a kind of log tables where load despatch instructions and unit declarations are stored. Those two tables do not change depending on which unit that is selected in the select unit pop up-menu. In the tables you can see the instructions reference number (each instruction, declaration, non-compliance event etc. has a unique reference number to make it easy to track), unit name, start and target times, start and target loads, type of instruction and declaration and a comment section. The comment section can include rejection reasons etc.

At the bottom right corner, a connection panel displays the current connection to the main prototype. This image will change to a red cross if something is wrong with the communication. Above this panel exists a summary panel that displays current values and scheduled events. The current section shows the current values for the selected unit in the pop up-menu. The scheduled section shows the nearest event that will occur for this unit. It displays a value (the target value) and a time stamp (at which the target value should be achieved). Dotted lines indicates that no future instruction or declaration exists for this particular unit.

At the top right corner you see the name of the plant and a pop up-menu at which you can select different kind of units. Under this menu you have a new declaration push button. The power plant operator will press this button if he want to make a new declaration and if pushed, he will be redirected to the GUI shown in figure 3.7. Remark that the declaration will be for the unit that was selected in the unit pop up-menu before pushing the new declaration push

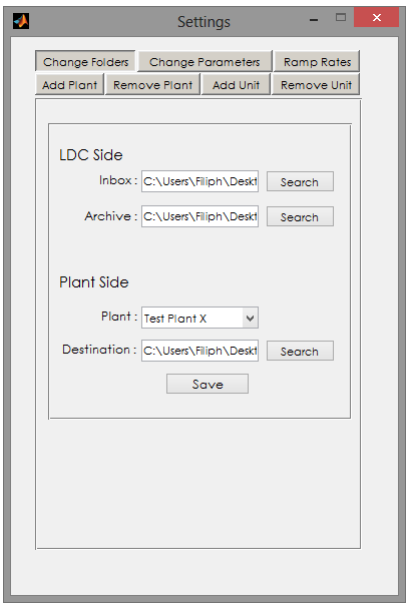


Figure 3.5: Here is the change folders toggle button selected. You can change the directories for inbox folders or archive folders for old stored files. You can also change destination folders on the power plant sides.

button. The most common declaration is for availability but the prototype also supports changes in maximum production and minimum production.

If the top left corner of the client prototype there exist a settings button that will display the settings GUI. This GUI is very similar to the main prototype settings GUI but with a few small alternations, for example, the power plant operator cannot add a new power plant etc.

3.2 High Level Design

This section walks through the prototype programs on a smaller scale with more specific details on how some complications are resolved.

3.2.1 Main Prototype

After the system has been initialized, an infinite timer starts. The timer has a fixed rate execution mode, which means that after exactly five seconds it will iterate the execution code one more time and after the execution it waits the remaining seconds and then executes the code again. Hence the timer has no stop since this prototype is developed as a program that never should be turned off. The tasks that it performs can be seen in figure 3.8. The second block is simulating the power plants, which includes simulating every unit output and adding noise to get a more realistic simulation. This step will be removed if the prototype is commercially implemented and instead of simulation the program will just extract real operational data from the power plants. The third block is the compliance monitoring part which is an important and complicated process.

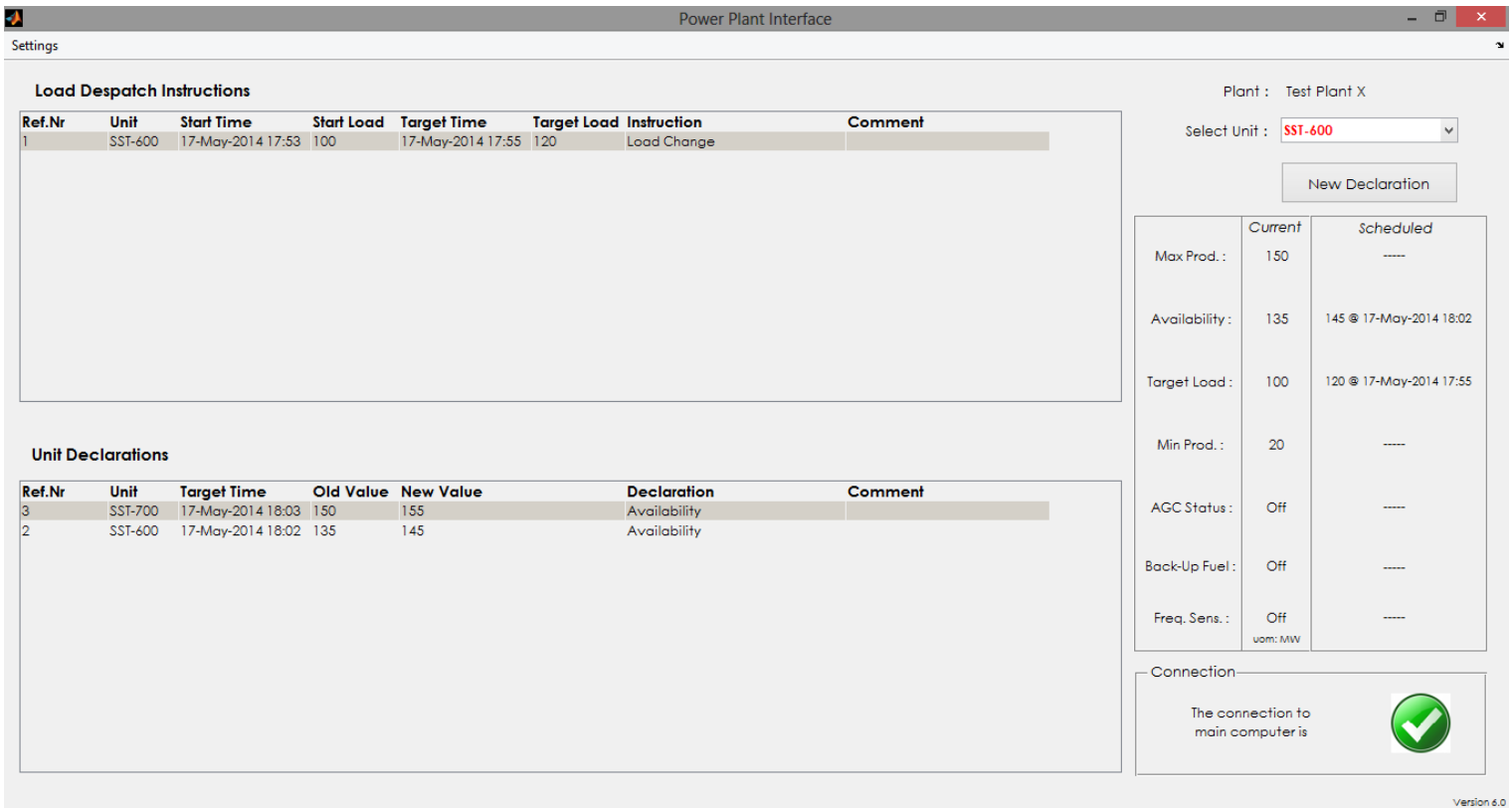


Figure 3.6: This is the client prototype interface. This interface does not have a graphical presentation of the data since these values are not available at plant side. Instead it has a table representation of all the received instructions and sent declarations.

The program will perform a series of tasks in order to determine if a reminder should be sent to the power plant operator or if LDC should re-declare unit availability on the power plants behalf:

1. Map corresponding unit to correct threshold value. If the unit output is outside the allowed interval, add a "point" to this units non-compliance parameter.
2. If the amount of non-compliance points is equal to five i.e. the unit has been outside the allowed threshold for five minutes, send a reminder to power plant operator and informing about unit non-compliance and encourage to adjust unit output or re-declare unit availability.
3. If instead the number of non-compliance points is equal to 10, the program will initiate the re-declaration process which includes:
   (a) Re-declare unit availability to the highest value the unit output reached within the trespassing period. This newly declared availability value is then valid all the way back to when the unit last declared its availability.

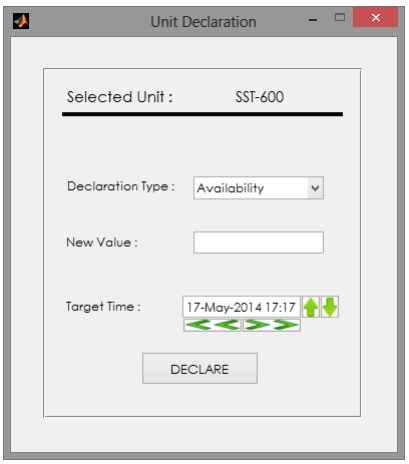


Figure 3.7: This GUI will open up when the power plant operator wants to make a new unit declaration. He can select between different kinds of declarations in the pop up-menu, select corresponding value and select a target time after which this value should apply.

(b) Issue a forced load change instruction to the power plant informing them that the unit target load is now equal to the new availability value. This is done since the unit cannot have a target load higher than its declared availability.

(c) Set future load despatch instructions and availability declarations to overridden and remove them from scheduled events.

(d) Add new records in the database of a non-compliance event.

(e) Reset the non-compliance points for this unit to zero.

(f) Inform power plant operator of the new changes.

The fourth box checks the connection to each power plant. Since the programs push files to each other by a shared library, the connection can be checked by see if the program has access to this shared folder. If the connection is down, the program will have its access denied and hence something is wrong. Depending on the connection, the prototype will update the connection panel in the main prototype interface by appropriate images (green checkmark or red cross).

The fifth and sixth step updates the graph and summary panel with new data. The last step is the check for new files step which is more elaborated in figure 3.9

The incoming files can have two different name formats. The first format is used when the power plant is declaring parameters for a unit and is of the form

$$CLIENT\_LDC\_DATA\_YYYYMMDDhhmm.txt \tag{3.1}$$

and the second form is used when power plants has acknowledged a despatched instruction and is of the form

$$CLIENT\_LDC\_DATA\_YYYYMMDDhhmm.acc.txt \tag{3.2}$$

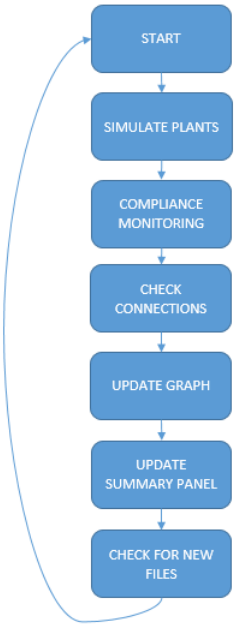


Figure 3.8: A block diagram showing the execution tasks that the main prototype timer performs. Another loop execution is started exactly five seconds after the previous loop was started.

$$CLIENT\_LDC\_DATA\_YYYYMMDDhhmm.rej.txt \tag{3.3}$$

depending if the instruction was accepted or rejected. Hence different actions are taken whether the file is a so called declaration file or just a acknowledge file sent from the power plant.

If the file was of the first type, the prototype will start to read the content in the file and send the content to a help GUI, the new declaration GUI. In this GUI the LDC operator can see what type of declaration it was, corresponding time stamp and value and be able to accept or reject the incoming declaration. Depending on the answer, the prototype will send a file of the form in the equation 3.2 or 3.3 but with the names CLIENT and LDC switched. The data content inside the declaration file is of the form:

$$plant\_id; unit\_id; declaration\_type; issue\_time; target\_time; target\_value; \tag{3.4}$$

If the received file was instead an acknowledge file, the prototype will map this acknowledge to the right instruction and then set it to accepted or rejected depending on the file name extension. One of the pop up-messages from figure 3.10 will also be displayed.

Before the LDC operator can reject or accept a new declaration file, a validation script will be run which determine if the new declaration is valid i.e. checks that availability is higher than minimum production and smaller than target load and maximum production and so on. If for some reason the declaration

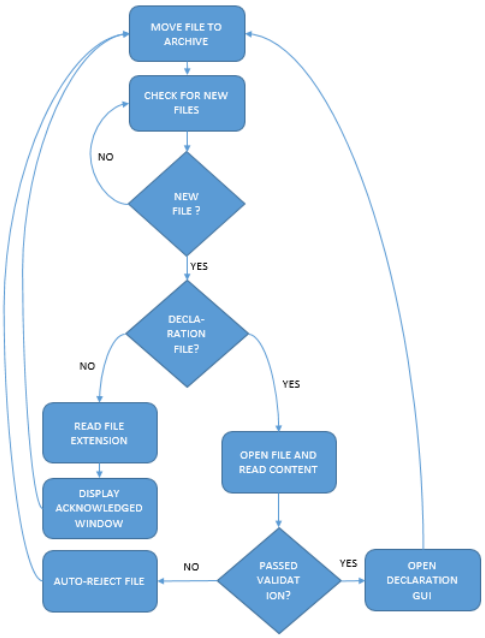


Figure 3.9: The steps taken when the prototype is searching for new files and the corresponding actions it takes.

does not pass the validation test an automatic rejection will be initiated which rejects the declaration without displaying the declaration to the LDC operator, only informing him that an automatic rejection has been done and why it was rejected. This automatic procedure is performed to avoid ambiguities in the system, if for example the declaration was not rejected but the operator was only informed that the declaration violated some parameters it could happened that the declaration still was accepted even though it was not meant to. In all processes when human interaction is present, some errors will occur.

These tasks are repeated every five seconds. On the other hand, the LDC operator can interact with the main prototype interface ad-hoc style. This means that none of the code written for the user interface controls is executed until that control is pushed. Thus a load despatch instruction can be sent at all the times.

If for some reason an instruction or declaration is rejected, a rejection reason must be entered and will be written inside the rejected acknowledgement file. Remark also that in each file that LDC transmits to the power plants will contain a unique reference number in order to keep track of all records and not mix them up.

3.2.2 Client Prototype

The client prototype works almost the same as the main prototype. It has a similar timer which executes each and every five seconds. It will update connections, summary and schedule panels and check for new incoming files. This side can also receive two different kinds of files: acknowledge files and new

(a)

(b)

Figure 3.10: One of these GUI's will be displayed to the operator depending on the acknowledgement of the instruction/declaration.

load despatch instruction files.

If the client prototype receives a load despatch instruction, it will begin by reading the content which is of the form

$$ref\_nr; unit\_id; instruction\_type; start\_time; target\_time; target\_load; \quad (3.5)$$

and here the instruction type determines which actions the prototype takes. The instruction type could be

- A regular LDI such as: load change, synchronize and desynchronize etc.
- A forced load change instruction which cannot be rejected and hence does not need an acknowledgement file to be sent back to LDC.
- A non-compliance reminder.
- A forced availability re-declaration instruction i.e. LDC has re-declared an unit availability on the power plant operators behalf.

If the instruction is of the first form the prototype will take normal action, the same as for the main prototype which was to display the content in the LDI GUI and let the operator examine the values and accept or reject the instruction. If on the other hand the instruction is a forced load change instruction, the prototype will display the content in the same LDI GUI but with the rejection push button made invisible i.e. the operator can only accept the instruction. If the program detects a non-compliance reminder it will, before deleting the

file, show the operator an electronic message informing that he must look after one of his units before LDC re-declares the availability. If the instruction is of the fourth type, an availability re-declaration, only a pop up-message will be displayed to the operator informing him that LDC has redeclared availability of one of his units due to non-compliance under a longer time.

The same is applied here that the operator interaction is ad-hoc and can hence send a declaration at any time but with a granularity of one minute.

3.2.3 MATLAB Classes

Four different classes has been developed in MATALAB to make it easier to simulate the behaviour of power plants. The hierarchy of the classes can be seen in figure 3.11. The mother class, Power Plants, is a singleton i.e. it can only exist in one instance. This class has plant objects as properties and these in turn has unit objects as properties.

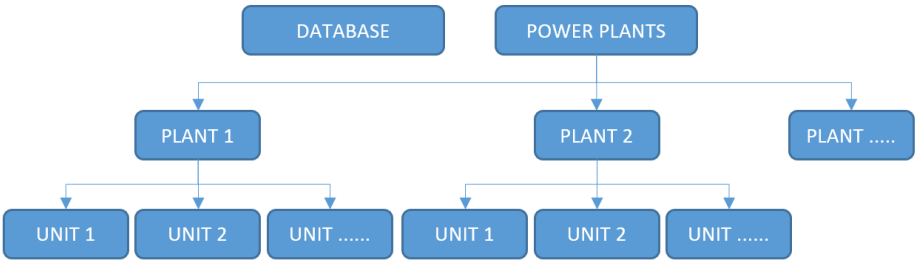


Figure 3.11: Hierarchy of the MATLAB classes. You can control each plant and unit directly from the Power Plants class. To avoid ambiguities, all plant and unit properties has protected access so the only way to change these is to use the created set and get methods.

The Unit class is a static class which means that it know its current values but not past nor future target loads. The Plant class handles these data for all units and has several other methods to alternate and calculate different things. To change values from the Power Plants class you must use parameter and value pairs of the form

$$power\_plant.set('Property1', Value1....,' PropertyN', ValueN); \tag{3.6}$$

where two of the property value pairs must be the plant and unit to which the values should be applied for. The corresponding value could either be the unit name (id) or the unit object. You could also skip the plant step and directly go to the correct plant object and call the set or get functions

$$plant.set('Property1', Value1....,' PropertyN', ValueN); \tag{3.7}$$

where only one pair must be a unit identifier and the rest is property value pairs.

The Power Plants class comprises functions such as

- Add plant or unit

- Remove plant or unit
- Step plants (used by the main prototype to simulate the unit outputs)
- Return plant and unit names (id's)

among many others.

The database class is a tool to store records of previously despatched instructions (both accepted and rejected) as well as unit declarations. The database class is a form of a table where no records can be removed in order to create a full trail of instructions and declarations. Hence this class don't have a remove record method but only an add method. The class comprises many other functions which mostly relates to if, when and to whom some specific future instructions exist.

Chapter 4

Conclusions

The prototype that has been developed as a main part of this thesis is fully functioning in the environment that it's operating in now, meaning that a few alternations must be done before it could be integrated within SCADA. These alternations and improvements can be found in the for future work section. It has altough been developed with as much standardisation as possible (file name conventions, file transfer techniques, instruction types related to the Abu Dhabi market and so on).

The main prototype consist of all the applications that was specified at the beginning of the project as well as a compliance monitoring part (this was sought but could be too time consuming). Applications such as sending load despatch instructions, receiving declarations from different units and power plants, change parameters within the system, log all data in a database for a complete process trace etc. is some of the developed functions. I think the interface is functional and gives the operator an overview of the complete power system and all its components. A lot of time was put down on developing the main prototype graph which I think was necessary since this tool gives the operator the ability to track unit (or several units simultaneously) outputs.

Many things has been learned while doing this thesis, both do's and don'ts. The most personal experience was the ability to study and learn how complicated graphical user interfaces is designed and developed. A lot of experience has also come to how the actual overall despatch concept work and the issues that exist in today's solutions. A lesson learned is that you should be absolutely clear on how the design of the GUI should look like before you start programming and designing the interface in the computer. This will save a lot of time and frustrating hours of re-programming.

For future work

If a few more months would be available, I think that a complete, secure and satisfactory system would be finished that could be fully integrated with SCADA at TRANSCO. The things that are left to be done are

- Create a log-in environment where the operators must log in to the system when they begin their control and hence you can see who despatched what

for a more precise track of information. This will also secure the system for potential threats.

- Create an Excel tool which exports information and data into an excel sheet that could be sent to appropriate destinations.
- Investigation in SCADA integration and make appropriate adjustments so the integration becomes as smooth as possible.
- Adjust certain sections in the program so it becomes more adjustable to other systems inside the DSSF.
- Add a feature in the database where you can sort the entries based on certain criteria such as: plant, unit and instruction etc.
- Develop a more secure system that can live through a power out without a system crash. A solution could be to save all settings, parameters and settings frequently and if the system gets interrupted the system can load the previously saved settings and it can begin were it left off.
- Add a beeping noise when a declaration or LDI is received to get the operators attention faster.
- Run through different kind of scenarios, even unlikely scenarios, to test the validity of the program.
- Create interfaces to other programs within the DSSF so the prototype could communicate with these as well.
- Evolve the prototype from an academic point of view to a more industrial fitted prototype. The prototype is for example developed to be used by truthful people and not by people who would like to make as much money as possible by dodging and cheating the system. At the moment the power plant operator can ignore a load despatch instruction by just ignoring the electronic message (by not pushing the accept or reject button). If this happens, the prototype sets the instruction to not-acknowledged and moves it to the archive without taking any other actions. To avoid this type of behaviour could be to drastically re-declare the unit availability to zero.

The AGC within SCADA has an application called AGC non-tracking test which is very similar to the compliance monitoring part in the prototype. This non-tracking test is useful to detect units that is under AGC but does not answer to the despatched AGC instructions. This test identifies these units and sends several reminder messages before SCADA shuts the unit off (it sets the unit to "not answering" inside the system and does not despatch more instructions to this unit). Since this test and the compliance monitoring part is so similar, a good way to get more understanding of SCADA and how a possible integration could be done is to study this code section inside SCADA. It may also be here the compliance monitoring code could be inserted.

Acronyms

ADWEA Abu Dhabi Water and Electricity Authority.

ADWEC Abu Dhabi Water and Electricity Company.

AGC Automatic Generation Control.

DCVS Data Collection and Verification System.

DSSF Despatch and Settlement Special Facilities.

EDL Electronic Despatch and Logging system.

FTP File Transfer Protocol.

GUI Graphical User Interface.

HMI Human Machine Interaction.

IWPP Independent Water and Power Producer.

LDC Load Despatch Center.

LDI Load Despatch Instruction.

MRC MATLAB Runtime Compiler.

NM Network Manager.

ODC Overall Despatch Concept.

RTU Remote Terminal Unit.

S&PS Settlement and Planning System.

SCADA Supervisory Control And Data Acquisition.

TRANSCO Abu Dhabi Transmission and Despatch Company.

UC Unit Commitment.

Bibliography

[1] J Darling/R Gomm. *Electronic Despatch & Logging, Functional Design Specification.* RWE Power International, 1.5 edition, 2009.

[2] *Data Acquisition System SIGLON (AMR Automated Meter Reading).* BAER, 4.10 edition, 2014.

[3] *Technical Supplements to Appendix D, Functional and Technical Specifications for Despatch Special Facilities.* ADWEA, 2001.

[4] *Technical Supplements to Appendix D, Functional and Technical Specifications for Settlement Special Facilities.* ADWEA, 2001.

[5] *MATLAB® Creating Graphical User Interfaces*, volume R2014a. MathWorks, 2014.

[6] Holly Moore. *MATLAB® for Engineers.* PEARSON, 3^{rd} edition, 2012.