Transport Management Services in C200 web server
AN APPROACH TO INTEGRATE INTERACTIVE SERVICES

CHRISTIAN ILLANES
Transport Management Services in C200 web server

An approach to integrate interactive services

Transportledningstjänster i C200 webserver

Ett försök i att integrera interaktiva tjänster

Christian Illanes

illanes@kth.se

Degree project in Computer Science and Engineering

Supervisor and examiner: Stefan Arnborg

Employer: Scania CV AB

Stockholm June 2015
Abstract

Title: Transport Management Services in C200 web server – An approach to integrate interactive services.

This report studies the possibility of integrating Transport Management Services (TMS) in a low-cost platform called Communicator 200 (C200). The platform is developed by Scania and is a black box that manages vehicle data, positioning, and wireless communication. The C200’s main purpose is to connect a hauler’s vehicles to their office system via wireless links and Internet. The thesis describes the design and implements a prototype of a service called Messaging that is used to send text messages between truck and office. The prototype is evaluated and tested with different web browsers in various hand-held terminals with different screen sizes. The main conclusion is that it is technically possible to integrate TMS in the C200 without altering the platform architecture too much.

Sammanfattning

Titel: Transportledningstjänster i C200 webbserver – Ett försök i att integrera interaktiva tjänster.

### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Problem Statement</td>
<td>2</td>
</tr>
<tr>
<td>1.3</td>
<td>Objective</td>
<td>2</td>
</tr>
<tr>
<td>1.4</td>
<td>Methodology, Resources, and Limitations</td>
<td>2</td>
</tr>
<tr>
<td>1.5</td>
<td>Outline</td>
<td>3</td>
</tr>
<tr>
<td>1.6</td>
<td>Target Audience</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Scania Fleet Management</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Scania Fleet Management Services</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Transport Management Services</td>
<td>5</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Vehicle Management Services</td>
<td>6</td>
</tr>
<tr>
<td>2.1.3</td>
<td>Driver Support</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Scania Fleet Management Portal</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>Vehicle Platforms</td>
<td>9</td>
</tr>
<tr>
<td>3.1</td>
<td>Scania Vehicle Informatics Platform</td>
<td>9</td>
</tr>
<tr>
<td>3.2</td>
<td>Communicator 200</td>
<td>10</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Hardware</td>
<td>11</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Software Architecture</td>
<td>12</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Application Architecture</td>
<td>12</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Communication Architecture</td>
<td>14</td>
</tr>
<tr>
<td>3.2.5</td>
<td>Test Bench</td>
<td>14</td>
</tr>
<tr>
<td>3.2.6</td>
<td>Software</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>Preliminary Study</td>
<td>15</td>
</tr>
<tr>
<td>4.1</td>
<td>Connection Alternatives</td>
<td>15</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Bluetooth</td>
<td>15</td>
</tr>
<tr>
<td>4.1.2</td>
<td>RS-232</td>
<td>15</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Ethernet</td>
<td>16</td>
</tr>
<tr>
<td>4.2</td>
<td>Previous works</td>
<td>16</td>
</tr>
<tr>
<td>4.3</td>
<td>Solution Proposal</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Prototype Implementation</td>
<td>18</td>
</tr>
<tr>
<td>5.1</td>
<td>The WebPortal Application</td>
<td>18</td>
</tr>
<tr>
<td>5.2</td>
<td>Choice of Technologies</td>
<td>19</td>
</tr>
<tr>
<td><strong>GLOSSARY</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>AJAX</td>
<td>Asynchronous JavaScript and XML</td>
<td></td>
</tr>
<tr>
<td>AP</td>
<td>Access Point</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
<td></td>
</tr>
<tr>
<td>BSP</td>
<td>Board Support Package</td>
<td></td>
</tr>
<tr>
<td>BWT</td>
<td>Burrows-Wheeler transform</td>
<td></td>
</tr>
<tr>
<td>C200</td>
<td>Communicator 200</td>
<td></td>
</tr>
<tr>
<td>CAN</td>
<td>Controller Area Network</td>
<td></td>
</tr>
<tr>
<td>CGI</td>
<td>Common Gateway Interface</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
<td></td>
</tr>
<tr>
<td>CSS</td>
<td>Cascade Style Sheet</td>
<td></td>
</tr>
<tr>
<td>CWT</td>
<td>Context Tree Weighting</td>
<td></td>
</tr>
<tr>
<td>DMC</td>
<td>Dynamic Markov compression</td>
<td></td>
</tr>
<tr>
<td>DRAM</td>
<td>Dynamic Random Access Memory</td>
<td></td>
</tr>
<tr>
<td>FMP</td>
<td>Fleet Management Portal</td>
<td></td>
</tr>
<tr>
<td>FMS</td>
<td>Fleet Management System</td>
<td></td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
<td></td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
<td></td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile communications</td>
<td></td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
<td></td>
</tr>
<tr>
<td>HDD</td>
<td>Hard Disk Drive</td>
<td></td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
<td></td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
<td></td>
</tr>
<tr>
<td>HW</td>
<td>Hardware</td>
<td></td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
<td></td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
<td></td>
</tr>
<tr>
<td>LZW</td>
<td>Lempel-Ziv-Welch</td>
<td></td>
</tr>
<tr>
<td>MMS</td>
<td>Multimedia Messaging Service</td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
<td></td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
<td></td>
</tr>
<tr>
<td>PSP</td>
<td>PlayStation Portable</td>
<td></td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
<td></td>
</tr>
<tr>
<td>ROM</td>
<td>Read-Only Memory</td>
<td></td>
</tr>
<tr>
<td>RS232</td>
<td>Recommended Standard 232</td>
<td></td>
</tr>
<tr>
<td>SDRAM</td>
<td>Synchronous Dynamic Random Access Memory</td>
<td></td>
</tr>
<tr>
<td>SMS</td>
<td>Short Message Service</td>
<td></td>
</tr>
<tr>
<td>STL</td>
<td>Standard Template Library</td>
<td></td>
</tr>
<tr>
<td>SVIP</td>
<td>Scania Vehicle Informatics Platform</td>
<td></td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
<td></td>
</tr>
<tr>
<td>TMS</td>
<td>Transport Management Services</td>
<td></td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
<td></td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
<td></td>
</tr>
<tr>
<td>UTC</td>
<td>Coordinated Universal Time</td>
<td></td>
</tr>
<tr>
<td>VMS</td>
<td>Vehicle Management Services</td>
<td></td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
<td></td>
</tr>
</tbody>
</table>
1 INTRODUCTION

Chapter 1 is the introductory chapter, giving a presentation of the background, problem statement, and objective of the thesis. A detailed outline is also presented together with a description of the resources, limitations and target audience of this thesis.

This thesis project for a Master degree in Computer Science was carried out at Scania CV AB in Södertälje, Sweden in 2008-2009. The topic for this thesis is Transport Management Services (TMS) which is a subset of the Scania Fleet Management System (FMS). TMS is dealt within the scope of a low cost vehicle communication platform called Communicator 200 (C200) and studies the possibility of integrating TMS into the C200.

1.1 BACKGROUND

In today’s demanding business environment every moment counts. Transport logistics and vehicle management are becoming increasingly complex and the growing competition between companies has strained them to improve their efficiency. Scania Fleet Management provides haulers with computer-based services that connect their vehicles to their office system via wireless links and Internet, see Figure 1. The main purpose of the FMS is to provide haulers with services that keep track of the company’s assets, improves the company’s efficiency, and streamlines the administration [1].

![Figure 1: Illustration presenting the link between truck and office.](image)

A low cost vehicle communication platform called C200 was being developed by Scania while this thesis project was carried out. This platform that was Linux based had no screen and had less memory than its counterpart called Scania Interactor. It was not trying to replace the Scania Interactor, a rugged on-board computer with color screen and Windows XP Embedded, but was rather attempting to break apart the telecommunication part from the Scania Interactor so it became a more refined computer.
1.2 Problem Statement

The Scania Interactor was an expensive vehicle platform that supported a great deal of services provided by Scania Fleet Management. The problem of the Scania Interactor was that the implemented services were complexly interwoven with the telecommunication part. As a part of the development of modularization and standardization of the FMS the C200 was originated. Since the C200 had no screen it did not support services where the driver could interact with the platform or with the office staff and vice versa. The C200 supported consequently only office level services\(^1\) if used by itself. It was thought that in the future it would be used as an intermediary node between the new Scania Interactor platform and the hauler’s office system. Meanwhile the C200 was being developed; one interesting question gave rise to this thesis.

- Is it possible to connect other platforms with screen to the C200 other than the new Scania Interactor platform?

If that question could be answered positively the following questions were also of interest:

- In what way can or should the devices communicate with the C200?
- Can services other than office level services be implemented into the C200?
- Is it possible to integrate interactive services without modifying the actual server and communication solution?
- What interactive services can or cannot be implemented and how can they be designed to fit the different platforms connected to the C200?

1.3 Objective

The main objective of this thesis was to study if it was possible to integrate TMS into the C200 presenting them on an arbitrary chosen hand-held device such as a Laptop, Personal Digital Assistant (PDA), Sony PlayStation Portable (PSP), smart phone etc. Electronic devices like these were becoming affordable and together with the inexpensive\(^2\) C200 platform Scania hoped to form a competitive product on the market.

1.4 Methodology, Resources, and Limitations

The main focus of this thesis was set upon product development, and the investigating part of this report concerned product review and evaluation. Hence, most of the resources have been internal data sheets and product specifications, rather than scientific articles. Since TMS is an umbrella term for a bunch of services, only one service was implemented and evaluated. This thesis was based on the assumption that it is sufficient to show that, if one service works from a technically point of view, then the other services will work too.

---

\(^1\) See the definition of “office level services” in chapter 2.1.2 Vehicle Management Services.
\(^2\) Inexpensive in comparison to the price of the Scania Interactor, approximately by a factor 10. The price of a C200 in the year 2008–2009 was about 4 000 – 5 000 SEK and 40 000 – 50 000 for a Scania Interactor 500.
This given that the other services use the same or similar technologies as the one implemented and that they do not exceed the accessible resources provided by the C200. The service that was implemented as a prototype is called Messaging. It is used to send text messages between the vehicle driver and the office staff. Messaging was a simple, yet complex enough, service that dealt with the questions put in 1.2 Problem Statement.

1.5 Outline

This report is divided into two parts, one theoretical and one practical. The theoretical part (chapter 2 and 3) provides information about the Scania FMS in the year 2009 and gives a thorough introduction to the different Scania vehicle platforms back by then. The practical part deals with the questions put in paragraph 1.2. It presents the solution proposal, the design issues and the developed prototype. At the end the prototype was evaluated. The chapters listed below give the reader a complete picture of the outline of this thesis.

Chapter 1 is the introductory chapter.

Chapter 2 gives a general introduction to the Scania Fleet Management System. This chapter helps the reader to get familiar with concepts introduced within Scania FMS and gives a better understanding of the background of this thesis.

Chapter 3 gives the reader a thorough introduction to the vehicle platforms introduced in paragraph 1.2. The C200 hardware and software architecture is presented and some information about the resources and limitations of the platform is given.

Chapter 4 introduces the problem prior to the design- and implementation-phase of the service prototype Messaging, namely the problem of how to connect different hand-held devices to the C200. It also discusses what previous works were of interest when considering how to implement the prototype. At the end a choice about what communication alternative to use was made.

Chapter 5 is about the prototype implementation. A functional description of an application called WebPortal that have similarities with Messaging is given, server software and parsers are discussed, the messages between the C200 and FMP are designed, and the prototype implementation is presented.

Chapter 6 presents the result of the final prototype tested on a selection of hand-held devices and web browsers available in the year 2009. It also makes an assessment of which services among transport management services and driver support that are appropriate or inappropriate candidates to be implemented into the C200 with respect to this thesis constraints and prototype result.

Chapter 7 discusses the result and presents the conclusions from this thesis.

Chapter 8 discusses how the developed communication solution can be improved.
1.6 **Target Audience**

Readers of this work are presumed to have a background in computer science in general and knowledge in web programming in particular. Concepts introduced within FMS are explained and are therefore not a required knowledge.
Chapter 2 introduces the Scania Fleet Management System and helps the reader to get familiar with concepts used in Scania FMS. It also gives a better understanding of this report’s background.

Fleet Management is a term widely used to describe the management of a company’s vehicle fleet. It is a function that allow companies which rely on transportation in their business to eliminate or minimize the risk associated with vehicle investment, improving efficiency, productivity and reducing their overall transportation costs. Fleet management systems can include a series of fleet management functions such as vehicle financing, vehicle maintenance, vehicle telematics, driver management, fuel management and more [2]. Functions provided by Scania Fleet Management are based on an integrated product concept that combines vehicles, vehicle support services, and financing. An essential part of vehicle support is the set of offerings called Scania Fleet Management services.

2.1 Scania Fleet Management Services

Scania Fleet Management services make use of vehicle computers, telecommunications and the internet to improve the communication between office staff and truck drivers. These services generate the data needed for accurate performance analysis and operational improvement in terms of fuel economy, reduced administration time, increased uptime and profitability. Scania FMS in 2009 consisted of several services which were categorized into different fleet management functions. Transport management, vehicle management, and driver support are three of those that will be described here.

2.1.1 Transport Management Services

As the reader may already know, the focus of this thesis is Transport Management Services (TMS). Although several services exist within this function, not all of them are of interest. This thesis concentrates on services that has some form of interaction with the driver. Those services are Scania Order Support, Scania Driver Log, and Messaging. Other services that are not of interest but that form part of this function are Positioning and Tracking. As the name indicates, TMS aims at improving the transportation part of the FMS. The transportation part is mainly improved by creating a good communication between drivers and office staff. It aims also at helping planners to keep track of the company’s vehicles. Here follows a brief explanation of the services within this function.

Scania Order Support is a software and service package that aims at speeding up business communication. Job details are sent via the office system to the vehicle computer where the driver can update the order status, e.g. accepted, rejected, loading, and delivered. This information is sent back to the office system throughout the day and gives the administration staff a good survey of the vehicle’s status. To streamline the administration

---

3 Telematics is the proposed name for the technology area and application scope arising from the fusion of computer technology and digital communications [37]. The term vehicle telematics is used to define connected vehicles interchanging electronic data.
this system is combined with the services *Positioning* and *Tracking*. Extensions to this service can be made. Enhanced cash flow is one extension that improves the customer service and reduces administration by taking care of invoices raised and dispatched before a driver returns to the depot.

*Scania Driver Log* is a semi-automatic software application that intends to help drivers to record their activities and hours in an easy way. Activities are recorded by the driver on the vehicle computer at the same time that detail information such as time, odometer reading, vehicle position and fuel consumption are registered automatically by the software. This information is sent to the office system providing the office staff with reliable data for activity reports, wage calculations, and business analysis. Nowadays, at the writing moment, the *Driver Log* service has (in the way it was implemented in the Scania Interactor) lost its significance since it has been fully replaced by another unit called digital tachograph. The unit together with the developed service registers the vehicle’s and driver’s activities and working hours automatically. The registered information is then sent to a remote storage at Scania and can be retrieved later by the customers (office personnel and vehicle drivers) through a web portal.

*Messaging* is a simple method for sending text messages to and from unlimited number of vehicles, in group or individually. Scania believes that the benefits of a text based communication are reduced communication costs, reduced misunderstandings and maximized efficiency. This service is limited to communications between the office and the vehicles to prevent extraneous messages.

*Positioning* sends information of a vehicle’s location provided by the GPS to the office system. Data is sent regularly on a configurable time interval, but can also be requested whenever the office staff demands it. With this system, planners can see whether a vehicle is far away from the cargo place or if it has arrived.

*Tracking* collects the vehicle’s positions giving the possibility for administrators to keep track of every vehicle and how it has progressed along its route. It is dependent on the *Positioning* service to work. Positions are showed on a map or in text in a list, see Figure 2.

### 2.1.2 Vehicle Management Services

The aim of Vehicle Management Services (VMS) is to keep track of the haulers’ vehicles. VMS are not treated in this thesis as the services only concerns administration staff. However these services will be mentioned in short to give the reader a broader view of the services provided by Scania FMS and to inform that these services together with *Positioning* and *Tracking* are the ones called “office level services” in this report.

*Vehicle Data* gathers automatically vehicle information such as driving distance, velocity, fuel consumption, and odometer readings. The information is presented on the office system.

*Trip Report* collects and presents detailed information about the vehicle data based on a specific vehicle and driver, during a specific trip. It can be used by the office staff to help drivers to improve their performance with respect to fuel consumption, brake use,
acceleration use etc, and can also be used to warn drivers that show bad behavior on the road by speeding.

Zone Alarm also called Geofencing sends an alert to the office system if a vehicle leaves or enters a predetermined geographical area. It can for example be used to notify when a vehicle is arriving to the next pick-up or drop-off point, or when leaving a specified country. The alert can also be sent to a designated cellular phone or email receiver.

Vehicle Alarm sends an alert when the theft alarm, alarm button, or a connected external alarm is triggered. The alarm message contains vehicle position and can be viewed on the office system or a designated cellular phone.

2.1.3 Driver Support
The aim of Driver Support is to improve the driver’s productivity, comfort and to encourage safe driving. Services within this function have a high degree of interactivity with the drivers and will thus be discussed later on the discussion chapter trying to answer questions like “what other services besides Messaging can be implemented into the C200 platform” and “how can they be designed to fit the different devices connected to the C200”. Here follows a brief description of the services.

Moving Map provides the driver with a map with the vehicle’s GPS-position and Scania workshops.

Navigation helps the driver to navigate and to plan the transportation route. It gives the driver turn-by-turn voice and virtual instructions down to street number level. Navigation can be integrated with Order Support providing the driver with the optimal path and final destination automatically.

Camera View lets the driver check the rear of the vehicle through the vehicle computer when reversing. It is useful to view close-ups and blind spots.

Telephone lets the driver make and receive calls through the vehicle computer. It provides hands-free and an address book.

Drive Time keeps track of the driver’s actual, daily, and weekly runtime, rest time, etc. It encourages the fulfillment of labor legislation, i.e. reminds the driver to comply with EU-regulations or similar. This service is partly replaced by the digital tachograph mentioned earlier (see description of Scania Driver Log).

2.2 Scania Fleet Management Portal
The Scania Fleet Management Portal (FMP) is an Internet service with which transport companies can make use of the concerned Fleet Management services. The system provides planners with a graphical map to visualize important fleet data. The planners can for example check (on a regular time intervals or whenever required) where their vehicles are, what route they are taking, what their destination is and how far they have left to go. With the service Zone Alarm operators can utilize the intelligent mapping software to establish a virtual fence around a planned route. If a vehicle surpasses the fence, the operator of the
Scania FMP is informed by email or SMS and can then prevent route deviations and thefts. Data sent between the vehicle computer and the office system is stored in Scania servers and can be accessed via the FMP. It can also be accessed, as well as sending data, through the transport company’s own system with the help of a gateway program that interprets data back and forth into a format recognized by the two systems (the Scania servers and the transport company’s own transport management system).

**Figure 2:** A screen shot of the Fleet Management Portal, the tracking page in 2009. The map to the right shows the geographical positions that a vehicle visited while the list to the left presents extra information for each point such as date and time, the driver’s name/alias, the vehicle’s ignition status and more.
3 VEHICLE PLATFORMS

Chapter 3 introduces the different vehicle platforms provided by Scania and gives an explanation of why the C200 platform was developed. The last part of this chapter presents the architecture and functionality of the C200.

The fleet management system is a set of components that need to work together for the whole system to work. Vehicle computers, that make use of the telecommunication part of the system, are a crucial part of that set of components. Without them it would not be possible to communicate with the vehicles (remotely) and thanks to them sophisticated programs can be made that ease the labor for drivers and office staff. From the driver’s point of view the vehicle computer needs to have some form of interactive interface to be a useful tool. That is not the case for the office staff since they only need that vehicle information is automatically gathered and sent by the platform. This chapter introduces the different Scania vehicle platforms available in 2009 and leads the reader into a technical description of the hardware and software architecture of the C200.

3.1 SCANIA VEHICLE INFORMATICS PLATFORM

Scania Vehicle Informatics Platform (SVIP) is the name used for vehicle computers provided by Scania that make use of the FMS telecommunication part and the vehicles’ Controller Area Network (CAN) system. The word informatics stands for the management and processing of data, information and knowledge [3] [4], and is one of the words that describes the main purpose of the vehicle computers. Scania Interactor 300 and 500 are two of the SVIPs. These rugged and versatile on-board computers are placed in the vehicle’s dashboard and interact with the users by means of a color touch screen, see Figure 3, or an included keyboard.

![Figure 3: Illustration of a dashboard with a Scania Interactor 300 (the touch screen to the right).](image)

The computers come with Microsoft Windows XP Embedded operating system and make use of the FMS standard, a common protocol for fleet management systems that has been agreed by leading manufacturers such as MAN, Scania, Volvo Trucks and more [5]. Scania Interactor 500 has a screen size of 10.4-inch, supports all Scania Fleet Management services mentioned in the previous chapter, and allows transport companies to install and run their own Windows-based applications. Scania Interactor 300 has a 6.3-inch screen, does not
support third party software, and supports all Scania Fleet Management services except for Telephone and Camera View.

### 3.2 Communicator 200

Even if the SVIPs are powerful computers with plenty of memory (some gigabytes) and capabilities of handling all kind of tasks, they can be too expensive for customers that only are interested in the investment of office level services\(^4\). By the use of modular design, Scania has the ambition of lowering the costs of their products.

"The basic idea underlying modular design is to organize a complex system (such as a large program, an electronic circuit, or a mechanical device) as a set of distinct components that can be developed independently and then plugged together. In systems engineering, modular design — or ‘modularity in design’ — is an approach that subdivides a system into smaller parts (modules) that can be independently created and then used in different systems to drive multiple functionalities. In a production context modular design in addition to reduction in cost (due to lesser customization, and less learning time) also offers flexibility in design. Modular design is an attempt to combine the advantages of standardization (high volume normally equals low manufacturing costs) with those of customization." [6]

The Communicator 200, also called C200, is a product manufactured to break apart the telecommunication part from the SVIPs. The result is a black box, see Figure 4. Its main purpose is already mentioned to enable communication between the truck and a remote user, in this case the office staff.

![Figure 4: The Communicator 200 hardware.](image)

---

\(^4\) See the definition of “office level services” in chapter 2.1.2 Vehicle Management Services
3.2.1 Hardware
The C200 consists of several circuits that are connected to a Central Processing Unit (CPU). The CPU’s task is to keep the system running and to handle all data exchange within the platform. To keep track of the vehicle, a Global Positioning System (GPS) receiver provides the C200 with vehicle position and speed. Data readings and writings from the local CAN bus is done through a CAN interface that supplies the platform with information about the vehicle ID, vehicle speed and other parameters available on the bus. The actual telecommunication unit is a General Packet Radio Service (GPRS) modem that handles all data traffic between the vehicle and the office. GPRS is a packet oriented mobile data service available to users of Global System for Mobile communications (GSM). GPRS embodies communication protocols (rules for the communication between different mobile devices) that make it possible for a user to use several time slots of a time frame that continuously is repeated [7]. GPRS can be used for services such as Short Message Service (SMS), Multimedia Messaging Service (MMS), and for internet communication services such as e-mail and World Wide Web access [8]. The C200 uses SMS messages for the initiating setup of the platform. Almost all other data communication are packets of data using the Internet protocols User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) through the GPRS module. The new version of Scania Interactor that was being developed in 2009 could be connected to the C200 making use of the FMS services provided by the platform through a dedicated Ethernet connection. Peripherals like the vehicle alarm system uses the digital inputs and outputs of the platform to trigger events and to initiate information transfer to the web portal. Figure 5 shows some of the platform’s connections. A Recommended Standard 232 (RS232), a serial binary data connection, was used for the development and debugging of software.

The C200 platform is equipped with a 32 bit CPU, a 16 megabyte RAM (Random Access Memory) with a bus width of 32 bit for application runtimes and runtime variables, and at least a 16 megabyte Flash memory for application and file storage. The memory amount is somewhat restricted compared to the SVIP’s (that had approximately 4 gigabyte) and will thus constitute a limitation in the development of an application prototype of the service Messaging.
3.2.2 Software Architecture

To facilitate future changes, the C200 is structured as layers, see Figure 6. The top part of the C200 in the figure shows the Application Layer containing the platform services. The service prototype **Messaging** was implemented in this layer. All applications are built in accordance to a software framework that will be described in the next section. To increase the applications portability and operating system independency Scania has developed an Application Programming Interface (API) containing classes and methods to communicate with the truck’s CAN system and with other applications. It has to be mentioned that it exists several levels of APIs in the API layer. E.g. a lower level of API is the one provided by the programming language C++’s Standard Template Library (STL) [9]. The Operating System (OS) layer runs Linux and has been chosen by Scania for several reasons. One significant reason is the low-cost solution (it is free) versus using proprietary software [10]. Other reasons are its favorable features of being upgradeable and portable. It has a lot of industry momentum, and is probably the next most common used embedded OS [11]. But the OS is not perfect and comes with some drawbacks. Linux requires a boot loader and it needs to load a lot of files from disk at boot time which leads to time-consuming startups. It is a large and complex operating system that requires a lot of knowledge. The kernel needs to be fine-tuned and configured to be adapted to the software and hardware environment. And the technical support is often not free. Underneath the OS layer a Board Support Package (BSP), i.e. hardware related software, enables communication between the hardware and the operating system. The bottom layer is the hardware (HW) layer containing the circuits described in the hardware section above.

![Figure 6: The C200 structured layers.](image)

3.2.3 Application Architecture

As mentioned previously, the service in the application layer are implemented in accordance to a software framework, see Figure 7. This framework (or application program design) aims to facilitate the development of the applications main structure and to make the system
more robust and unified. The benefits of a system architecture that promotes modular programming are decreased development time because separate groups would work on each module with little need for communication, product flexibility since drastic changes could be made to one module without a need to change others, and comprehensibility because the system could be studied one module at a time [12]. Properties like these improves the understanding of the entire system and makes maintenance cheaper in the long run.

Since every application has something to do with data exchange it can be adapted to a general architecture suitable for information flow. According to Figure 7, a C200 application consists of a single threaded framework (an abstraction layer that hides the specifics of the underlying OS) with one capsule and a collection of sockets attached to it. The capsule is the core object and contains the application’s main code. It is event driven and reacts to external events through client- and input-sockets. It can also react to internal events provided by timer-sockets when their timers expire. Events are converted by the sockets into method calls in the capsule. Every application runs in its own process and communicates with other services through an internal message based protocol. This method of communication detaches the applications from each other and makes them less vulnerable to failures in other applications. A common way for an application to operate is that data is inserted on the input-side, processed in the capsule, and distributed to clients that want the data on the client-side. Data may flow in any direction between the client-side and the input-side. Another kind of socket is the server-socket that produces client-sockets whenever a remote process connects to the server port.

Figure 7: C200 application architecture.
3.2.4 Communication Architecture
All data handling and communication is done using buffer-objects. A buffer-object is a scalable byte based buffer (in this case a C++ byte based vector) with methods to operate on it. This object is inherited by several objects that together form the C200’s internal message based protocol. Parameter-objects with system-wide identifiers are the most common objects that distribute data between processes. These are for example used to send information between the service Messaging (in the truck) and the FMP (the office). Message-objects comes with three types of attributes (configure, subscribe, and notify) and are used to establish a connection between two applications in the truck. Depending on the application’s task it can either subscribe on data from other applications or it can distribute data by notifying the other applications when a change has occurred. It is not restricted to either subscribe or notify but can do both.

3.2.5 Test Bench
To test the applications (the services) that are being developed in the platform a software test bench is provided for each service. A C200 test bench is an application that tests another application. It generates events to the test-object and monitors its behavior. This means that it only tests the application’s external functionality, i.e. the test-object can be seen as a black box that is being tested (see Figure 8).

![Figure 8: C200 test bench.](image)

Both applications (the test-object and its test bench) are run as separated processes and communicate with sockets making use of the internal message based protocol. The test bench’s execution flow is sequential.

3.2.6 Software
As the reader might have noticed, the programming language used to develop the C200 applications is C++. Data that need to be stored without being lost when the C200 is shut down (i.e. persistent data) are stored in databases with the help of the server-less database engine called SQLite [13].
4 Preliminary Study

Chapter 4 introduces the problem prior to the design-and-implementation-phase of the service prototype Messaging, namely the problem of how to connect the hand-held devices to the C200. It also discusses what previous works have been done that is similar in full or in parts to the Messaging prototype. At the end a communication solution was chosen out of the presented alternatives.

Since a part of this thesis project aims to investigate if TMS can be presented in hand-held devices such as laptops, PDAs, Sony PSP, mobile phones et cetera a preliminary study was carried out to investigate the connection alternatives between the different devices and the C200. An essential part of the investigation was to find out how the devices and the platform can or should communicate with each other, without making any greater modifications to the C200’s actual hardware and software solution. It was good practice to examine if there were any previous work done that could be useful in full or in parts to this work, because we would not want to reinvent the wheel. At the end of this chapter a communication proposal was chosen and later applied (see the next chapter). The chosen solution laid the foundation of how the service prototype was implemented.

4.1 Connection Alternatives

It has to be remembered that the C200’s main purpose was to break apart the telecommunication part from the Scania Interactor acting as a communication node between the different systems of the vehicle (e.g. the CAN-system and the Scania Interactor) and the office. The C200 is therefore not necessarily well-adapted to the task that this thesis project intends to investigate, which is creating a light version of the Scania Interactor in the C200 with the help of an arbitrary chosen hand-held device connected to it. This subsection of the chapter investigates the connection alternatives between the C200 and the hand-held devices.

4.1.1 Bluetooth

When taking a look at the C200’s product specifications (data sheets), documents that have been written when designing the hardware, and other reports discussing what components to incorporate into the product, a Bluetooth module could be found. Bluetooth is a wireless communications technology that is simple, secure, and wide spread. It can be found in billions of devices ranging from mobile phones and computers to medical devices and home entertainment products. Bluetooth technology is a low power and low cost solution that operates in the unlicensed industrial, scientific and medical (ISM) band at 2.4 to 2.485 GHz, using a spread spectrum, frequency hopping, full-duplex signal at a nominal rate of 1600 hops/s [14]. Unfortunately this technology was never built into the first version of the C200, due to hardware limitations and the gaining of some features when not using Bluetooth.

4.1.2 RS-232

A second alternative of communication is the RS-232 standard. In the context of the C200, the main use of this connection is to attach a personal computer for developing, maintaining, testing, and setup purposes. RS-232 is a standard for serial binary data signals commonly
used in computer serial ports, but have gradually been replaced in personal computers by USB for local communications [15]. Many handheld devices supports connections to the USB port [16] and can make use of it through an USB to RS-232 adapter.

### 4.1.3 Ethernet

A third alternative is the Ethernet port that was supposed to be used for the future Scania Interactor platform. Ethernet is a widespread standard for local area networks (LANs) to which computers, printers, and other devices can be connected using cable. Many hand-held devices, mainly modern laptops and smartphones, support the wireless LAN (WLAN) technology and can make use of the C200’s Ethernet port with the help of a wireless access point (AP), a wireless router or another device that makes this connection possible. The game console Sony PSP and the nowadays obsolete PDA’s [17] have a WLAN network card built-in and will also be able to connect to the C200. The most prevalent protocol used in LAN systems today is the TCP/IP protocol suite, i.e. TCP and UDP over IP and the like [18].

### 4.2 Previous Works

Before any decision was made about what type of connection to use, it was good to get acquainted with the C200 application layer to find out similarities between the Messaging prototype and the platform applications.

An application called WebPortal proved to be interesting since it communicates with its users (i.e. developers) via the Ethernet port. The application acts as a lightweight web server that listens on port 80 for incoming requests from clients, e.g. web browsers. It provides information about the C200-system such as internal fault codes emerged during runtime stored in databases, data from the CAN-system, GPS status among others. This application does not form part of the Scania Fleet Management Services mentioned in chapter 2, but exists to ease the labor for developers. How this application works internally will be explained in chapter 5. Parts of its communication technology were used in this work.

Another application that was of importance to this thesis is the CommHandler. CommHandler is a key application that handles all data traffic to and from the C200 in a secure and encrypted way. This application is only mentioned in general as a module that takes care of sending and receiving the messages to and from the service prototype Messaging. The Application Layer contains other services that in one or another way are connected to each other. None of the remaining applications were of great importance to this thesis and will therefore not be mentioned.

### 4.3 Solution Proposal

The most promising solution at the time seemed to be the Ethernet alternative. Since many hand-held devices support WLAN and given that the C200 already had a working example that used Ethernet, the solution proposal was to connect a Wireless Access Point to the C200, see Figure 9.
This communication proposal implied that the service prototype *Messaging* was either implemented as a web server or as an application in an existing web server in the C200, and therefore the name of this thesis.

The choice of the Ethernet solution was not only made for these reasons, but was also influenced by the following statements:

- Since Bluetooth was not built-in into the C200, a major modification needed to be done to the hardware solution to make use of this technology.
- It is preferred to keep the RS-232 connection for developing, maintaining, testing, and setup purposes.
- There were future plans in connecting a satellite modem to the RS-232 port, for occasions when the C200 does not have GSM/GPRS coverage.
- The Ethernet port is anyway going to be used by a device that makes use of the C200 services. Why not use the same port if that device is not present?
- And finally, Ethernet is a well-established technology that has survived several decades and that is gaining terrain in the connected world.
5 Prototype Implementation

Chapter 5 is about the design and implementation phase of the Messaging prototype. Since this service has many similarities with the WebPortal application, this chapter begins with an overview of the WebPortal’s communication architecture. After that, the technology requirements of the hand-held devices are determined, web servers and parsers are discussed, messages used between the C200 and FMP are designed, and two versions of the prototype implementations are presented.

As stated in the previous chapter, the service Messaging could either be implemented as a part of an existing web server or as a web server itself. To better understand how the WebPortal works internally, an overview of its communication architecture is presented.

5.1 The WebPortal Application

The WebPortal is an application tool that provides information about the C200-system such as internal fault codes, data from CAN-system, GPS-status, GSM signal strength etc. The information is accessed through a computer with a web browser, connected to the C200’s Ethernet port. Figure 10 shows a conceptual scheme of how data flows from a client to the web server application in the C200 and back to the client.

Starting with a client, a web browser opens a socket from the local machine (the hand-held device) to the remote host (the C200) associated by a Uniform Resource Identifier (URI). The socket requests to connect to port 80, if nothing else is specified, and sends data according to the HTTP-protocol. Data is received by a web server (the WebPortal) with the help of an HttpServerSocket-object, which in turn produces an HttpSocket for every new incoming

---

5 “A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent.” [38]
HTTP-request. A produced HttpSocket keeps track of the connection to the requesting client and lives under a short period until the request is responded or a timer expires. Requests are interpreted and processed in a WebHandler-object that generates a response, a web page or data content if implemented as a web service, to the client. The WebPortal application follows the application architecture described in chapter 3, with the addition of having the HttpServerSocket-object and HttpSocket-object included in the application framework to better handle external http requests. The application architecture enables the WebPortal to communicate with other applications in the application layer, and let WebHandlers make use of this communication possibility. Various kinds of fleet management services and developing tools that require interaction with its users (the truck driver or the C200 developer) can be implemented into WebHandlers.

5.2 Choice of Technologies

Hand-held devices and other clients that want to make use of the developing tools and the not yet implemented fleet management services in the C200, are required to support several technologies. These technologies are WLAN, TCP/IP, HTTP and HTML. It is beneficial to choose these technologies as a solution, since more and more hand-held devices are supporting connections to the internet through WLAN. Further technology requirements depends on how the Messaging application is implemented and what it is assumed that the hand-held devices should support. One alternative is that the application is implemented as a web service and the hand-held devices implements the user interface as a mobile app that utilizes the web service. A second alternative would be that the C200 provides the hand-held devices with complete web pages that can be presented in web browsers. In that case it is good if the devices also support CSS (Cascade Style Sheet), JavaScript, and AJAX (Asynchronous JavaScript and XML), even though it is not necessary. CSS would be used for the presentation of the web pages in different clients with varying screen sizes and resolutions, JavaScript to add interactivity to the pages, and AJAX to minimize the bandwidth usage. Since the existing office level services are the prioritized tasks to handle it is important for the Messaging application to minimize the work load on the C200 and to interfere with the office level services to a lesser extent. The benefit of supporting many clients without the effort of developing several mobile apps lead to the choice of the second alternative in this master thesis project.

5.3 Parsers and Web Servers

The HTTP-parser, used by the WebHandler, was at the moment of implementing the Messaging application at an early stage of its development and was consequently neither error-free nor complete. One thought was to use an already developed third party lightweight web server instead. Third party web servers tend to be less buggy and more reliable than an own developed web server (here referred to as the HTTP-parser). This

---

6 It is here referred to the services that do not belong to the office level services. Office level services are already implemented in the C200.
because they often are tested by various users and developers under a longer time where errors are reported and later corrected in the next version. There are many lightweight web servers available for use, all having their advantages and disadvantages. Some candidates that seemed to be interesting were fnord, mini_httpd, thttpd, EHS, among others. These candidates have small memory footprints\(^7\), are written in C/C++ and supports Common Gateway Interface (CGI) [19] [20] [21] [22]. But since one of the constraining questions stated in chapter 0 was to investigate if it is possible to integrate interactive services without modifying the actual server and communication solution, no effort was put in trying to find a winning candidate and therefore the existing HTTP-parser was used anyways.

## 5.4 Designing the Messages

**Message** is all about transmitting messages between the vehicle driver and the office personnel. But how are these messages transmitted, what information do they contain, and how are they encoded? There are many possible solutions and depending on the requirements set, the messages can be designed accordingly. All data traffic between the C200 and the office servers goes through the GSM/GPRS infrastructure. This wireless communication solution was chosen by Scania because of its global coverage, its beneficial roaming solution, and the possibility of using a dedicated and secured line. In fact it is the most widely used wireless technology in the world [23]. Since the data traffic is charged per transmitted byte, good care was taken when designing the messages.

### 5.4.1 Selecting the Information

The information that should be contained in the messages depends on what it aims to solve or be used for. In this master’s thesis project it was not taken any consideration of how the service would be used by the customers, but rather focused on showing how an Interactor service could be implemented in the C200. To simplify the task not all fields were used from the old Interactor service **Message**.

The fields that were included in the prototype were:

- Time stamp, the date and time when the message was transmitted.
- Message flags, e.g. an indicator to let the recipient know that the sent message is important.
- Subject, the message title.
- Message body, the actual message.

The fields that were excluded were:

- GPS-position, to see the location of the vehicle when the message was sent.
- Thread id, to be able to have a threaded conversation.
- Icon, to display different kinds of icons depending on the message type.

\(^7\) Approximately 93 KB for fnord 1.11, 140 KB for mini_httpd 1.19, 497 KB for thttpd 2.25b, and 4.39 MB for EHS 1.5.0.
5.4.2 Encoding the Messages with XML

One way of encoding the messages is by using XML (Extensible Markup Language). XML has the advantage of being able to create standardized and platform independent documents that are easy to use and understand, i.e. are self-explanatory for humans.

An example would be to encode the following message:

<table>
<thead>
<tr>
<th>08-07-06 15:02:39</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>Test message</td>
</tr>
<tr>
<td>This is a message</td>
</tr>
</tbody>
</table>

*Figure 11: Message example. The first row is the time stamp, the second row the flags, the third row the message subject, and the forth row the message body.*

Into the following XML-format:

```xml
<message>
  <timestamp>08-07-06 15:02:39</timestamp>
  <flags>1</flags>
  <subject>Test message</subject>
  <body>This is a message</body>
</message>
```

*Figure 12: XML-encoded message.*

The disadvantage of this kind of encoding is that it is text-based, i.e. takes 1 byte per character if coded with ASCII extended, and takes a lot of extra memory due to the metadata. Counting the characters of the not encoded message (Figure 11) gives 47 characters, i.e. 47 bytes, and the XML-encoded message (Figure 12) 136 characters, i.e. 136 bytes. Spaces for indentation, carriage return and line feed were excluded in the counting.

5.4.3 Compressing the message

XML is not an economic approach. To improve the communication performance of sending XML-encoded messages the (EXI) Efficient XML Interchange standard could be used. It is a W3C recommendation that enhances the communication performance of XML structured data between computers by reducing bandwidth requirements without compromising efficient use of other resources such as battery life, code size, processing power, and memory [24]. Another approach is to use Gzip (GNU zip). Gzip is based on the DEFLATE algorithm [25] which is a combination of LZ77 (Lempel-Ziv coding) and Huffman coding [26]. But according to an evaluation made by W3C the EXI format is consistently smaller than Gzipped XML [27], see Figure 13.
Compressing the example message in Figure 12 to EXI with compression enabled and schema-informed encoding enabled gave a message reduction from 136 bytes to 49 bytes. This means that the following XML schema (see Figure 14) must be shared by both the C200 and the office server to be enabled to communicate correctly.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="message">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="timestamp"/>
        <xs:element type="xs:byte" name="flags"/>
        <xs:element type="xs:string" name="subject"/>
        <xs:element type="xs:string" name="body"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 14: XML Schema to compress the example message.

5.4.4 A More Compact Version

A more compact way of encoding the message is by creating an application-specific protocol that is used to parse the message into binary strings (byte arrays in this case). This protocol focuses on economizing data. The two end users, the C200 and the office system, need to
implement this protocol to be able to understand each other. The protocol makes profit from the fixed message structure to compress the information in the data fields.

5.4.4.1 The Time Stamp
Starting with the time stamp, the number of characters in the example message, Figure 11, i.e. 08-07-06 15:02:39, uses 17 characters = 17 bytes = 136 bits. These characters can instead be encoded with the 32-bit UNIX time stamp, which is a way to track time as a running total of seconds since January 1, 1970 at UTC [28]. The time stamp for the date given in the example is 1215356559 seconds, which is represented by 4 bytes instead of the original 17 bytes. The problem with the 32-bit UNIX time stamp is that it will cease to work on January 19, 2038 due to a 32-bit overflow. To get rid of this problem a 64-bit UNIX time stamp can be adopted (that ceases in the year 292 billion). In this project an application specific format is used instead which gives better data compression. To not have the problem of representing years, a two digit format will be used as in the example. It is assumed that hundred years is sufficient time for the message to loose importance, even decades may suffice. This gives us the possibility to encode the year, month, day hour, minute, and second separately, all with different fixed amount of bits for their representation. For example the year can be encoded with 7 bits, which is sufficient to represent numbers between 0 and 99. Further on the months, 1 to 12, are represented by 4 bits, the days, 1 to 31, by 5 bits, and the minutes and seconds, 0 to 59, by 6 bits each. This gives us a total of 7 + 4 + 5 + 5 + 6 + 6 = 33 bits. Since we are restricted to encode these bits into whole bytes, 5 bytes are required, i.e. 40 bits. The time stamp in the example will look like in Table 1.

<table>
<thead>
<tr>
<th>Decimal format</th>
<th>Binary format</th>
</tr>
</thead>
<tbody>
<tr>
<td>08 (year)</td>
<td>0001000 (7 bits)</td>
</tr>
<tr>
<td>07 (month)</td>
<td>0111 (4 bits)</td>
</tr>
<tr>
<td>06 (day)</td>
<td>00110 (5 bits)</td>
</tr>
<tr>
<td>15 (hour)</td>
<td>01111 (5 bits)</td>
</tr>
<tr>
<td>02 (minute)</td>
<td>000010 (6 bits)</td>
</tr>
<tr>
<td>39 (second)</td>
<td>100111 (6 bits)</td>
</tr>
</tbody>
</table>

Table 1: Conversion of the time stamp example from a decimal to binary format.

<table>
<thead>
<tr>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
<th>Byte 4</th>
<th>Byte 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001000</td>
<td>011</td>
<td>00110</td>
<td>01111</td>
<td>000</td>
</tr>
</tbody>
</table>

Table 2: Illustration of the encoded time stamp. The binary strings from Table 1 are concatenated one after the other starting with the year. 'x' means that the bits are not defined.

As shown in Table 2, 5 bytes are needed to represent the time stamp. That is 1 byte (or actually 1 bit) more than the 32-bit UNIX time stamp, but 103 bits less than the original example message in Figure 11.

5.4.4.2 The Flags
The remaining 7 bits of byte 5 in Table 2 can be used for the flags. What the flags stand for is up to the developer to decide. In this version only two flags was used (2 bits), one that indicates if a message is important or not and the other to tell if the message subject and body is compressed or not. The remaining 5 bits were kept undefined.
5.4.4.3 The Message Subject and Body

There are a plethora of lossless data compression algorithms that can be used to encode the message subject and body. Some examples are Huffman coding [29], Context Tree Weighting (CWT) [30], Burrows-Wheeler transform (BWT) [31], Dynamic Markov compression (DMC) [32], Lempel-Ziv (LZ77 and LZ78) [33] [34], Lempel-Ziv-Welch (LZW) [35], Deflate (a combination of LZ77 and Huffman coding) [26], and many more. An investigation of which compression algorithm that renders the best result, i.e. best compression ratio and computational performance, for a written language with presumably short messages was not carried out. Instead an arbitrary chosen compression algorithm was chosen to test the concept. The compression algorithm that was tested in this prototype was Huffman coding. Compressing the subject and the message body in Figure 11 gave a reduction from 29 bytes down to 13 bytes. This successful compression by 55% is not always the case, sometimes the compression may even increase the size of the message. In those cases, it is better to send the message subject and body uncompressed. To tell the parser whether the message is compressed or not the flag (the bit) in the previous section (see 5.4.4.2) is used. The parser must also be able to separate the subject from the message body. To do that an end mark of some kind is needed at the end of the subject. Another alternative is to restrict the subject length to an upper limit and adding that extra information about the subject length in front of the subject. In this prototype the user was limited to write 255 characters in the subject field and therefore an extra byte was aggregated in front. Here follows an example of the entire message in hexadecimal form, uncompressed (see Figure 15) and compressed (see Figure 16).

```
10 E6 78 53 C0 0C 54 65 73 74 20 6D 65 73 73 61 67
54 68 69 73 20 69 73 20 61 20 6D 65 73 73 61 67
65 65
```

*Figure 15: The example message uncompressed represented in hexadecimal form.*

```
10 E6 78 53 E0 AE 84 BE A7 3D 0E B3 58 65 F5 39
C0 36
```

*Figure 16: The example message compressed represented in hexadecimal form.*

The message size of the example message is reduced from its originally 47 bytes down to 35 bytes with the message subject and body uncompressed and down to 19 bytes if compressed.

5.5 Graphical User Interface

When it comes to the Graphical User Interface (GUI) many disciplines are involved such as information architecture, graphic design, human-computer interaction, interaction design, user-interface engineering, user experience design, among others. Many of these disciplines overlap and are difficult to separate one from another. To take all these disciplines into account is out of the scope of this master thesis’ intentions, which is to show if it is possible to implement transport management services in the C200, independently of the GUI’s quality. It was therefore not put any greater efforts in finding an optimal GUI candidate. Even though web-mail layouts and graphical setups of traditional SMS services found on
mobile phones are probably not the best adapted GUIs for users in a moving environment, an imitation of those two GUIs was made. The reason of doing this was due to the belief of that most users are familiar with the use of those services and will not find it difficult to understand how to use the prototype.

5.6 Security Issues

Another dimension to consider when developing an application for a driver whose primary purpose is to handle the vehicle safely on the road is to design the application in such way it not distracts the driver while performing critical tasks. Studies have shown that the risk of being involved in a crash or a near-crash situation increases dramatically when using devices that have a high degree of visuomanual interaction [36]. Texting while driving is one of those tasks that steal attention producing increased eye-off-road time. To mitigate the effects of driver distraction the Messaging application could be implemented in such way it is locked out while the vehicle is in motion. The developed prototype did not implement this feature. Instead it was left as a suggestion for further development.

5.7 First Prototype

The first prototype is a standalone module that is basically a modification of the WebPortal with the extension of a connection to the CommHandler. To avoid conflicts in the communication between a web browser and the Messaging prototype, the WebPortal and its integrated programs were removed. Figure 17 shows an overview of the involved components in the C200 when sending a message from a web browser to the office and vice versa.

![Diagram](image)

*Figure 17: An overview of the first prototype*

A message travels from a web browser into the Messaging application where it is stored in a database and thereafter sent to the CommHandler. The CommHandler takes care of encrypting the message and transmits it to the office (i.e. Scania communication servers) where the message is stored in a database. Figure 18 shows the result of the developed GUI.
Figure 18: Snapshot of the first Messaging prototype

5.8 SECOND PROTOTYPE

The next step in the development of the prototype was to incorporate the application into the WebPortal. This was solved by implementing web handlers in the WebPortal and then moving the Messaging logic into a web handler (see Figure 10). The concept of having web handlers was to improve the extendibility making it easier to add new functionality (new applications) to the WebPortal. With the rise of web handlers a routing function was also implemented so the http-requests could be handled by the right application. An overview of how the second prototype looks like can be seen in Figure 19.

A message travels from a web browser into the WebPortal application where it is routed to the to the requested web handler, in this case the Messaging application. When the Messaging application receives the message the same flow as in the first prototype is used. Figure 20 shows the WebPortal GUI.
Figure 20: Snap shot of the WebPortal application with Messaging

The GUI consists of a menu with the available applications to the left, some icons showing the connection status (the GPRS, GPS, and CAN) to the right, and in the middle a frame for the presentation of the chosen application. The top of the page contains a banner with the Scania logotype.

5.9 FMP-SIDE TEST APPLICATION

To test the prototype in its whole context a standalone C# application was developed on the office side (on the local computer). It was developed to see if new messages could be sent to the vehicle and to ensure that the messages sent from the vehicle were received and stored in the database at the Scania communication server. Figure 21 is a snapshot of the test application.

Figure 21: Snapshot of the test application at the office
6 Results

This chapter presents the result of the final prototype tested on a selection of hand-held devices and web browsers available in the year 2009. It also makes an assessment of which services among transport management services and driver support that are appropriate or inappropriate candidates to be implemented into the C200 with respect to this thesis constraints and prototype result.

The chosen hand-held devices to test the prototype were a laptop, a smart phone, and a PSP. In common they had support for WLAN but differed in screen size, memory capacity, audience focus, user interface, etc. Despite the differences, the goal was to show that the solution worked satisfactorily from a holistic point of view, i.e. it should be possible to send and receive messages regardless of the chosen terminal. Here on follows a technical description of the tested clients and how they managed to run the prototype.

6.1 Laptop

The first device tested with the prototype was a laptop running Windows XP, see Figure 22.

![Figure 22: The laptop in which the prototype was executed in. Manufacturer: Acer, model: TravelMate 5520, CPU: AMD Athlon 64x2 (1.7 GHz x 2), HDD: 160GB, SDRAM: 1GB, Display: 1280x800 pixels.](image)

The prototype was tested with three different web browsers. Those were (see Figure 23) Internet Explorer 7, Mozilla Firefox 3, and Opera 9.6, all having support for CSS, JavaScript, and AJAX. The results were that the Messaging service worked well with all three browsers.
Figure 23: The three web browsers (IE7, Firefox 3, and Opera 9.6) the prototype was tested on.

Even though a normal use case scenario would not involve a laptop in the truck, the test could be considered valuable since it could be compared with other products that have similar characteristics. One such product was the supposed next generation of the Scania Interactor platform. But the development of the new Scania Interactor was cancelled due to product strategy changes and because of the emerging market of third party devices, e.g. like today’s smartphones and tablets, that could outcompete the Interactor in the future.

6.2 PORTABLE SONY PLAYSTATION

The second device tested with the prototype was a PSP. With its 480 x 272 pixels the page content was displayed nice and tidy. One problem with the NetFront (the micro web browser) version that was tested at the time was its lacking support for AJAX. Therefore when the Messaging application was chosen from the menu an XMLHttpRequest (i.e. an AJAX-request) was launched causing an error on the page. The error showed as a collapsed menu and a collapsed main frame, see Figure 24. So the prototype failed to function correctly with the device. One way of solving the problem was by upgrading the browser with a newer version. According to NetFront’s official homepage in 2009 the successor version 3.5 had support for AJAX. Another solution would have been to implement the WebPortal and the Messaging application without the use of AJAX. In any case the first alternative was preferable to minimize the work load on the C200.
6.3 Mobile Phone with LAN Support

The third and last device to run the prototype was a mobile phone with WLAN (see Figure 25). It ran the operating system Symbian and the web browser Opera mobile. It had a 240 x 320 pixel screen and was the smallest display used for the tests. As can been seen in the figure the GUI looked different from the previously presented examples. By using the CSS media type property set to “handheld” the presentation was customized and compressed as much as possible to fit the small screen. The top part contained the Scania logotype, and beneath the menu followed by the main frame where the services were presented. The bottom part contained the connectivity status bar. There were still work to do to the prototype since the GUI was not error free. For example the flags indicating if a message was important or not were not present and the signal strength of the connectivity status disappeared. Despite these minor issues, the prototype worked properly.
Figure 25: The prototype tested on a smart phone. Manufacturer: Sony Ericsson, Model: W960i, CPU: ARM 9, 32-bit RISC processor (208 MHz), RAM: 128 MB, ROM: 256 MB, Display: 240 x 320 pixels.

6.4 Implementation Assumptions of Other Services

The proof-of-concept of being able to implement a prototype of the interactive service Messaging and presenting it on an arbitrary chosen hand-held device leads us to the standpoint of being able to make assumptions of what other services with similar technical characteristics could be implemented into the C200 in an analogous manner.

Among the transport management services; Order Support and Driver Log were fully capable and technically appropriate candidates to be implemented since they bear a resemblance to the Messaging application in terms of having small memory requirements and being similar from a technical point of view. Order Support is simply an extension of the service Messaging with the task of receiving orders from the office, i.e. specialized messages with job details, and sending back progress status of the orders. Driver Log on the other hand has been replaced by a digital tachograph unit and is therefore no longer meaningful to be implemented in the C200. The same applies to the service Driver Time.

When it comes to the driver support services Moving Map and Navigation, which are dependent of a map, they make no good candidates due to large memory requirements or heavy usage of the GPRS communication solution. I.e. map information is either needed to be retrieved continuously from a map server while the vehicle is in movement causing expensive communication costs when roaming or the entire map needs to be store in the C200 which was not possible in the first version of the C200 due to memory limitations. The service Camera View and Telephone were not supported by the first C200 hardware and could therefore neither be implemented without modifying the hardware.
Chapter 7 discusses the result and presents the conclusions from this thesis.

A lot have happened since this study was conducted. Several versions of the Scania Communicator have been released, the development of the new Scania Interactor platform was discontinued, and new fleet management services have emerged. Smart phones have evolved into full-fledged personal computers and the connected world has become increasingly accessible. The Scania communicator was standardized in the year 2011 meaning that every new Scania vehicle (trucks and busses) produced from factory comes with the communicator platform installed and makes it easy for customers to subscribe and sign up for the fleet management services. With the absence of the old Scania Interactor platform, Scania has lost some of the services that were available under the Scania Interactor era. Some of the services, like the map services, have reappeared in third party devices like smartphones, tablets, GPS devices and the like, while others such as Messaging and Order Support have vanished.

The approach of implementing the interactive transport management service Messaging into the C200 and presenting it on an arbitrary chosen handheld device was successfully achieved. The goal was to propose a solution on how this service could be implemented taking into consideration the stated questions in paragraph 1.2 Problem Statement. One and maybe the most significant constraint that affected the output of this work was the limiting factor of not making any greater modifications to the C200’s actual hardware and software solution. It was therefore a crucial step in the process to investigate the possible alternatives and thereafter to choose a promising approach. This resulted in a web server implementation of the service. The solution made use of the standardized web communication protocols and got advantageous when it comes to reach out to as many handheld devices as possible owing to that more and more devices were gaining support to the Internet through WLAN.

The proof-of-concept of being able to implement Messaging gave us a good indication of what other services were feasible to implement. Among the transport management services, Order Support and Driver Log were fully capable and technically appropriate candidates to be implemented since they bear a resemblance to the Messaging application in terms of having small memory requirements and being similar from a technical point of view. Order Support can be seen as an extension of Messaging with the difference that it receives specialized messages with job details and where the truck driver sends back messages with progress status to the office. Driver Log on the other hand, even if it is feasible to implement in the C200, has lost its significance since it has been replaced by a digital tachograph unit and a service around it that register the vehicle’s and driver’s working hours and activities automatically and sends the collected data to a remote storage. The data is later accessible through the Scania Tachograph Portal.

When it comes to driver support services, which aims at helping truck drivers in their work, they have a high grade of interactivity with the laborer. Moving Map for instance provides
the driver with the vehicle location on a map displayed on a screen. The position and the displayed map are updated continuously. The map information required more memory than the first version of the C200 could provide, i.e. if the whole map was to be stored internally. Of course, this could be remedied by adding more memory to the platform, but then we would have altered the C200 hardware. Map information could also be retrieved remotely from a map server, but the solution has the drawback of getting expensive if lot of information is transferred through the established GPRS communication solution when roaming. The same applies to the Navigation service since it also is dependent of a map. Neither of the services Moving Map nor Navigation were suitable candidates to be implemented into the C200 at the time this thesis was conducted. The service Camera View and Telephone were not supported by the C200 hardware and could therefore neither be implemented without modifying the hardware.

If the restriction of not being allowed to make any greater modification to the C200’s hardware and software solution could be ignored, new interesting services could be developed. One such service would be to have support for improved daily inspection of the vehicle. The idea is to have the vehicle’s lights, turn signals, brake light, reverse light, horn, windshield wipers and so on accessible and controlled through the handheld device when the vehicle is standing still. In that way the driver can more easily do the daily inspection of the vehicle. It would also be good if different type of fluid levels could be monitored such as the vehicle’s coolant level, washer fluid level, oil level, and so on. The daily inspection could be designed as a software wizard, guiding the driver throughout the task.

Just the knowledge of being able to attach hand held devices to interact with the vehicle opens up the opportunity of inventing new innovative and sophisticated services.
8 Future Work

Chapter 8 discusses how the developed communication solution can be improved.

It is good to keep in mind that the elaborated prototype was developed and adapted for the first version of the C200. This work was an attempt in trying to rescue some of the interactive services from the Scania Interactor platform keeping its original design with respect to its original targeted audience. Messaging as it was developed in 2009 aimed at improving the communication between the office personnel and the driver throughout a device mounted on the vehicle. Nowadays, a majority of the users owns a smartphone that could be used instead, replacing the Messaging solution as a service through the C200. Scania is at the writing moment (July of 2014) finishing a new version of the service that includes connecting smartphones and tablets to the Fleet Management Portal. This approach detaches the drivers from the vehicle creating a different use of the service. Still, even if the C200 Messaging service is becoming obsolete, the proposed communication solution could be useful and further developed to better support other yet not invented services that need some form of communication with the vehicle. With the introduction of a built in wireless access point and a Bluetooth unit in the newest versions of the C200 the communication solution could be restudied to investigate the pros and cons between these two alternatives. If the WLAN solution proofs to be the better one, the web server solution presented by this prototype should be questioned and further studied. Can third party light weight web servers cope better with technology advancements and standardizations? How well do they perform and scale? And how secure are they with respect to hacking?
REFERENCES


