

| | |
|-------------------------------|---|
| Project number: | 35-2-38-08 |
| Project acronym: | NS FRITS |
| Project title: | North Sea Freight and Intelligent Transport Solutions |
| Thematic priority: | Improving Accessibility to places in the North Sea Region |
| Area of intervention: | To promote the development of efficient and effective logistics solutions |
| Start date of project: | 01/01/09 |
| Duration: | 36 months |

| | |
|---|--|
| Deliverable reference number: | N/A |
| Deliverable title: | NS FRITS System Services and Functionalities Description |
| Version: | 1.0 |
| State within Consortium: | DRAFT: - FOR APPROVAL: - APPROVED: X |
| Lead contractor of this deliverable: | Avanti Communications |
| Other contributing contractors: | Volvo, Avonwood |



Project co-funded by Interreg IVB North Sea Region Programme (2007 – 2013)

DISSEMINATION LEVEL

| | | |
|-----------|---|----------|
| PU | Public | |
| PP | Restricted to other programme participants (including Interreg IVB Services) | |
| RE | Restricted to a group specified by the consortium (including Interreg IVB Services) | X |
| CO | Confidential, only for members of the consortium (including Interreg IVB Services) | |

Contents

| | | |
|----------|--|-----------|
| 1 | INTRODUCTION | 3 |
| 1.1 | PROJECT SCOPE | 3 |
| 1.2 | PURPOSE OF THE DOCUMENT | 3 |
| 2 | REFERENCE DOCUMENTS | 4 |
| 2.1 | REFERENCE (R) AND APPLICABLE (A) DOCUMENTS..... | 4 |
| 2.2 | ABBREVIATIONS | 4 |
| 3 | DOCUMENT OVERVIEW | 5 |
| 3.1 | EXECUTIVE SUMMARY | 5 |
| 3.2 | DOCUMENT LOCATION IN PROJECT ACTIVITIES | 5 |
| 4 | LIST OF FUNCTIONALITIES | 6 |
| 4.1 | LOCATION- BASED TRANSPORT INFORMATION PROVISION (INCLUDING TRAFFIC AND WEATHER)..... | 6 |
| 4.1.1 | <i>Functionality concept description</i> | 6 |
| 4.1.2 | <i>Benefits of the functionality</i> | 6 |
| 4.1.3 | <i>Useful analogy.....</i> | 6 |
| 4.1.4 | <i>Who will find this functionality useful?.....</i> | 6 |
| 4.1.5 | <i>Additional comments</i> | 7 |
| 4.2 | FLEET MANAGEMENT | 7 |
| 4.2.1 | <i>Functionality concept description</i> | 7 |
| 4.2.2 | <i>Benefits of the functionality</i> | 7 |
| 4.2.3 | <i>Who will find this functionality useful?.....</i> | 7 |
| 4.2.4 | <i>Additional comments</i> | 8 |
| 4.3 | DISTANCE-BASED TRANSPORT INFORMATION PROVISION | 8 |
| 4.3.1 | <i>Functionality concept description</i> | 8 |
| 4.3.2 | <i>Benefits of the functionality</i> | 8 |
| 4.3.3 | <i>Who will find this functionality useful?.....</i> | 9 |
| 4.3.4 | <i>Additional comments</i> | 9 |
| 4.4 | NS FRITS AGREEMENTS | 9 |
| 4.4.1 | <i>Functionality concept description</i> | 9 |
| 4.4.2 | <i>Benefits of the functionality</i> | 9 |
| 4.4.3 | <i>Useful analogy.....</i> | 10 |
| 4.4.4 | <i>Who will find this functionality useful?.....</i> | 10 |
| 4.4.5 | <i>Additional comments</i> | 10 |
| 5 | APPENDICES | 11 |

1 INTRODUCTION

1.1 PROJECT SCOPE

The NS FRITS (North Sea Freight Intelligent Transport Solutions) project aims to improve competitiveness and the quality of life in the North Sea Region by addressing freight volumes, inter-modality, congestion, emissions and associated security threats by advancing coherence for the freight supply chain through an innovative process that, when fully operational, will provide a dedicated information architecture capable of informing drivers and other major actors in the freight logistics supply chain of changing circumstances within the region's major transport corridors and between transport modes.

1.2 PURPOSE OF THE DOCUMENT

This document is a summary of the functionalities and capabilities of the NS FRITS system. It is of interest to all stakeholders technical and non-technical alike. The capabilities of the system have been particularly summarised and simplified in this document for brevity and ease of comprehension.

2 REFERENCE DOCUMENTS

2.1 REFERENCE (R) AND APPLICABLE (A) DOCUMENTS

- [1] Concepts document - A
- [2] Results of user survey - A
- [3] User requirements - A
- [4] System requirements – A
- [5] Pilots description and approach – A

2.2 ABBREVIATIONS

| | |
|-----|--|
| ETA | Estimated Time of Arrival |
| HGV | Heavy Goods Vehicle |
| ICT | Internet and Communications Technology |
| POI | Point Of Interest |
| TIS | Transport Information Services |

3 DOCUMENT OVERVIEW

3.1 EXECUTIVE SUMMARY

This document provides a summary of the NS FRITS system capabilities. It emphasises the fact that the NS FRITS system is a toolbox from which users can choose the specific capabilities that suit their needs. The document breaks down the services provided by the NS FRITS system into the following;

- Location-based transport information provision (including traffic and weather)
- Distance-based transport information provision
- Fleet management
- NS FRITS Agreements

The document does not delve into details about the services but rather describes the functionalities including benefits that each service provides - this is deliberate so that the document is easily comprehensible by technical and non-technical stakeholders alike.

For each service or functionality, the rationale for its inclusion in the NS FRITS system is described, followed by the benefits that the functionality provides to users, then example entities in the transport logistics sector that could take advantage of the functionality are described, and finally some additional information is provided. This format is replicated for each of the functionalities. The document concludes with an appendix that constitutes photos taken whilst the system was being tested over several weeks in and around London.

It is worth pointing out that for some of the functionality described in this document, similar offerings exists in the ITS market place but these are usually prohibitively expensive for the smaller entities in the freight logistics transport industry at which the NS FRITS system is targeted. The NS FRITS system is unique in that it has utilised best-of-breed free and open source tools to develop an affordable Transport Information Services (TIS) platform that is deployable on a variety of hardware devices and designed to meet specific user needs.

3.2 DOCUMENT LOCATION IN PROJECT ACTIVITIES

This document is not an official deliverable as such but constitutes output from WP4 activities that fit logically into WP5. Its location in the context of other project activities is shown in Figure 11 below;

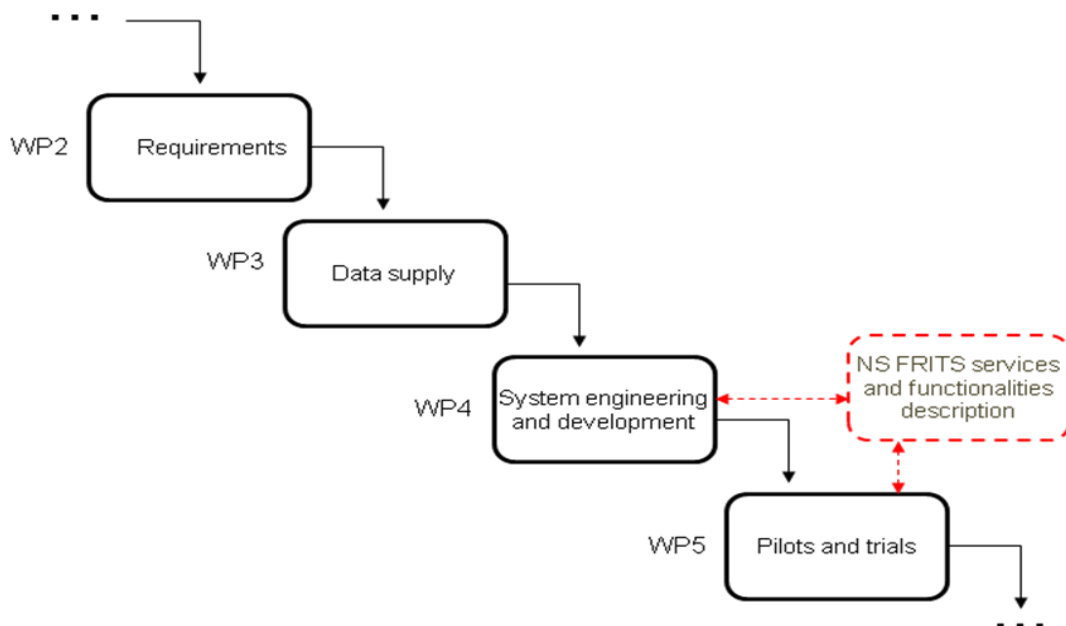


Figure 1: Document location in project activities

4 LIST OF FUNCTIONALITIES

4.1 LOCATION- BASED TRANSPORT INFORMATION PROVISION (INCLUDING TRAFFIC AND WEATHER)

4.1.1 Functionality concept description

The NS FRITS system provides location-based Points Of Interest (POI) information sourced in real time that HGV drivers can find helpful and practical. The points of interest are categorised as follows;

- Traffic information (including road accidents/incidents, weather and obstructions)
- Crime hotspots
- Safe/secure parking locations including truck stops
- Customs information
- Ferry information

The user typically selects the categories that they are interested in when they start the application on their mobile device. Information about the selected category is then provided to them (by text-to-speech) based on their location i.e. whilst they are driving. The information is overlaid on a map at the appropriate location using descriptive and intuitive icons e.g. a relatively insignificant road obstruction that does not impact much on traffic is shown with a small icon, whereas one with major impact is shown with a bigger icon etc.

The user also has the option of searching for POI information at any location. Communication with the mobile device can be performed hands-free i.e. using text-to-speech and speech-to-text.

All information is provided to the driver in their preferred language where available.

4.1.2 Benefits of the functionality

Information provided by the NS FRITS system is sourced from reputable national transport and traffic management authorities and thus is a one-stop-shop that satisfies a truck drivers POI information needs. The information serves as a decision support tool enabling drivers to make more informed choices.

Cost of freight crime targeting commercial vehicles and other loads within the European Union alone is estimated at about 8.2 billion Euros¹. Knowledge of safe places to park is very sought after by truck drivers². Lost revenue by the operator, and time by the driver resulting from time spent driving out-of-route miles to avoid undesirable incidents/locations or in search of POI is reduced. Plus, all information provided by the NS FRITS system is retrieved in real time meaning that the truck driver gets up to date information. Routing information is provided that takes into account vehicle dimensions including weight and load restrictions as well as toll calculations etc.

In a nutshell, this functionality eliminates travel woes for trucks driver by empowering them with requisite transport information thus ensuring that they make informed choices.

4.1.3 Useful analogy

GPS car navigation system specifically for truck drivers – Figure 2 below shows NS FRITS POI information on end user devices.

4.1.4 Who will find this functionality useful?

Freight operators with only a few trucks including one man operators that require the competitive advantage afforded by an HGV optimised decision support tool

¹ Europol, *Cargo theft report*, 2009, The Hague

² The NS FRITS consortium and stake holder organisations, *NS FRITS driver survey results*, 2009, U.K.



Figure 2: NS FRITS application on smartphone and tablet devices

4.1.5 Additional comments

The functionality can be used in isolation or in conjunction with all the other functionalities of the NS FRITS system. However, in the trials, there is always the risk that if the information source is not updated frequently, users could get disillusioned by the system if the same information is provided over and over again each time they use the application.

4.2 FLEET MANAGEMENT

4.2.1 Functionality concept description

Real time visibility of assets (driver and cargo) is important in freight logistics transport. Transport of valuable and high value cargo typically needs to be tracked to ensure safe delivery at the destination or to ensure safety of the population in the vicinity of the transport trajectory. It is estimated that empty journeys constitute over 35% of national transport and about 30% of international transport³. With such startling statistics, a system that enables a fleet dispatcher/manager to see where all his/her trucks are at all times, and allows he/she to communicate with them without contravening lawful usage of in-vehicle systems is invaluable.

NS FRITS provides just such a system.

4.2.2 Benefits of the functionality

A freight operator using this functionality of the NS FRITS system will be able to see where all his/her trucks are at all times thus optimising fleet utilisation since orders can be assigned to trucks as a function of location. This results in direct benefits to the freight operator that include better management of tight schedules, improved accountability, safety and security of mobile assets and personnel, and ultimately, improved customer service and enhanced revenue.

Additional benefits also include reduced fleet operation cost by lessening unauthorised vehicle use, idling, or out-of-route situations as well as more affordable insurance premiums.

This functionality enables location transparent management since the vehicle fleet to be managed from anywhere, anytime.

4.2.3 Who will find this functionality useful?

Freight operators with only a few trucks that cannot afford the bespoke expensive fleet management systems used by their larger competitors – Figure 3 below shows the NS FRITS fleet management and smartphone interface.

³ European Commission, *Freight Transport Vademecum*, 2009, Belgium



Figure 3: Fleet management and smartphone interface

4.2.4 Additional comments

This functionality can be promptly customised and enhanced to meet specific needs or requirements. The functionality can be used in isolation or in conjunction with all the other functionalities of the NS FRITS system.

4.3 DISTANCE-BASED TRANSPORT INFORMATION PROVISION

4.3.1 Functionality concept description

One of the major objectives of the NS FRITS project is to enable enhancement of the information provided to Heavy Goods Vehicle drivers, the majority of who travel across international borders where traffic, regulations, and language can become incomprehensible.

Furthermore, some entities involved in freight logistics transport would like to provide very specific and time-dependent information to users of their services but do not have the requisite ICT tools to do so and thus resort to inefficient mechanisms like print media i.e. brochures/flyers etc.

4.3.2 Benefits of the functionality

This functionality provides the ICT tools to enable any entity involved in freight transport logistics to provide information to users of their services. The information can include images, documents, sound files and links to other web resources etc. What makes this piece of functionality unique is that it enables the information provider to specify exactly when he/she wants the user to receive the information. For example, a ferry operator could provide information such as documents that will be needed, procedures, driving instructions etc. A driver who is scheduled to use the ferry will get the information through his/her NS FRITS enabled smartphone or in-cab device – information on required documents will be provided when he/she starts up the application at home, procedures will be provided when he/she is about a few miles close to the destination and driving instructions will be provided when he/she is even closer. What is even better is that the NS FRITS system can translate the information such that the driver gets it in their preferred language.

Other transport authorities including data sources/providers that have information in legacy formats that is not particularly computer friendly can also use this functionality to provide information to users.

4.3.3 Who will find this functionality useful?

GCD Glomb (a container terminal operator in Germany- see below) could find this functionality desirable as they can easily utilise it to notify drivers of the correct in-gate queue to get into etc.

This functionality was successfully tested with the Norwegian customs at Svinesund who have problems with foreign HGV drivers not understanding local regulations and procedures – Figure 4 below shows the web interface used to enable this functionality.

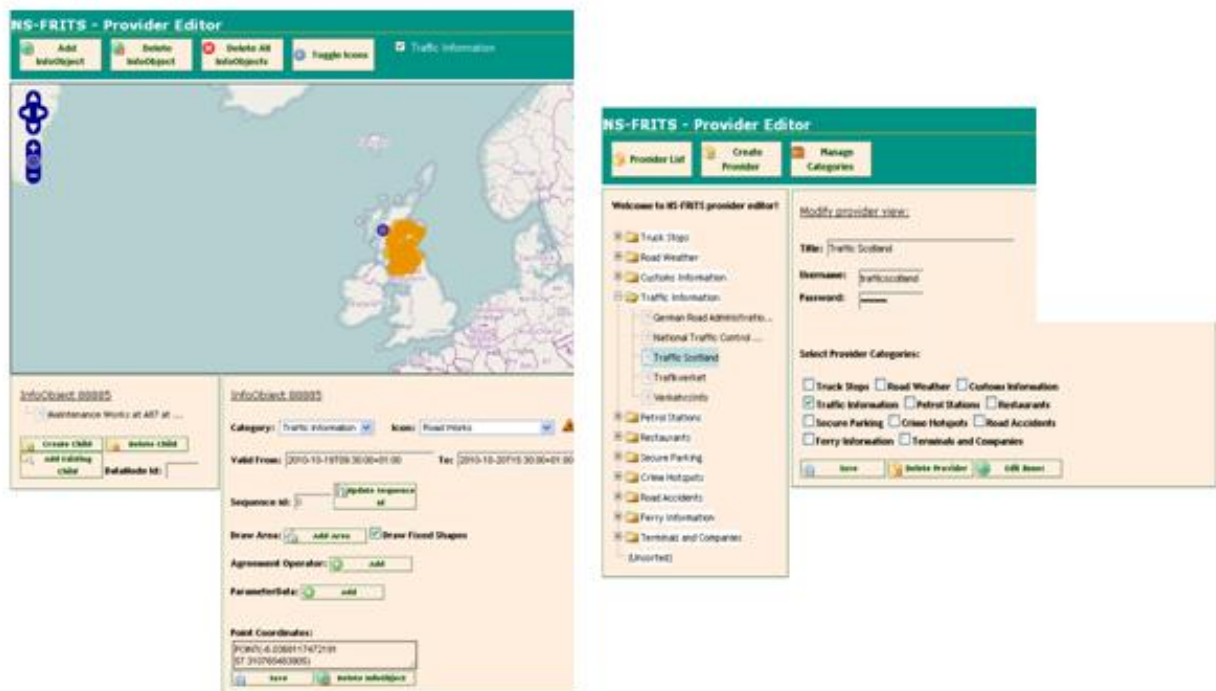


Figure 4: NS FRITS Information provision interface

4.3.4 Additional comments

This functionality can be used by any entity in the transport industry wishing to use ICT tools to broadcast specific time-dependent information to a large group of users in different languages. The functionality can be used in isolation or in conjunction with all the other functionalities of the NS FRITS system. It can be customised to meet additional user needs.

4.4 NS FRITS AGREEMENTS

4.4.1 Functionality concept description

The concept of Agreements in NS FRITS provides a generic mechanism that enables two or more actors in the freight logistics supply chain such as a driver and a goods terminal operator, who have agreed on a transaction at a specific time to keep each other informed about deviations.

4.4.2 Benefits of the functionality

The mechanism provides the goods terminal operator with real time visibility of the Estimated Time of Arrival (ETA) of each driver. This helps the goods terminal operator plan operational processes and predict work load since the NS FRITS agreements service can provide information on which driver is scheduled to arrive at what time. The service can also help the driver by providing him with information about delayed containers so that he does not go to the terminal before his goods are ready for pickup. This helps avoid wasted driver time and effort including queues etc.

4.4.3 Useful analogy

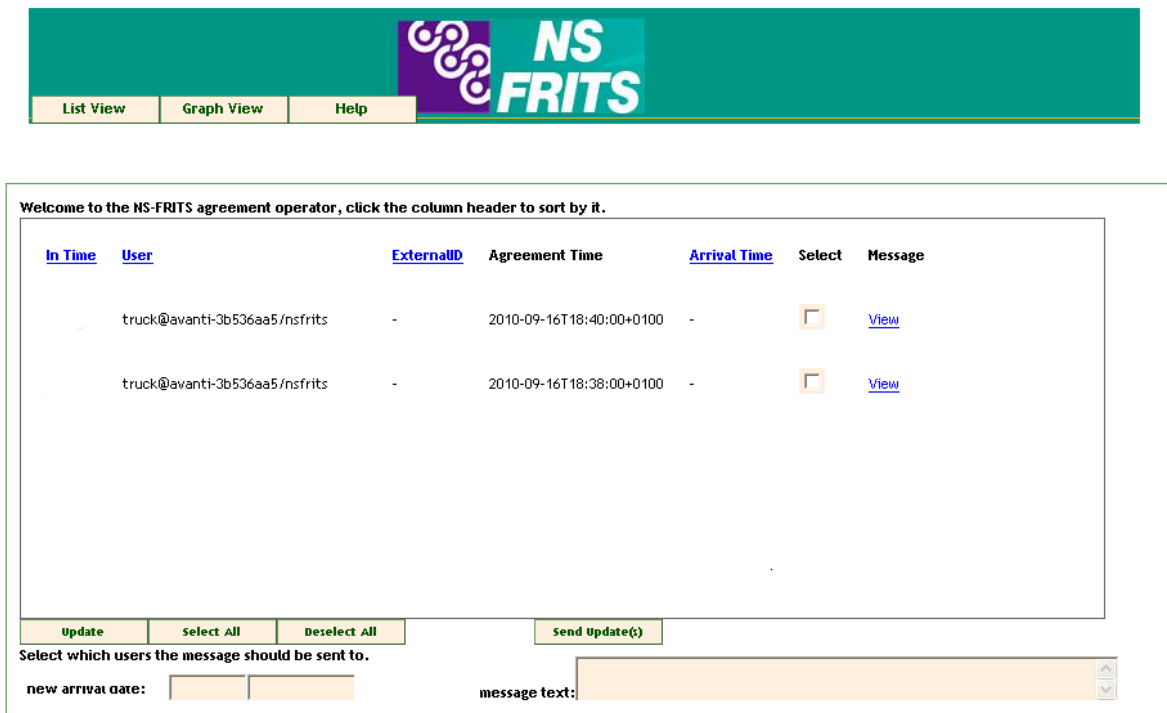
Live bus/train timetable or live flight information at an airport.

4.4.4 Who will find this functionality useful?

The concept was developed to solve a recurrent problem at GCD Glomb, a container terminal operator based in Bremerhaven, Germany. Glomb's main services are handling, clearance and shipping of cargo, including transport to destination. Glomb owns a large fleet of trucks that it uses but also interacts with other freight operators who prefer to use their own trucks.

4.4.4.1 Problems that the NS FRITS Agreements concept can help with;

1. Glomb does not know exactly when a truck driver will turn up to pick their cargo. Sometimes the driver turns up and the cargo has not yet arrived.
2. Sometimes a lot of drivers turn up at the same time resulting in queues at the in-gate terminal and service degradation due to inadequate resourcing.
3. Drivers waste valuable time waiting around that would have been put to more productive use if they'd had prior notification of changing circumstances.



The screenshot shows the NS FRITS interface. At the top is a header bar with the NS FRITS logo and three buttons: 'List View', 'Graph View', and 'Help'. Below the header is a main content area with a table of truck arrival data. The table has columns: 'In Time', 'User', 'ExternalID', 'Agreement Time', 'Arrival Time', 'Select', and 'Message'. There are two rows of data, both for 'truck@avanti-3b536aa5/nsfrits'. Below the table is a control panel with buttons for 'Update', 'Select All', 'Deselect All', and 'Send Update(s)'. Below the buttons is a text input field for 'new arrival date:' and a text area for 'message text:'.

| In Time | User | ExternalID | Agreement Time | Arrival Time | Select | Message |
|---------|-------------------------------|------------|--------------------------|--------------|--------------------------|----------------------|
| | truck@avanti-3b536aa5/nsfrits | - | 2010-09-16T18:40:00+0100 | - | <input type="checkbox"/> | View |
| | truck@avanti-3b536aa5/nsfrits | - | 2010-09-16T18:38:00+0100 | - | <input type="checkbox"/> | View |

Update Select All Deselect All Send Update(s)

Select which users the message should be sent to.

new arrival date: message text:

Figure 5: NS FRITS example interface as used by an operator to view truck arrival times.

4.4.5 Additional comments

It is perhaps safe to assume that many other freight operators face similar problems to those faced by Glomb as described above. The Agreements functionality can be used in isolation or in conjunction with all the other functionalities of the NS FRITS system.

A two day intensive trial of this functionality with some trucks and a terminal operator could demonstrate the practical benefits and usefulness of this capability.

5 APPENDICES

Appendix 1: Photographs from internal trials around London, UK



Appendix 2: NS FRITS system specification – Agreement service

Application description

The agreement provider application is an application that allows data providers to interact with NS FRITS users through the NS FRITS agreement protocol. An NS FRITS Agreement is non-legally binding contract between a truck driver or a transport manager and an agreement provider. Agreements are generally set up by NS FRITS Users to inform their time of arrival and allow agreement providers to send updates to them.

The purpose of an agreement is to enable direct communication between data providers and users. After an agreement has been setup, data providers can send updates containing for example delays instantly to the users. Users also have the possibility to send messages to the data provider if they for example will arrive later.

This application allows agreement providers to view the time of incoming trucks, send updates to the drivers and receive notifications of truck delays.

Application features

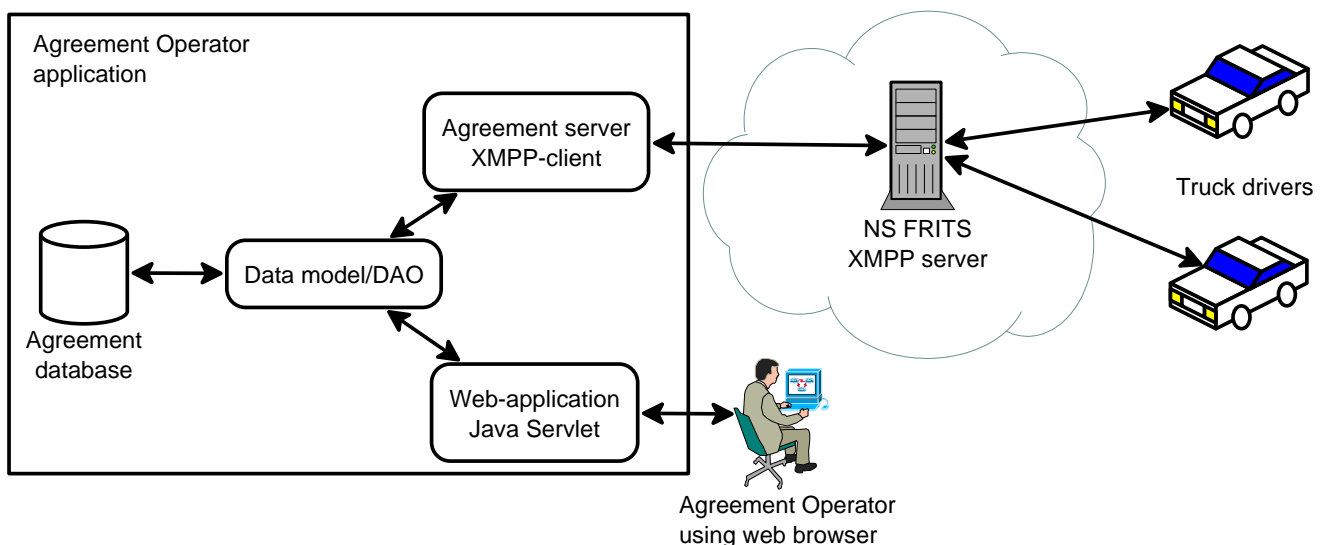
- Ability to accept incoming agreements from drivers.
- Ability to sent update messages to one/several/all agreements.
- Ability to register and display updated arrival times from drivers.
- Ability to display incoming trucks and their arrival time in a list.
- Ability to view messages that drivers has sent.
- Ability to visualize the amount of incoming drivers in the near future with a graph.

Justification

The NS FRITS agreement application is useful for any data provider that also wants to publish an agreement service. For example a ferry terminal could see exactly what time all trucks are arriving, send messages to individual trucks or inform them of delays. This application should be seen as an example of a generic implementation of an agreement provider. In the future, providers can modify this application to suit their specific needs better.

Application design

The Agreement Operator application consist of four different modules as illustrated in the figure below. The Agreement server accepts agreements from truck drivers, sends out updates to existing agreements and receives agreement updates from truck clients. It uses the Data model/DAO module to be able to query the database, which stores all agreements. The web application also uses the Data model/DAO to be able to fetch current agreements, updates and messages and display them to an agreement operator.



The technical level design of the agreement operator is similar to the [NS FRITS Core server](#), [NS FRITS Data provider](#) and the [NS FRITS Data model](#) applications. Read those documents for more information.

Operating environment

The web-application have been tested with Apache Tomcat 6.0.20. The server-application is runnable on all platforms supporting the Java Virtual Machine and JRE 1.6.

Dependencies

Agreement server

The following Libraries are needed to run the NSFRITS-ExternalOperator project:

- PostgreSQL JDBC (postgresql-8.4-701.jdbc4.jar)
A JDBC driver for the PostgreSQL database that is being used.
- c3p0 JDBC DataSources/Resource Pool (c3p0-0.9.1.2.jar)
Library used for database connection pooling.
- Smack API 3.1.0 (smack.jar and smackx.jar)
Providing all XMPP communication

The following projects are needed to run the NSFRITS-ExternalOperator project:

- NS FRITS-ExternalOperator-Datamodel

Web application

The following Libraries are needed to run the NSFRITS-ExternalOperator project:

- PostgreSQL JDBC (postgresql-8.4-701.jdbc4.jar)
A JDBC driver for the PostgreSQL database that is being used.
- c3p0 JDBC DataSources/Resource Pool (c3p0-0.9.1.2.jar)
Library used for database connection pooling.

The following projects are needed to run the NSFRITS-ExternalOperator project:

- NS FRITS-ExternalOperator-Datamodel

Interfaces and classes including important functions

Classes

<A UML diagram that summaries the classes that constitute the application and how they relate to each other>

Communication interfaces

The communications protocols are described in [NS FRITS system specification - API](#).

User interface







The user interface of the agreement operator application will be described in a series of screenshots, explaining the functionality available.

NS-FRITS - Agreement Operator

NS FRITS

List View Graph View Help

Welcome to the NS-FRITS agreement operator, click the column header to sort by it.

| In Time | User | ExternalID | Agreement Time | Arrival Time | Select | Message |
|---|---------------------------------------|------------|--------------------------|--------------------------|--------------------------|----------------------|
|  | test7@segotxl493.vtd.volvo.se/nsfrits | EXT03577 | 2010-08-20T19:00:00+0200 | 2010-08-20T19:42:33+0200 | <input type="checkbox"/> | View |
|  | test4@segotxl493.vtd.volvo.se/nsfrits | EXT85941 | 2010-08-20T18:00:00+0200 | 2010-08-20T18:51:33+0200 | <input type="checkbox"/> | View |
|  | test9@segotxl493.vtd.volvo.se/nsfrits | EXT97433 | 2010-08-20T19:00:00+0200 | 2010-08-20T18:42:52+0200 | <input type="checkbox"/> | View |
|  | test1@segotxl493.vtd.volvo.se/nsfrits | EXT15139 | 2010-08-20T18:00:00+0200 | 2010-08-20T18:15:53+0200 | <input type="checkbox"/> | View |
|  | test6@segotxl493.vtd.volvo.se/nsfrits | EXT32039 | 2010-08-20T18:00:00+0200 | 2010-08-20T17:56:53+0200 | <input type="checkbox"/> | View |
|  | test5@segotxl493.vtd.volvo.se/nsfrits | EXT81161 | 2010-08-20T18:00:00+0200 | 2010-08-20T17:55:52+0200 | <input type="checkbox"/> | View |

Update Select All Deselect All Send Update(s)

Select which users the message should be sent to.

new arrival date: 2010-08-20 14:49:53+0200 message text:

When the agreement operator launches the application, all trucks that successfully has set up an agreement will be displayed in the list.

The clock icon to the left symbolizes if the user will be in time or not. A red clock means that the user will be late, while a blue clock means that the user will be in time.

The external id field is an optional parameter that the truck can send to connect the agreement to a real contract, like for example a booking id. The idea behind this is to enable the development of smarter application in the future, like for example view all drivers that are late for a particular ferry.

The agreement time is the time the driver sets when he/she sets up the agreement.

The arrival time contains a newer updated time sent by the driver. If the driver detects that he/she will be late or early, this value is set.

The operator can also sort the values according to vehicles In time, User, ExternalID or Arrival time by pressing the column header. When sorting according to Arrival time, the program chooses the value in the arrival time column if available, otherwise the agreement time.

NS-FRITS - Agreement Operator

List View Graph View Help

Welcome to the NS-FRITS agreement operator, click the column header to sort by it.

| In Time | User | ExternalID | Agreement Time | Arrival Time | Select | Message |
|---------|---------------------------------------|------------|--------------------------|--------------------------|-------------------------------------|----------------------|
| | test7@segotxl493.vtd.volvo.se/nsfrits | EXT03577 | 2010-08-20T19:00:00+0200 | 2010-08-20T19:42:33+0200 | <input type="checkbox"/> | View |
| | test4@segotxl493.vtd.volvo.se/nsfrits | EXT85941 | 2010-08-20T18:00:00+0200 | 2010-08-20T18:51:33+0200 | <input checked="" type="checkbox"/> | View |
| | test9@segotxl493.vtd.volvo.se/nsfrits | EXT97433 | 2010-08-20T19:00:00+0200 | 2010-08-20T18:42:52+0200 | <input type="checkbox"/> | View |
| | test1@segotxl493.vtd.volvo.se/nsfrits | EXT15139 | 2010-08-20T18:00:00+0200 | 2010-08-20T18:15:53+0200 | <input checked="" type="checkbox"/> | View |
| | test6@segotxl493.vtd.volvo.se/nsfrits | EXT32039 | 2010-08-20T18:00:00+0200 | 2010-08-20T17:56:53+0200 | <input checked="" type="checkbox"/> | View |
| | test5@segotxl493.vtd.volvo.se/nsfrits | EXT81161 | 2010-08-20T18:00:00+0200 | 2010-08-20T17:55:52+0200 | <input checked="" type="checkbox"/> | View |

Update Select All Deselect All Send Update(s)

Select which users the message should be sent to.

new arrival date: 2010-08-20 18:30:53+0200 message text:

The select checkbox is used to select what agreement(s) the operator wants to send a new message to. After selecting, the operator enters a text in the textbox, a new arrival date and then presses the 'Send Update(s)' button.

NS-FRITS - Agreement Operator

List View Graph View Help

Welcome to the NS-FRITS agreement operator, click the column header to sort by it.

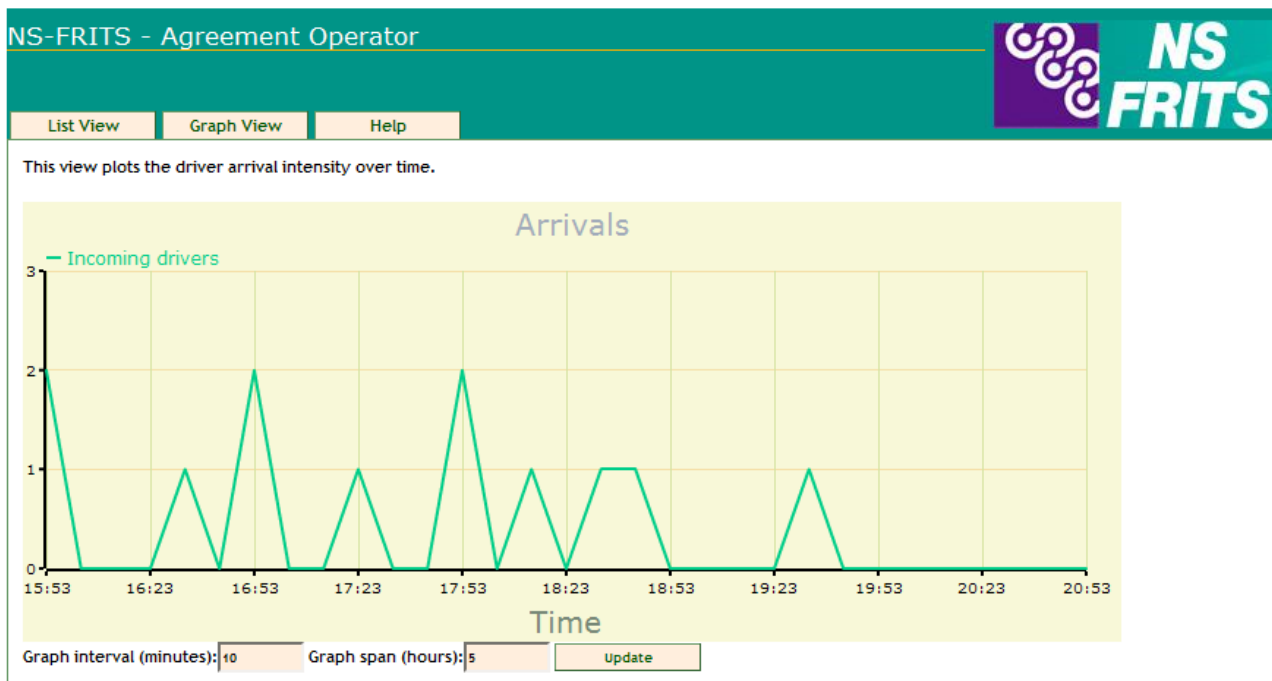
| In Time | User | ExternalID | Agreement Time | Arrival Time | Select | Message |
|---------|---------------------------------------|------------|--------------------------|--------------------------|-------------------------------------|----------------------|
| | test7@segotxl493.vtd.volvo.se/nsfrits | | | 19:42:33+0200 | <input type="checkbox"/> | View |
| | test4@segotxl493.vtd.volvo.se/nsfrits | | | 18:51:33+0200 | <input checked="" type="checkbox"/> | View |
| | test9@segotxl493.vtd.volvo.se/nsfrits | | | 18:42:52+0200 | <input type="checkbox"/> | View |
| | test1@segotxl493.vtd.volvo.se/nsfrits | EXT15139 | 2010-08-20T18:00:00+0200 | 2010-08-20T18:15:53+0200 | <input checked="" type="checkbox"/> | View |
| | test6@segotxl493.vtd.volvo.se/nsfrits | EXT32039 | 2010-08-20T18:00:00+0200 | 2010-08-20T17:56:53+0200 | <input checked="" type="checkbox"/> | View |
| | test5@segotxl493.vtd.volvo.se/nsfrits | EXT81161 | 2010-08-20T18:00:00+0200 | 2010-08-20T17:55:52+0200 | <input checked="" type="checkbox"/> | View |

Update Select All Deselect All Send Update(s)

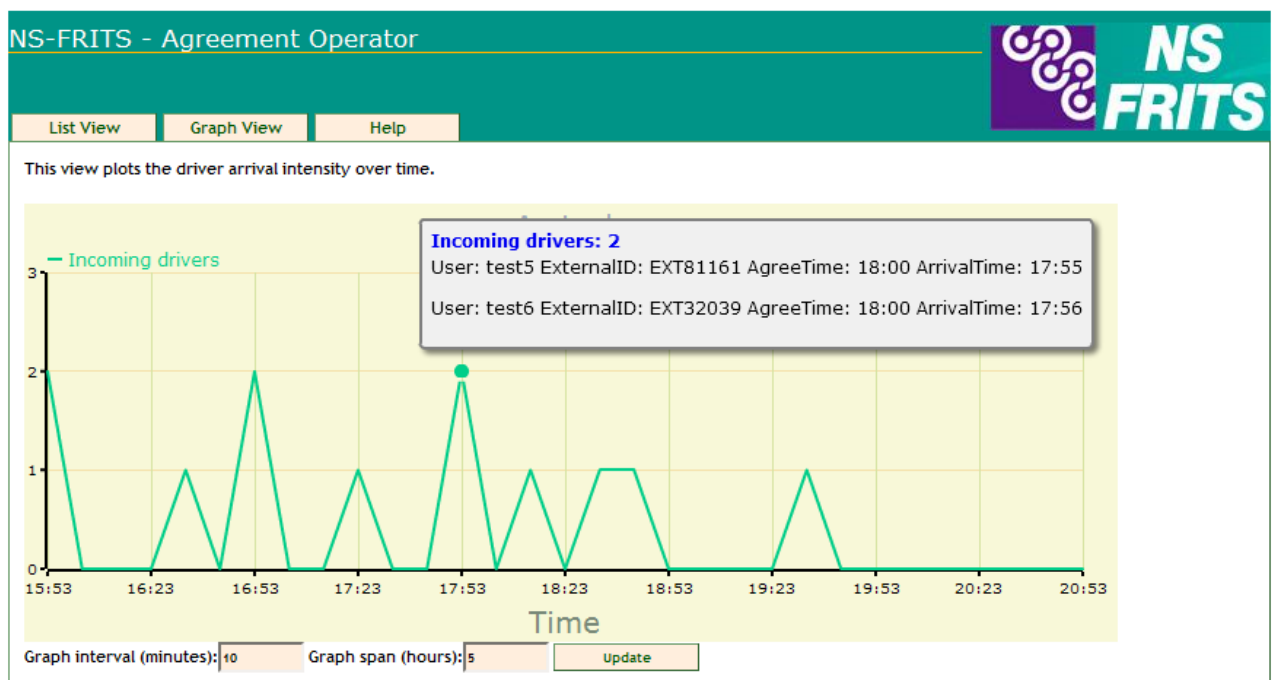
Select which users the message should be sent to.

new arrival date: 2010-08-20 18:30:53+0200 message text:

By clicking the link in the message column, the operator can see the last message the driver sent.



By pressing the Graph View, the application will visualize the intensity of driver by drawing a graph with incoming drivers on the y axis and time on the x axis. It is possible to change the display settings of the graph by entering new values in input boxes below.



By hovering the mouse over a certain point in the graph, the operator can more verbose information about the drivers arriving at that particular time.

Fault check, diagnostics and FAQ

It is possible to try out the Agreement operator with a test client distributed with the NSFRITS-ExternalOperator project. This client is located in the test package, **eu.nsfrits.externaloperator.testclient** and is started by running the **Sender.java** class. This client simulates several trucks, each setting up.

Appendix 3: NS FRITS system specification – Android client

Application description

This application aims to help the trucker to decide which route is the best to take mainly providing him with security information. It also enables the trucker to receive and send updates about the arrival time to a place where he has an agreement with an external operator.

Application features

- Log-in to the NSFRITS server.
- Choose categories.
- Choose languages.
- Receive a new order from a transport manager.
- Send a list of checked things on the vehicle.
- Choose, find and edit a route and retrieve the information along.
- Search for information around a certain location.
- Subscribe to a service to receive automatic information updates.
- Send an alert to the NSFRITS server and to the dispatcher.
- Settle an agreement with an external operator and receive updates about the arrival time.
- While driving :
 - send updates about his position,
 - compute the time remaining to reach a location in order to update the arrival time,
 - find the new information in the vicinity.

Justification

This application is useful because it implements all the functionality that the NSFRITS API provides. It allows the trucker to be helped for preparing his journey by giving him information along the route or when he is driving by receiving updates in the vicinity. It also enables a better synchronisation between the truckers and the external operator via the agreement system which keeps both part updated about the arrival time and via a messaging mechanism.

This application is a good example of the way that existing applications will have to integrate the NSFRITS API in their system. But it's also a proper application which can be fully used in real life by independent truckers who wish to use the NSFRITS services to help them before and during their journey.

Application design

The application design tries to follow the MVC architecture.

Model

The model is mainly based on a class collection fed by the information retrieved from the NSFRITS server. Global data is also kept in the model to configure the application and ensure the consistence.

View

The main view of the application is a map view and a menu from which the user can open several dialogs to configure the application or where he can trigger several kinds of actions (requests to the server).

The view can also be a navigation view (Google service).

Controller

The controller is the link between the view and the model. When the user triggers an action from a dialog window, a specific controller is called to retrieve information from the NSFRITS server (POIs along a route, agreement updates) or from a Google service (e.g. geocoding, find route).

Then the controller injects the information in the model which can be used to update the view.

Operating environment

The application can be run in any Android environment. The application works with Eclair and Froyo.

The application has been tested on an Emulator via a plug-in bonded to Eclipse and on a Google HTC device. Some amendments will certainly be needed to adapt the software to other platforms, especially the resolutions of the dialog windows.

Dependencies

The following libraries are needed to run the application.

- XMPP library for Android (asmack-2010.05.07.jar)
Used for communication with the NSFRITS Server via the XMPP protocol.

The following libraries come with the android SDK:

- Android library to be able to develop Android applications(android.jar)
- Google library to use Google Maps in Android applications (maps.jar)

Implementation constraints

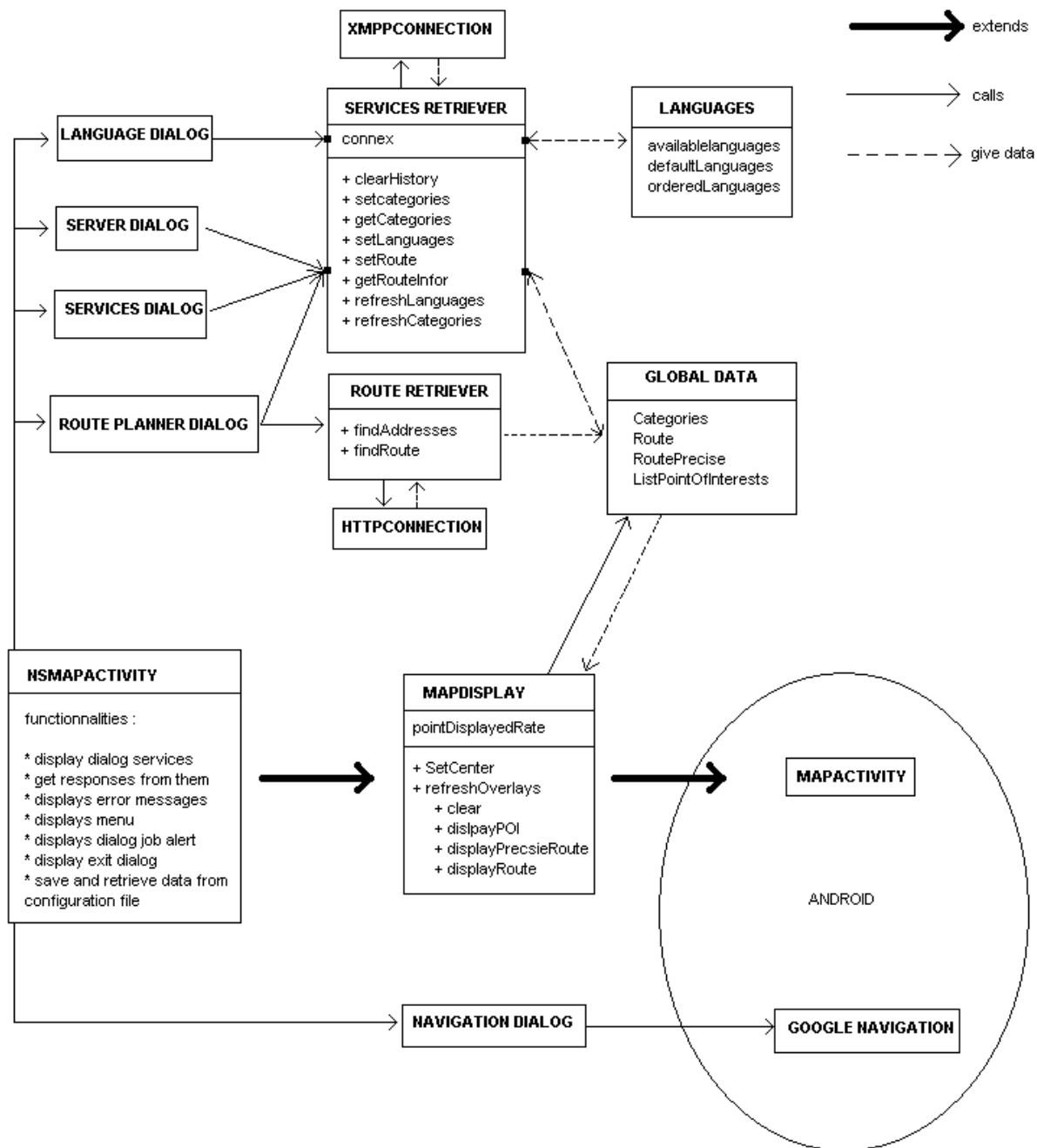
The development of an Android application has been accomplished via a plug-in bonded to Eclipse. An emulator of the Android device is used to test the application. Several problems exist with the utilisation of the emulator:

- The launch of the emulator is very long
- The network connection is sometimes non-existent.
- The utilisation of certain applications are very slow (e.g. displacements in Google maps or XMPP communication)
- Certain functionalities are not available on the emulator (e.g. the speech to text can't be used on the phone)
- Certain functionalities can't be used on the phone (e.g. mock coordinates via the DDMS)
- Certain functionalities don't work the same way on both platforms (e.g. multi-touch on the phone screen does not react in the same way than on the emulator)

Lots of developed functionalities in the application are based on Google services. The application is then dependent of a third party. It needs to register for a Google map key for example and it cannot work if some services are not available.

Interfaces and classes including important functions

Classes



External interfaces

The Android client application does not have any external interfaces.

Communication interfaces

The Android client application uses the Extensible Messaging and Presence Protocol (XMPP) to communicate with the NSFRITS server.

The communication interface is described in [NS FRITS system specification - API](#).

Software interfaces and interrelation with other applications

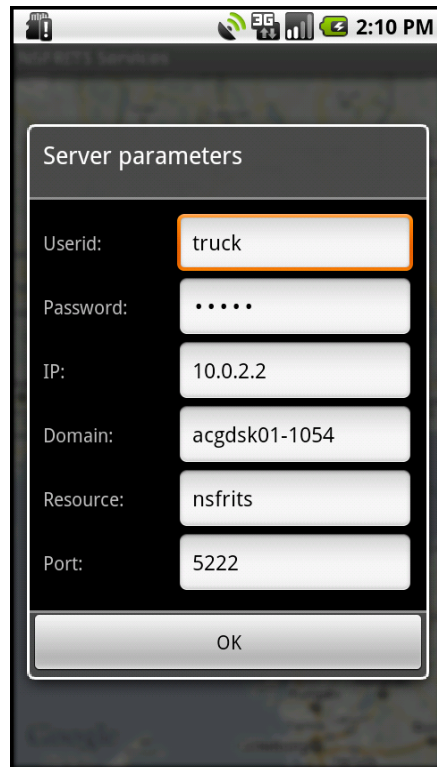
The main interaction with another application is the communication with the NSFRITS server via XMMP.

The android application can also communicate with a transport manager. It can receive a job from it and if there is an accident or any other problem, it can send an alert.

The android application can also set up agreements with external operators. Those agreements enable the user and the external operator to update each other about the arrival time. They can also exchange messages for a better comprehension of the situation.

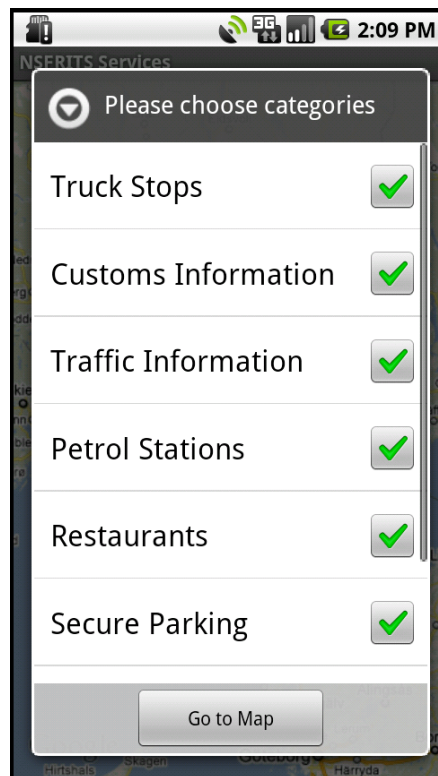
User interface

This section will be summarized in a series of screenshots, each describing use cases of the application.



The screenshot shows a mobile application interface on an Android device. At the top, the status bar displays icons for signal strength, Wi-Fi, and battery, along with the time 2:10 PM. The main screen shows a dialog box titled "Server parameters". Inside the dialog, there are several labeled input fields: "Userid:" containing the text "truck", "Password:" containing five dots, "IP:" containing "10.0.2.2", "Domain:" containing "acgdsk01-1054", "Resource:" containing "nsfrits", and "Port:" containing "5222". At the bottom of the dialog is a button labeled "OK".

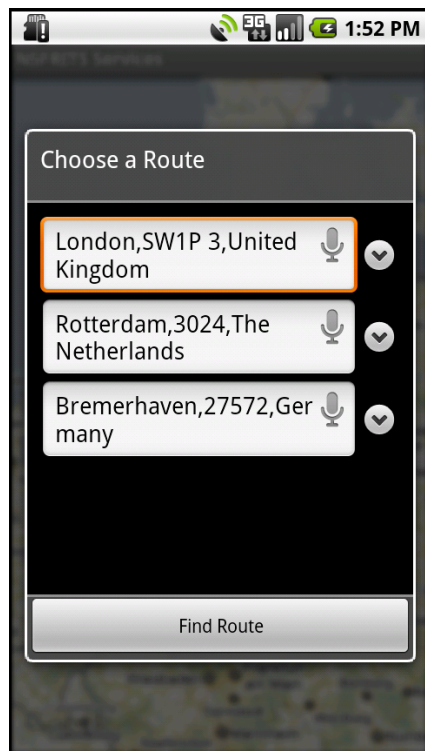
The user has to connect to the NS FRITS server by providing his login credentials and the XMPP server details.



The user chooses the categories of service he wants to see display on the map.

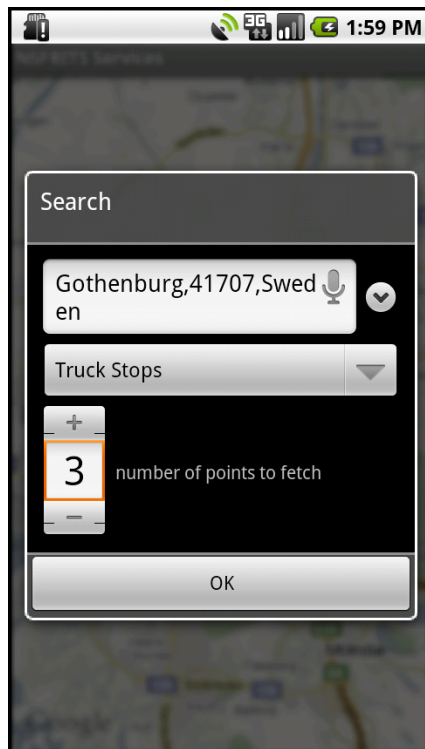


The user can choose the languages preference order in which he would like to see the information displayed.

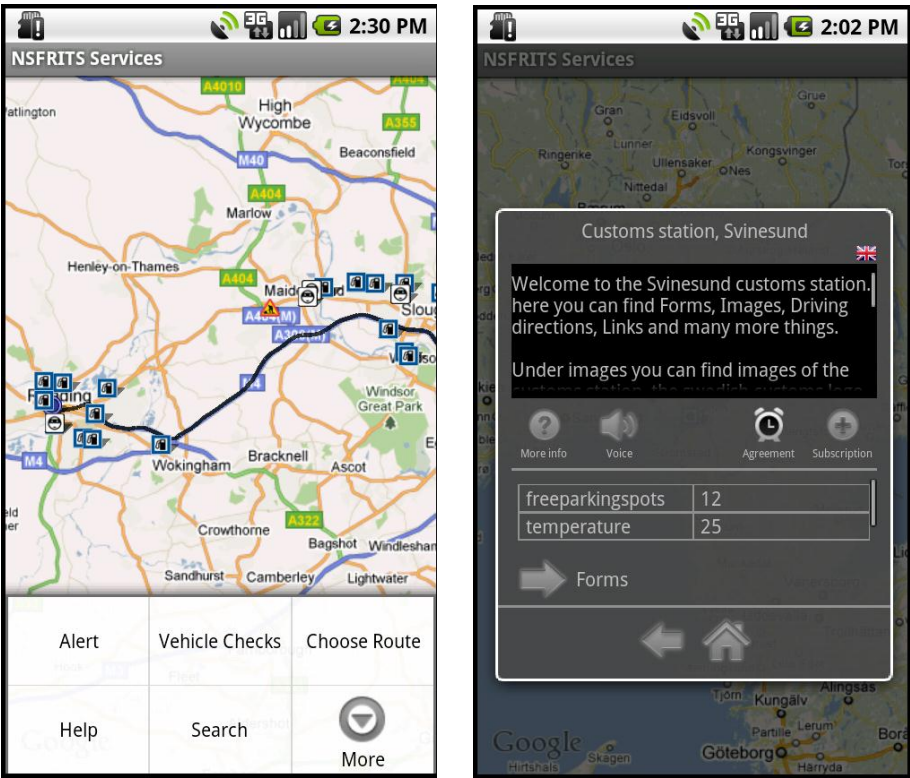


The user can choose, find and edit a route. He can find the destination via several possibilities:

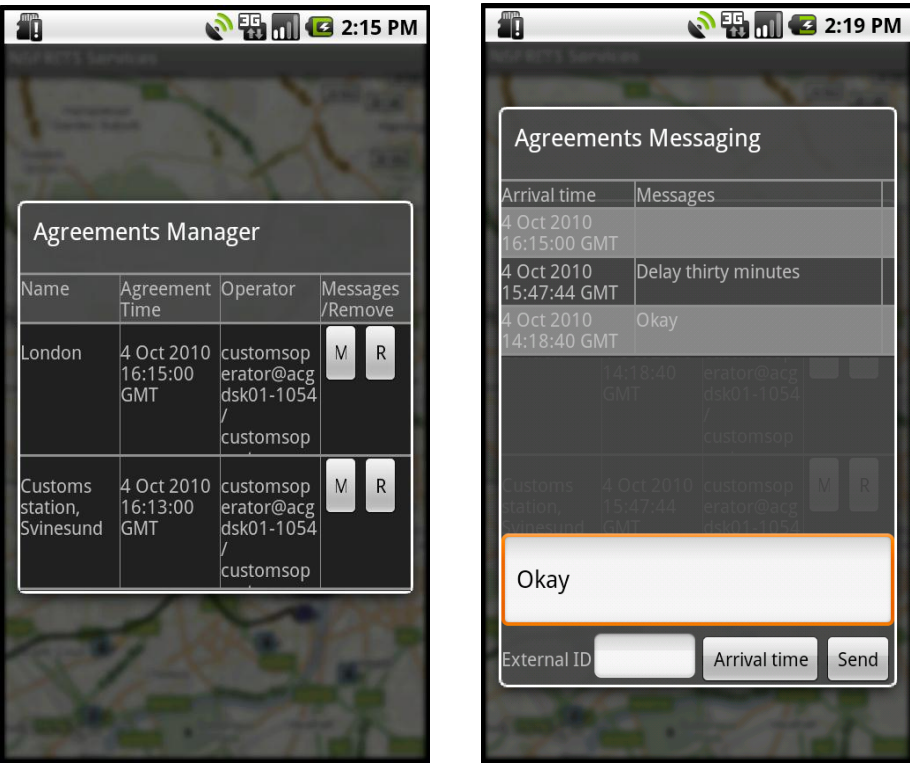
- Write down the name of the destination.
- Use the text to speech functionality to guess the destination from the user voice.
- Choose the location on the map.
- Use the current location of the vehicle via the onboard GPS.



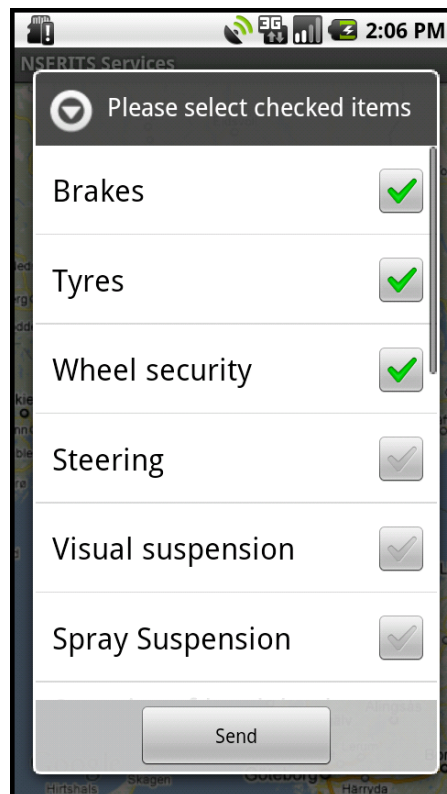
The user can search information around a certain location and for a special kind of categories.



The user can see the points of interest along the route and can click on those points to see more information about them. Those points of interest are retrieved when the user choose a route but can also be retrieved automatically when the user drives.



The user can manage the agreements he has performed with external operators and manually send updates about his arrival time and/or send a message.

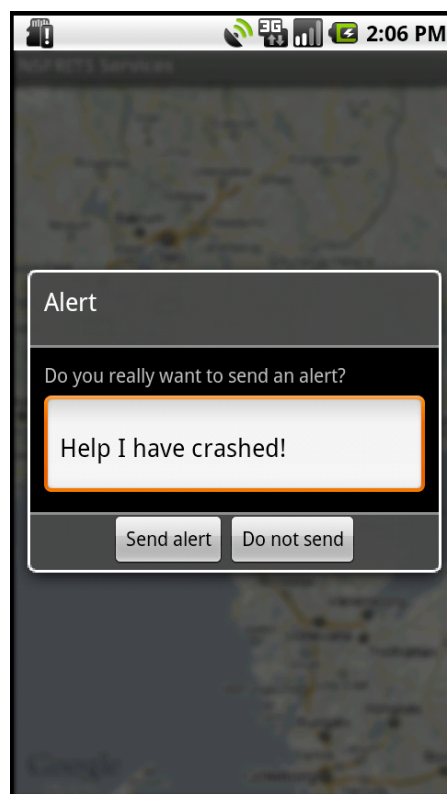


The screenshot shows a mobile application interface titled "NSERITS Services". A modal dialog box is displayed with the heading "Please select checked items". It contains a list of vehicle components with corresponding checkboxes:

| Item | Status |
|-------------------|-------------------------------------|
| Brakes | <input checked="" type="checkbox"/> |
| Tyres | <input checked="" type="checkbox"/> |
| Wheel security | <input checked="" type="checkbox"/> |
| Steering | <input type="checkbox"/> |
| Visual suspension | <input type="checkbox"/> |
| Spray Suspension | <input type="checkbox"/> |

At the bottom of the dialog is a "Send" button. The background of the app shows a map.

The user has to check his vehicle before driving by filling a form that he has to send to the transport manager.



The screenshot shows the same mobile application interface. An "Alert" dialog box is displayed over a map background. The dialog contains the following elements:

- Title: Alert
- Question: Do you really want to send an alert?
- Text input field containing: Help I have crashed!
- Buttons: "Send alert" and "Do not send"

An alert button enables the user to send his location to warn the other users about his situation. It can also be configured to send an alert to a transport manager.

Fault check, diagnostics and FAQ

If the Google map is not displayed, it means that either you don't have an internet connection or your Google map key is not correct (you need to have a unique key for your application and for your emulator).

If you don't manage to connect to the NSFRITS server, you should ensure that the server you are using is running and that your credentials are correct for this server.

If any problem happens during the execution of the application, you can use the DDMS view provided in

Appendix 4: NS FRITS system specification – API

IQ Methods

Currently implemented methods using the <iq/> stanza.

Method: getInfoForPoint

Argument: *point*

Argument type: WKT Point

Expected response: list of *infoobject*

Description: Returns all InfoObjects near the point provided in the argument. Only InfoObjects inside the users previously set categories will be returned. The result will be displayed in the users highest priority language.

Example:

```
<!-- Request -->
<iq id='qxmpp10' to='nsfrits@VTECW389.vcn.ds.volvo.net/nsfrits' type='get'>
  <query xmlns='nsfrits:iq'>
    <methodname>getInfoForPoint</methodname>
    <arguments>
      <argument>
        <key>point</key>
        <value>POINT(11.878967 57.737883)</value>
      </argument>
    </arguments>
  </query>
</iq>

<!-- Response -->
<iq from='nsfrits@vtecw389.vcn.ds.volvo.net/nsfrits' to='truck1@vtecw389.vcn.ds.volvo.net/QXmpp'
id='qxmpp10' type='result'>
  <query xmlns='nsfrits:iq'>
    <methodresponse>
      <infoobject>
        <id>20</id>
        <validity>
          <from>2010-03-30 10:09:39</from>
          <to>2010-04-10 23:56:19</to>
        </validity>
        <sequenceid>1</sequenceid>
        <location>
          <area>POLYGON((11.546630859516 57.554946161479,11.546630859516
57.825065399809,11.975097656374 57.825065399809,11.975097656374
57.554946161479,11.546630859516 57.554946161479))</area>
          <point>POINT(11.644134521626 57.701266095542)</point>
        </location>
        <datanode>
          <id>23</id>
          <description>
            <title>Öckerö parking</title>
            <text>Parking space for the Öckerö ferry.</text>
            <language>en</language>
          </description>
          <attachment>
            <mime-type>image/png</mime-type>
            <size>389878</size>
            <url>http%3A%2F%2F131.97.51.233%3A8084%2FServiceAdmin%2Fuploads%2F23%2Fcar.PNG</url>
          </attachment>
        </datanode>
```

```

</infoobject>
</methodresponse>
</query>
</iq>

```

Method: getInfoForRoute

Argument: -

Expected response: list of *infoobject*

Description: Returns all InfoObjects along the users previously set route. Only InfoObjects inside the users previously set categories will be returned. The result will be displayed in the users highest priority language.

Example:

```

<!-- Request -->
<iq id='qxmpp31' to='nsfrits@VTECW389.vcn.ds.volvo.net/nsfrits' type='get'>
  <query xmlns="nsfrits:iq">
    <methodname>getInfoForRoute</methodname>
  </query>
</iq>

```

```

<!-- See getInfoForPoint for example response -->

```

Method: getCategories

Argument: -

Expected response: list of *category*

Description: Returns all available categories and a description to each of them.

Example:

```

<!-- Request -->
<iq id='qxmpp34' to='nsfrits@VTECW389.vcn.ds.volvo.net/nsfrits' type='get'>
  <query xmlns="nsfrits:iq">
    <methodname>getCategories</methodname>
  </query>
</iq>

```

```

<!-- Response -->
<iq from='nsfrits@vtecw389.vcn.ds.volvo.net/nsfrits' to='truck1@vtecw389.vcn.ds.volvo.net/QXmpp'
id='qxmpp34' type='result'>
  <query xmlns='nsfrits:iq'>
    <methodresponse>
      <category>
        <id>1</id>
        <description>
          <title>customs</title>
          <text>some text</text>
          <language>en</language>
        </description>
      </category>
      <category>
        <id>2</id>
        <description>
          <title>road service</title>
          <text>Road services</text>
          <language>en</language>
        </description>
      </category>
      <category>
        <id>4</id>
        <description>
          <title>Foreign Law</title>

```

<text>Foreign laws deal with the fact that different countries, and other regions like federal states or individual cities, may have different rules and regulations concerning truck transports. </text>

<language>en</language>
 </description>
 </category>
 </methodresponse>
 </query>
 </iq>

Method: setCategories

Argument: categories

Argument type: comma separated list of category id:s

Expected response: -

Description: Sets the categories a user is interested in.

Example:

```
<!-- Request -->
<iq id='qxmpp37' to='nsfrits@VTECW389.vcn.ds.volvo.net/nsfrits' type='set'>
  <query xmlns="nsfrits:iq">
    <methodname>setCategories</methodname>
    <arguments>
      <argument>
        <key>categories</key>
        <value>42,123,9</value>
      </argument>
    </arguments>
  </query>
</iq>
```

```
<!-- Response -->
<iq from='nsfrits@vtecw389.vcn.ds.volvo.net/nsfrits' to='truck1@vtecw389.vcn.ds.volvo.net/QXmpp'
id='qxmpp37' type='result'>
  <query xmlns='nsfrits:iq'/>
</iq>
```

Method: setLanguages

Argument: languages

Argument type: comma separated list of languages according to ISO 639-1

Expected response: -

Description: Sets the languages a user is interested in in a priority list, highest priority first.

Example:

```
<!-- Request -->
<iq id='qxmpp38' to='nsfrits@VTECW389.vcn.ds.volvo.net/nsfrits' type='set'>
  <query xmlns="nsfrits:iq">
    <methodname>setLanguages</methodname>
    <arguments>
      <argument>
        <key>languages</key>
        <value>sv,en</value>
      </argument>
    </arguments>
  </query>
</iq>
```

```
<!-- Response -->
<iq from='nsfrits@vtecw389.vcn.ds.volvo.net/nsfrits' to='truck1@vtecw389.vcn.ds.volvo.net/QXmpp'
id='qxmpp38' type='result'>
```

```
<query xmlns='nsfrits:iq'/>
</iq>
```

Method: setRoute

Argument: route

Argument type: WKT Linestring

Expected response: -

Description: Sets a users route to the server.

Example:

```
<!-- Request -->
<iq id='qxmpp41' to='nsfrits@VTECW389.vcn.ds.volvo.net/nsfrits' type='set'>
  <query xmlns='nsfrits:iq'>
    <methodname>setRoute</methodname>
    <arguments>
      <argument>
        <key>route</key>
        <value>LINESTRING(11.948318 57.714785, 11.961365 57.724319, 11.981277 57.726519,
11.992264 57.715518, 11.995010 57.696809, 12.039642 57.617095, 12.065048 57.585455, 12.051315
57.508447, 12.046509 57.477450, 12.075348 57.466374, 12.019043 57.410941)
        </value>
      </argument>
    </arguments>
  </query>
</iq>

<!-- Response -->
<iq from='nsfrits@vtecw389.vcn.ds.volvo.net/nsfrits' to='truck1@vtecw389.vcn.ds.volvo.net/QXmpp'
id='qxmpp41' type='result'>
  <query xmlns='nsfrits:iq'/>
</iq>
```

Method: clearHistory

Argument: -

Expected response: -

Description: Removes all cached InfoObjects for that particular user from the server.

Example:

```
<!-- Request -->
<iq id='qxmpp44' to='nsfrits@VTECW389.vcn.ds.volvo.net/nsfrits' type='set'>
  <query xmlns='nsfrits:iq'>
    <methodname>clearHistory</methodname>
  </query>
</iq>

<!-- Response -->
<iq from='nsfrits@vtecw389.vcn.ds.volvo.net/nsfrits' to='truck1@vtecw389.vcn.ds.volvo.net/QXmpp'
id='qxmpp44' type='result'>
  <query xmlns='nsfrits:iq'/>
</iq>
```

Method: getAllIOTypes

Argument: -

Expected response: -

Description: Returns all types known by the server with a description and an icon.

Example:

```
<!-- Request -->
<iq id='qxmpp44' to='nsfrits@VTECW389.vcn.ds.volvo.net/nsfrits' type='set'>
```



```

<query xmlns="nsfrits:iq">
  <methodName>getAllIOTypes</methodName>
</query>
</iq>

```

```

<!-- Response -->
<iq from='nsfrits@segotxl493.vtd.volvo.se/nsfrits' to='test@segotxl493.vtd.volvo.se/nsfrits' id='GfaGH-6'
type='result'><query xmlns='nsfrits:iq'><methodresponse>
<iotype>
  <id>1</id>
  <icon>traffic-x6-1</icon>
  <description>
    <id>3</id>
    <title>Accidents</title>
    <text> </text>
    <language>en</language>
    <sound> </sound>
  </description>
</iotype>
<iotype>
  <icon>misc-station</icon>
  <description>
    <id>2</id>
    <title>Station</title>
    <text> </text>
    <language>en</language>
    <sound> </sound>
  </description>
</iotype>
</methodresponse></query></iq>

```

Method: clearSubscriptions

Argument: -

Expected response: -

Description: Removes all InfoObject subscriptions for a user.

Example:

```

<!-- Request -->
<iq id="M0Kka-3" to="service.VTECW405/nsfrits" type="get">
<query xmlns="nsfrits:iq">
  <methodName>clearSubscriptions</methodName>
  <arguments></arguments>
</query>
</iq>

```

```

<!-- Response -->
<iq type="result" id="M0Kka-3" from="nsfrits@service.vtecw405" to="test2@vtecw405/nsfrits">
<query xmlns="nsfrits:iq"/>
</iq>

```

Method: addSubscription

Argument: id

Argument type: InfoObject id

Expected response: -

Description: Adds an InfoObject subscription to a user.

Example:

```

<!-- Request -->
<iq id="M0Kka-6" to="service.VTECW405/nsfrits" type="get">

```

```
<query xmlns="nsfrits:iq">
  <methodName>addSubscription</methodName>
  <arguments>
    <argument>
      <key>id</key>
      <value>59779</value>
    </argument>
  </arguments>
</query>
</iq>
```

```
<!-- Response -->
<iq type="result" id="M0Kka-6" from="nsfrits@service.vtecw405" to="test2@vtecw405/nsfrits">
  <query xmlns="nsfrits:iq"/>
</iq>
```

Method: deleteSubscription

Argument: id

Argument type: InfoObject id

Expected response: -

Description: Deletes a specific InfoObject subscription.

Example:

```
<!-- Request -->
<iq id="M0Kka-6" to="service.VTECW405/nsfrits" type="get">
  <query xmlns="nsfrits:iq">
    <methodName>deleteSubscription</methodName>
    <arguments>
      <argument>
        <key>id</key>
        <value>59779</value>
      </argument>
    </arguments>
  </query>
</iq>
```

```
<!-- Response -->
<iq type="result" id="M0Kka-6" from="nsfrits@service.vtecw405" to="test2@vtecw405/nsfrits">
  <query xmlns="nsfrits:iq"/>
</iq>
```

Method: searchPOI

Argument: point, category, results

Argument type: WKT point, category id, int

Expected response: -

Description: returns the (results) closest InfoObjects of the specified category near the specified point.

Example:

```
<!-- Request -->
<iq id="M0Kka-8" to="service.VTECW405/nsfrits" type="get">
  <query xmlns="nsfrits:iq">
    <methodName>searchPOI</methodName>
    <arguments>
      <argument>
        <key>point</key>
        <value>POINT(15 59.1)</value>
      </argument>
      <argument>
        <key>category</key>
```

```

    <value>5</value>
  </argument>
</argument>
  <key>results</key>
  <value>2</value>
</argument>
</arguments>
</query>
</iq>

```

```

<!-- Response -->
<iq type="result" id="19GRI-8" from="nsfrits@service.vtecw405" to="test2@vtecw405/nsfrits">
  <query xmlns="nsfrits:iq">
    <methodresponse xmlns="">
      <infoobject>
        ...
      </infoobject>
      <infoobject>
        ...
      </infoobject>
    </methodresponse>
  </query>
</iq>

```

Asynchronous messages

Client to server

Message extension: position

Argument type: WKT point

Description: Sends an update of the clients position to the server. This allows the server to send asynchronous alerts of nearby events to the client.

Example:

```

<message id="FDhuP-5" to="service.VTECW405/nsfrits">
  <position xmlns='nsfrits:async'>POINT(23.8 51.2)</position>
</message>

```

Message Extension: position and alert

Argument type: WKT point, message

Description: Sends an alert to the server, allowing the server to create an InfoObject and send out alerts to nearby clients.

Example:

```

<message id="FDhuP-6" to="service.VTECW405/nsfrits">
  <alert xmlns='nsfrits:async'>help I have crashed!</alert>
  <position xmlns='nsfrits:async'>POINT(23.7 51.1)</position>
</message>

```

Server to client

Message Extension: update

Description: Sends an asynchronous update to the client, it can either be a subscription update or an alert update as specified by the “type” attribute. Each update contains one or more InfoObjects.

Example:

```

<message to="test2@vtecw405" from="nsfrits@service.vtecw405">

```

```

<updates xmlns="nsfrits:async" type="subscription">
  <infoobject xmlns="">
    <id>59779</id>
    <provider>Aral</provider>
    <category>3</category>
    <iotype>traffic-h3-1</iotype>
    <validity>
      <from>2010-08-05T14:11:40+02:00</from>
      <to>2020-08-02T14:11:40+02:00</to>
    </validity>
    <sequenceid>10</sequenceid>
    <location>
      <area>POLYGON((10.76997      51.87287,10.80997      51.87287,10.80997      51.83287,10.76997
51.83287,10.76997 51.87287))</area>
      <point>POINT (10.78997 51.85287)</point>
    </location>
    <datanode>
      <id>70986</id>
      <description>
        <title>Aral, [38855] Aral Station Autohaus Wernigerode Gmbh - Dornbergsweg 49 - Tel. 03943/21195
[wernigerode]</title>
        <text>Aral, [38855] Aral Station Autohaus Wernigerode Gmbh - Dornbergsweg 49 - Tel. 03943/21195
[wernigerode]</text>
        <language>en</language>
      </description>
    </datanode>
  </infoobject></updates></message>

```

Agreement API

Method: Agreement registration

Argument: agreeetime, externalid

Argument type: timestamp (ISO 8601), string

Expected response: id of the registered agreement

Description: Sets up an agreement service with the operator at a specific time.

Example:

```

<!-- Request -->
<iq id="G47CE-14" to="customsoperator@segotxl493.vtd.volvo.se/customsoperator" type="get">
  <query xmlns="operator:iq:register">
    <agreeetime>2010-08-25T13:00:00+0200</agreeetime>
    <externalid>EXT72688</externalid>
  </query>
</iq>

<!-- Response -->
<iq
  from='customsoperator@segotxl493.vtd.volvo.se/customsoperator'
to='test1@segotxl493.vtd.volvo.se/nsfrits' id='G47CE-14' type='result'>
  <query xmlns='operator:iq:register'>
    <id>57</id>
  </query>
</iq>

```

Method: Agreement unregistration

Argument: agreement id

Argument type: int

Expected response: -

Description: Deletes an agreement with the server.

Example:

```
<!-- Request -->
<iq id="iR5Cz-102" to="customsoperator@segotxl493.vtd.volvo.se/customsoperator" type="get"><query
xmlns="operator:iq:unregister">
  <id>106</id>
</query></iq>
```

```
<!-- Response -->
<iq
                                from='customsoperator@segotxl493.vtd.volvo.se/customsoperator'
to='test1@segotxl493.vtd.volvo.se/nsfrits' id="iR5Cz-102" type='result'>
  <query xmlns='operator:iq:unregister'/>
</iq>
```

Message Extension: update

Description: Sends an agreement update with a new agreement time and optionally a message containing a description of the update. Works both client to server and server to client.

Example:

```
<message id="iR5Cz-92" to="customsoperator@segotxl493.vtd.volvo.se/customsoperator">
  <updates xmlns='operator:message:updates'>
    <arrivaltime>2010-08-25T14:50:28+0200</arrivaltime>
    <id>106</id>
    <text>will be arriving later</text>
  </updates>
</message>
```

Appendix 5: NS FRITS system specification – Core server

Application description

The NS FRITS core server is the application responsible of serving requests from NS FRITS clients. The application connects as a client to an XMPP server, where it responds to a predefined API of requests from NS FRITS clients. The API uses XMPPs built in method-response stanza, IQ. Some of the available API calls include: `getInfoObjectAlongRoute`, `setLanguages`, `setCategories` and `searchForPOI`.

Application features

- Serve NS FRITS clients with data from the NS FRITS database through various API calls.
- Provides a subscription service of InfoObjects, sending updates to clients as soon as object changes.

Justification

The NS FRITS core server is responsible of providing an interface to the developers of NS FRITS clients; it is also responsible of the communications layer of NS FRITS. Since the core server constitutes the backend of the NS FRITS system and clients would simply not work without it, no further justification is needed.

Application design

Overview

The application consists of three layers, the XMPP layer, the API layer and the Data layer. The XMPP layer handles connections with the XMPP server and incoming requests from NS FRITS clients. Requests are then forwarded to the API layer where the correct method and its parameters are identified. Finally the API layer uses the [NS FRITS Data model](#) to interact with the database and receive a result. The result is then forwarded to the API layer which serializes it to XML so it can be sent back through the XMPP layer.

XMPP Layer

The XMPP layer is largely based on the Smack API library, which is a Java library for XMPP communication. All communication in XMPP is handled by the use of streaming XML, encapsulating each message in special tags called stanzas. Two stanzas are being used in NS FRITS, the IQ stanza and the Message stanza. The IQ stanza is used for request-response mechanisms such as function and method calls, while the Message stanza is used for asynchronous messages, meaning messages that do not require any response. These two basic stanzas are then extended with additional information relevant to NS FRITS.

Two classes handle the extension of IQ and Message stanzas, **NSFRITSIQ.java** and **NSFRITSMessageExtension.java**, both inside the **eu.nsfrits.server.custommessages** package. These two classes are responsible of modelling each request and response internally and serializing them to XML.

To be able to create an internal model of a request or response, the stanza first need to be parsed. For this purpose several parsers have been written. The parsing classes can be found inside **NSFRITSIQImplementer.java**, **NSFRITSAlertImplementer.java**, **NSFRITSPositionImplementer.java** and **NSFRITSUpdatesImplementer.java** inside the **eu.nsfrits.server.custommessages** package. Each parsing class implements either a `parseIQ` or a `parseExtension` method, which takes an XML pull parser as a parameter (provided by the Smack Library), and returns an NSFRITSIQ or an NSFRITSMessageExtension object.

Whenever a packet is received by the server, it is matched against the registered extensions by looking at the namespace and tag name of the first tag inside the stanza. If a match is found, it is then sent to its respective parser and converted into an NSFRITSIQ or an NSFRITSMessageExtension object.

When the server first starts it registers several listeners, each waiting for a particular packet type. The listeners are then responsible of retrieving the content of each packet and calling the appropriate method in the API Layer. The listeners are located in the **eu.nsfrits.server.main**, and named **AsyncListener.java** and **NSFRITSIQListener.java**.

API layer

The API Layers purpose is to convert requests from the XMPP layer into XML data that can be sent back in responding packets. Since most of the data operations are handled at database level the API Layer is very simple, only bridging requests to the corresponding DAOs in the [NS FRITS Data model](#). The NS FRITS API class is named **NSFRITSAPI.java** and can be found in the **eu.nsfrits.server.api** package.

Data layer

The Data Layer is handled by the [NS FRITS Data model](#) application.

Operating environment

The NS FRITS core server application is purely built with J2SE 1.6, thus supporting all platforms implementing the Java platform.

Dependencies

The following Java Libraries are needed to run this application:

- Smack API 3.1.0 (smack.jar and smackx.jar)
Providing all XMPP communication

And the following Java projects are needed:

- NS FRITS Data model

(Note that the dependencies of NS FRITS Data model also need to be imported in this project)

Interfaces and classes including important functions**Classes**

<A UML diagram that summaries the classes that constitute the application and how they relate to each other>

External interfaces

The NS FRITS core server does not have any external interfaces.

Communication interfaces

The communication interface is described in [NS FRITS system specification - API](#).

Software interfaces and interrelation with other applications

<Functions, parameters - how to set them, where to call the functions from and what parameters to include in the function calls etc. Plus what the output would be if everything is okay>

User interface

The server is started running the NSFRITS-Server jar file. This is done by running the command `java -jar NSFRITS-Server.jar` inside the directory of the jar file. The first argument is mandatory and is the (DNS) name of the XMPP server. There are more optional parameters available represented by the name of the parameter an '=' sign and the value of the parameter.

The parameters currently available are:

- debug=(true/false)
If debug is set to true the server will display a Smack Debug window, displaying the raw XML data of each sent and received packet. The default value is false.
- security=(true/false)
If security is set to false, encryption of packets will be disabled, this can be useful for troubleshooting with a packet analyzer/sniffer. The default value is true.

Fault check, diagnostics and FAQ

General tips

If the core server is not working as expected, the first thing to check should be the log. Whenever the server runs, it will periodically check if any new subscription messages are pending. In case any exceptions are thrown, they will be displayed in the log. Note that some exceptions might occur because of database constraints, they should not be considered errors but instead a result of an erroneous input.

If the server fails to retrieve any results and no exceptions are shown, make sure that all imports are correct. Another way to troubleshoot is to bypass the communications part and call the DAO functions directly by writing a separate class, this can detect if the problem is related to XMPP/communication or the database/data model.

Test client

The core server can be tested by a test client distributed with the server. It is located in the test package **eu.nsfrits.server.testclient**. The client is started by running the **Sender.java** class.

Several test clients can be added by modifying the **Sender.java** class, while the **VehicleSim.java** can be modified to control what command(s) each client executes.

Appendix 6: NS FRITS system specification – Data inserter

Application description

The Data inserter application is an application used for automatic data insertion into NS FRITS from existing data sources. The application currently supports insertion of CSV-based (Comma Separated Value) data and (certain) XML data. Since it's impossible for the application to know all possible data formats around, the application has to be adapted when a new format is added.

Application features

- Converting and inserting CSV-based data sources into NS FRITS.
- Converting and inserting XML-based data in form of AVCIS Crime Hotspots.
- Provides several CSV-patterns to simplify insertion of new data.

Justification

Many companies and data providers publish location based information in the CSV-format for commercial GPS-applications. Information like road restaurants and petrol stations could be useful for HGV drivers, and therefore for the NS FRITS system. This application provides a way to easily parse and convert this kind of data into the NS FRITS format.

This application also provides a skeleton to write conversion modules for XML-based data by adapting to the current Crime Hotspot parser.

Application design

There are two main packages in the Data inserter, **eu.nsfrits.providers.main** and **eu.nsfrits.providers.parsepatterns**.

The **eu.nsfrits.providers.main** package contains runnable classes for inserting various kinds of data. At the moment, it contains sub packages for crime hotspots, petrolstations and restaurants. Each of these sub packages contains several runnable classes that each inserts data for a particular provider.

The **eu.nsfrits.providers.parsepatterns** package contains classes to simplify the parsing of CSV-based data. The class **SimpleCSVProvider** is responsible of converting a CSV-file to NS FRITS format and inserting the converted data into the database. When an instance of the **SimpleCSVProvider** is created, the user needs to specify a provider, a category, a type, a language and a pattern. The next section will describe how patterns work.

CSV Patterns

Since CSV-data exists in many different configurations, the class is instantiated with a pattern type variable. This variable defines what kind of pattern the CSV-file follows, and how the parser should interpret it. The advantage by using patterns is that the same pattern often reoccurs for different data providers, and can therefore be reused.

When a new data source is identified, the data format should manually be analyzed to identify if any of the existing patterns match the data. If so, that pattern could be used, otherwise it is possible to define a new pattern for that data format. The more patterns added, the higher probability it is that a match will be found.

An example of a pattern is to interpret the first two fields as coordinates, the third as NS FRITS title and the third plus the fourth as NS FRITS description. For example, this pattern would be useful for the CSV-data below about petrol stations from the company "JET":

```
12.55051,57.92909,"JET Alingsås", "E20 / Malmgatan"
12.89,56.25121,"JET Ängelholm", "Klippanvägen / Heimdallgatan"
15.88984,59.39598,"JET Arboga", "Arboga: Flygvägen 2"
12.94036,57.73055,"JET Borås/Skaraborgsvägen", "Borås: Skaraborgsvägen 32"
```

The result of the parser would generate InfoObjects like this (serialized to XML):

```
<infoobject>
  <id>37311</id>
  <provider>Jet</provider>
  <category>3</category>
```

```

<iotype>traffic-h3-1</iotype>
<validity>
  <from>2010-07-30T09:49:29+02:00</from>
  <to>2020-07-27T09:49:29+02:00</to>
</validity>
<sequenceid>0</sequenceid>
<location>
  <area>POLYGON((12.53051      57.94909,12.57051      57.94909,12.57051      57.90909,12.53051
57.90909,12.53051 57.94909))</area>
  <point>POINT (12.55051 57.92909)</point>
</location>
<datanode>
  <id>42710</id>
  <description>
    <title>JET Alingsås</title>
    <text>JET Alingsås E20 / Malmgatan</text>
    <language>en</language>
  </description>
</datanode>
</infoobject>

```

Operating environment

The Data Inserter application is purely built with J2SE 1.6, thus supporting all platforms implementing the Java platform.

Dependencies

The data inserter depends on the [NS FRITS Data model](#) project for database connections and helper functions such as reverse geocoding.

The following Libraries are needed to run this project:

- OpenCSV 2.2 (opencsv-2.2.jar)
Provides support for parsing CSV-data into Java objects.

Interfaces and classes including important functions

Classes

<A UML diagram that summaries the classes that constitute the application and how they relate to each other>

Software interfaces and interrelation with other applications

<Functions, parameters - how to set them, where to call the functions from and what parameters to include in the function calls etc. Plus what the output would be if everything is okay>

User interface

There is no user interface for the Data inserter. However, the data inserts prints out data in XML format after a successful parse.

Fault check, diagnostics and FAQ

Q: I get an exception, when trying to insert data from file x for provider y!

A: Make sure that the path to the file is correct and that the provider y actually exists before inserting. You need to create the provider first from the Provider Administrator.

Appendix 7: NS FRITS system specification – Data model

Application description

The NS FRITS data model application provides an interface to modify the content of the NS FRITS database. This application is never used as a stand alone application; instead it is used by other NS FRITS applications such as the NS FRITS server, the NS FRITS data provider and the Data inserter.

Application features

- Provides an interface for other applications to modify the content of the NS FRITS database in an object oriented way through Data Access Objects (DAO).
- Provides core classes modelling the content of the NS FRITS database, all serializable into JSON and XML format.
- Provides the ability to automatically translate text and generate text-to-speech audio files using Google's Translating services.
- Provides a reverse geocoding service from ws.geonames.org.
- Provides global project settings for other NS FRITS applications such as languages, formatting time, translation functions, proxy settings, etc.

Justification

- By focusing all database access to one application, bugs and errors can be limited to one place.
- Outlying applications do not need to have knowledge about what database and version that is being used. The database can be changed or moved without having to rewrite any of the other applications.
- Since access to NS FRITS is provided through DAOs, no knowledge of SQL or any other database language is needed to develop an application that uses the NS FRITS database.

Application design

The application is designed using the Data Access Object (DAO) design pattern. Several DAO classes are provided, each giving access to multiple Create Read Update Delete (CRUD) methods on various parts of the database.

The application is split into the following main packages:

- **eu.nsfrits.db.dao**
This package contains all DAO classes, which are used as an interface to the database.
- **eu.nsfrits.db.core**
This package contains all core classes, i.e. object representation of the database content.
- **eu.nsfrits.db.util**
This package contains utility methods and constants such as formatting time, checking if a language is valid, proxy settings, translation, etc.
- **eu.nsfrits.db.helper**
This package contains methods used for reverse geocoding.
- **eu.nsfrits.db.googlelets**
This package contains functions for text to speech conversion.

Data access objects

The DAO pattern is based on a simple class, DAO.java, which all other DAO-classes extend. To use a function in the DAO, the DAO object must first be instantiated with the default constructor. This lets the DAO-object fetch a connection from the database connection pool. It is important to always close the DAO object

after it has been used to return the connection to the pool and avoid blocking connections for other functions. This is best done by using a finally clause that closes the connection in case any exception is thrown.

If one wishes to use functions from several DAO-objects in the same class or function, it is possible to reuse connections by instantiating subsequent DAOs with the same database connection. The connection of a DAO can be fetched from the getConnection() function. If this is performed, only the initial DAO needs to be closed after usage.

There are five different Data Access Objects available as an interface to the database:

- **ProviderDAO**

Contains functions related to Providers, Categories, Types, Login, etc.

- **SpatialDAO**

Contains functions related to spatial operations. All spatial operations are executed at database level directly through SQL statements. This DAO only contains four public functions:

- getInfoObjectsAlongRoute, takes a username as parameter and returns all InfoObjects of the categories specified by that user intersecting with the route set by that user.
- getInfoObjectsNearPoint, takes a username and a point as parameters and returns all InfoObjects of the categories specified by that user intersecting with the supplied point.
- searchNClosestPOI, takes a category, point and a number, n, as parameters. It returns the n closest InfoObjects to the specified point of the specified category.
- getAllUsersWithin, takes an InfoObject as parameter, and returns all users that are currently within the InfoObjects area. This function only selects users that have the InfoObject's category selected.

- **TranslationDAO**

Contains functions related to the NS FRITS Translator service. For example, adding a translator job, getting all translator jobs or updating the translator status of a job.

- **UsersDAO**

Contains functions related to the NS FRITS Users, like setting user categories and routes, clearing user history, adding/deleting user subscriptions, etc.

- **ZoneDAO**

Contains functions related to InfoObjects, DataNodes, Attachments, ExternalOperators, ParameterData and Descriptions. This is the largest DAO class with the highest number of functions available.

Text-To-Speech

One of the implemented features was converting text into speech. This feature is based on Google's text-to-speech API but could easily be replaced with another existing text-to-speech service. The text-to-speech function takes three arguments: A text file, the language used and a path where the file should be stored.

The workflow of the function is the following:

- Split the text that should be converted into chunks with a maximum length of 100 characters (Google's API does not accept longer inputs than 100 characters).
- Call Google's API with each chunk of text. Each call results in an mp3-file containing the text-to-speech audio of that particular text.
- Convert the downloaded mp3-files into PCM wav files, and concatenate them into one large file.
- Finally convert the result into Speex (An open low sized audio format specialized for voice encoding) and store at the location received as an argument.

Reverse geocoding

The NS-FRITS datamodel application also provides reverse geocoding based on a web service from the free geocoding service: Geonames.org. Reverse geocoding is the process of transforming a geographical coordinate into a string address.

The geocoding service is available from the following url:

<http://ws.geonames.org/findNearbyPostalCodes?lat=57.7&lng=12>

Where the lat and lng variables are replaced with the coordinates latitude and longitude values. The service then replies in an XML format, which is parsed for the result.

The publicly available function is named `coordinateToAddress` inside the `ReverseGeoCoder` class in the **eu.nsfrits.db.helper** package. It takes a point and a connection as parameters and returns a string of the reversely geocoded location. If the reverse geocoding process fails, the function instead returns the coordinates as a string.

Operating environment

The core function of the application is purely built by using J2SE 1.6, thus supporting all platforms implementing the Java platform. The only function not platform independent is the text to speech function, which uses external operating system commands to move files.

Dependencies

The following Java Libraries are needed to run the core functionality of this application:

- Java Topology Suite 1.1 (jts-1.11.jar)
A library providing spatial operations to Java.
- Google Translation API (google-api-translate-java-0.92.jar)
A library wrapping Googles AJAX API for text translation.
- JSON-Simple (json_simple.jar)
A simple toolkit for building JSON-objects from Java.
- PostgreSQL JDBC (postgresql-8.4-701.jdbc4.jar)
A JDBC driver for the PostgreSQL database that is being used.
- c3p0 JDBC DataSources/Resource Pool (c3p0-0.9.1.2.jar)
Library for database connection pooling.

And the following are needed for text to speech:

- JLayer 1.0 (jl1.0.jar)
A library used for converting mp3 files into PCM wav.
- JSpeex (jspeex.jar)
Converting PCM wav into speex (the audio format used in NS FRITS).

Interfaces and classes including important functions

Classes

Class diagram over the core objects: [Datamodel class diagram](#)

Class diagram over all DAOs: [DAO class diagram](#)

External interfaces

Other applications use the NS FRITS data model by first instantiating the appropriate DAO object, then using the public functions inside the object to call the wanted method(s).

Communication interfaces

The NS FRITS data model does not support any external/network communication.

Software interfaces and interrelation with other applications

<Functions, parameters - how to set them, where to call the functions from and what parameters to include in the function calls etc. Plus what the output would be if everything is okay>

User interface

The NS FRITS data model has no user interface.

Fault check, diagnostics and FAQ

Q: The Text-to-speech feature does not work, how do I get it running?

A: Make sure the library and temp path are correct in the **TextToSpeech.java** class in the **eu.nsfrits.db.google tts** package. Also make sure that the proxy is correctly configured if you are using any, check the ProxySettings file in the eu.nsfrits.db.util package.

Q: None of the DAO-functions work, what is the problem?

A: Make sure that the address of the database is configured correctly in the **Configurations.java** class inside the **eu.nsfrits.db.util** package. Also make sure that the database is accessible from another interface such as the pgAdmin for postgres.

Appendix 8: NS FRITS system specification – Data provider

Application description

The NS FRITS Data provider application is the application used by independent data providers to insert data into NS FRITS. It is deployed as a web application, where each data provider can log in into their respective account (An account must in before hand be created from the [NS FRITS Provider administrator](#)). When logged in, all information entities created by that provider will be displayed on a map. The data provider can also create new information entities or modify existing ones by clicking on the map.

Each information entity contains multiple data nodes structured as tree where each data node contains a title, a text description. It is also possible to upload files from interface, which can be attached to data nodes. Furthermore a data provider can specify a time interval for the information entity, add parameter data such as free parking spots or temperature, and add a link to an [NS FRITS Agreement service](#). Application features

Application features

- Viewing information entities on a map.
- Creating/Deleting information entities.
- Modify the content of an information entity.
- Uploading files to an information entity.
- Adding an URL link to an information entity.
- Specifying an area to an information entity.
- Specifying a validity of an information entity.
- Adding an address to an Agreement Operator to an information entity
- Adding one or more parameter data to an information entity.
- Specifying a category of an information entity.
- Specifying an icon type of an information entity.
- Filtering so only information entities of a certain category should be displayed on the map.

Justification

This application is useful for independent data provider, i.e. smaller data providers that do not have any existing data source and only wishes to publish one or a few information entities in the NS FRITS system. They can use this application to easily create or update information and share it through the NS FRITS system.

Application design

The design of the data provider application is divided into two parts, the frontend and the backend. The frontend is written in a combination of JavaScript, CSS and HTML, while the backend is based on the Java Servlet API (J2EE).

The frontend

The frontend is mainly based on AJAX technologies so everything is displayed in one single HTML page, which is updated with JavaScript. All JavaScript are located in the **web/js** folder while the main page is located in **web/zoneedit.jsp**. A second file, **web/login.jsp**, is used for login in. Note that since the application is only a prototype, no security issues have been taken into account while designing the login feature.

For map support the JavaScript library, OpenLayers, is being used. It supports maps from several different providers, but currently the free map service OpenStreetMaps is being used. Several other JavaScript libraries are being used for various functions, Prototype is being used for AJAX-calls, Uploadify for file uploads and YUI for the icon tree.

When the user interacts with the GUI, the appropriate Servlet is called from the backend by a JavaScript. The backend will then perform the wanted action and return the response in the JSON-format, which then is parsed by the client side JavaScript. Finally the GUI is updated by the JavaScript with DHTML actions.

The backend

The backend consists of several Java Servlets located in the **eu.nsfrits.admin.servlets** package. The Servlets then use the [NS FRITS - Data model](#) application to access its DAO function to query the NS FRITS database. The results are then converted to JSON format before being returned to the frontend.

Unlike the Core Server application, the database connection pooling is not handled by the NS FRITS - Data model. Instead the connection pooling is declared in the projects **web.xml** file as a JNDI (Java Naming Directory Interface), so all servlets in the entire web application share the same connection pool.

Operating environment

The server can be run in any environment supporting Apache Tomcat. The application was tested version 6.0.20, but other version should probably work too.

The frontend requires the use of a web browser with JavaScript activated, currently the interface have been tested with Google Chrome 5.0.375, Safari 5.0.1 and Firefox 3.6.8.

Dependencies

The following libraries are needed to run the application.

- Commons FileUpload (commons-fileupload-1.2.1.jar and commons-io-1.4.jar)
Used for handling file upload at the server side.
- JSON Simple (json_simple.jar)
Library for parsing strings to JSON Objects and vice versa.
- c3p0 JDBC DataSources/Resource Pool (c3p0-0.9.1.2.jar)
Library used for database connection pooling.

The following projects are needed to run the application:

- [NS FRITS - Data model](#)

Interfaces and classes including important functions

Classes

<A UML diagram that summaries the classes that constitute the application and how they relate to each other>

External interfaces

The Data provider application does not have any external interfaces.

Communication interfaces

The Data provider application does not have any communication interfaces.

Software interfaces and interrelation with other applications

<Functions, parameters - how to set them, where to call the functions from and what parameters to include in the function calls etc. Plus what the output would be if everything is okay>

User interface

This section will be summarized in a series of screenshots, each describing use cases of the application.

NS-FRITS - Provider Editor

| | |
|--------------------------------|---------------------------------|
| Username: | tullverket |
| Password: | ***** |
| <input type="checkbox"/> Login | <input type="checkbox"/> Cancel |

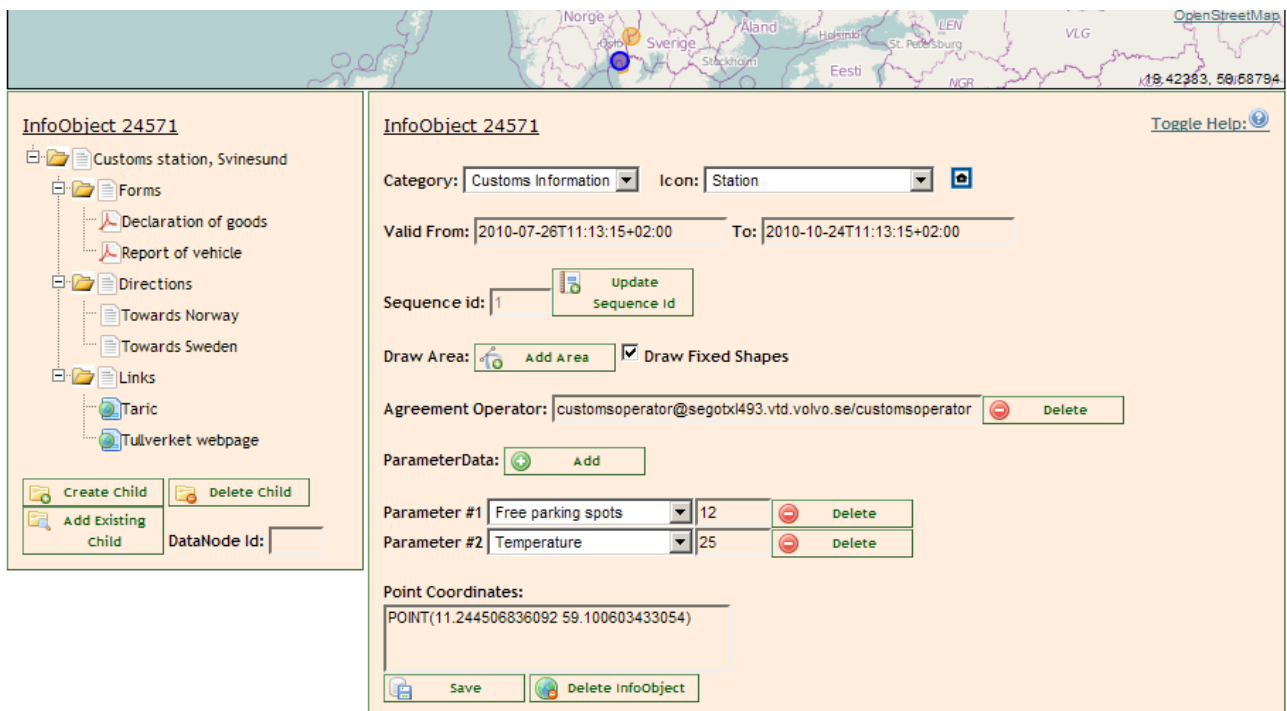
The provider logs in with its NS FRITS username and password. If correct, it will be redirected to its Provider page.

NS-FRITS - Provider Editor Logged in as: Tullverket Sverige

☒ Customs Information
 ☒ Crime Hotspots

The provider can now view each information entity it has created on the map; they are displayed as yellow round circles. In the top bar, the provider can choose to add new information entities, delete the selected information entity or delete all created information entities.

The checkboxes to the right displays what categories are being displayed right now. In this example, the provider has access of creating information entities in the Customs Information and Crime Hotspot category.



InfoObject 24571

Customs station, Svinesund

- Forms
 - Declaration of goods
 - Report of vehicle
- Directions
 - Towards Norway
 - Towards Sweden
- Links
 - Taric
 - Tullverket webpage

Create Child Delete Child Add Existing Child DataNode Id:

InfoObject 24571 [Toggle Help:](#)

Category: Customs Information Icon: Station

Valid From: 2010-07-26T11:13:15+02:00 To: 2010-10-24T11:13:15+02:00

Sequence id: 1 [Update](#) [Sequence Id](#)

Draw Area: [Add Area](#) ☒ Draw Fixed Shapes

Agreement Operator: customsoperator@segotx1493.vtd.volvo.se/customsoperator [Delete](#)

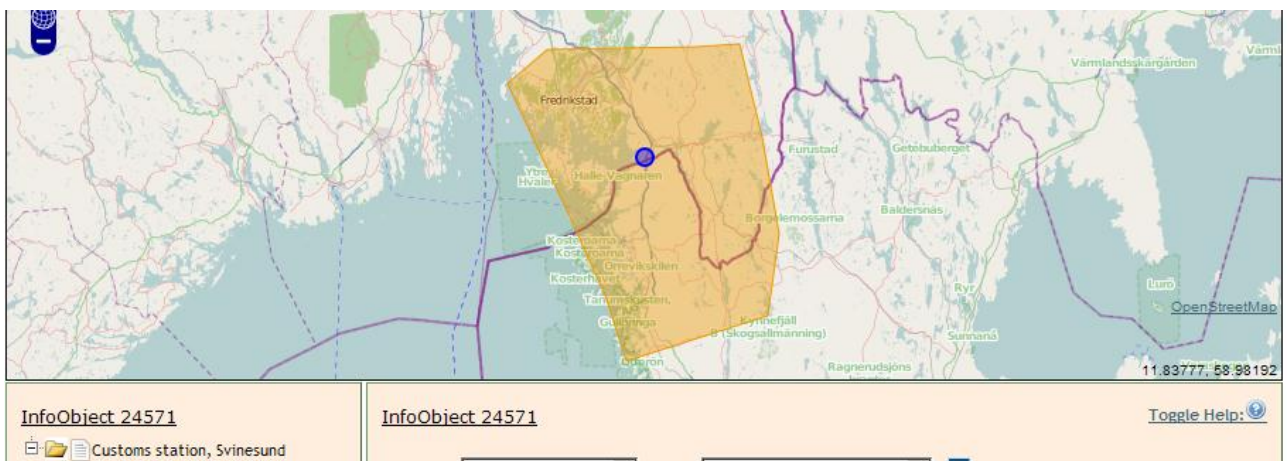
ParameterData: [Add](#)

| | | | |
|--------------|--------------------|----|------------------------|
| Parameter #1 | Free parking spots | 12 | Delete |
| Parameter #2 | Temperature | 25 | Delete |

Point Coordinates:
POINT(11.244506836092 59.100603433054)

[Save](#) [Delete InfoObject](#)

The data provider selects an information entity by clicking it on the map, all information bound to it will then be displayed under the map. In the left box all data nodes are displayed in a tree structure, while in the right box the information of the current object is displayed. A data provider can then click any node in the tree to display its containing data.



InfoObject 24571

Customs station, Svinesund

InfoObject 24571 [Toggle Help:](#)

Category: Customs Information Icon: Station

By using the Draw area button, a data provider can specify what area the information entity covers.

InfoObject 24571

- Customs station, Svinesund
 - Forms
 - Declaration of goods
 - Report of vehicle
 - Directions
 - Towards Norway
 - Towards Sweden
 - Links
 - Taric
 - Tullverket webpage

DataNode 38611 [Toggle Help:](#)

Descriptions:

| Title | Text | Lang | soundURL | Select |
|-------|--|------|---|--------------------------|
| Forms | Here you can find information of different forms | en | http://VTECW405.vcn.ds.volv.net:8 | <input type="checkbox"/> |
| Forms | Hier finden Sie Informationen über verschiedene Formen | de | http://VTECW405.vcn.ds.volv.net:8 | <input type="checkbox"/> |
| Формы | Здесь вы можете найти информацию о различных формах | ru | http://VTECW405.vcn.ds.volv.net:8 | <input type="checkbox"/> |

[New Description](#)
[Save](#)
[Edit Attachments](#)
[Delete](#)
[Back](#)

[Create Child](#)
[Delete Child](#)

[Add Existing child](#)
 DataNode Id:

When a node in the tree is selected, the text bound to it will be displayed in all different languages it has been entered in.

InfoObject 24571

- Customs station, Svinesund
 - Forms
 - Declaration of goods
 - Report of vehicle
 - Directions
 - Towards Norway
 - Towards Sweden
 - Links
 - Taric
 - Tullverket webpage

DataNode 38612 - Attachments [Toggle Help:](#)

Attachments:

| TypeName | Lang | URL | Select |
|----------------|------|---|--------------------------|
| fordonsanmalan | all | http://VTECW405.vcn.ds.volv.net:8084/NSFRITS-Admini | <input type="checkbox"/> |

[Create Link](#)
[Update](#)
[Save](#)
[Delete](#)
[Back](#)

Press Browse to select file(s) to upload as attachments:

[BROWSE](#)

By pressing the 'Edit Attachment' button, the provider can upload files or attach web links to a data node.

Appendix 8: NS FRITS system specification – DATEX connector

Application description

The DATEX Connector is an application that periodically connects to a DATEX II source, fetches data from it, converts the DATEX II data into the NS FRITS format, and finally inserts it into the NS FRITS database. This application has specifically been developed to function with DATEX data from the National Traffic Control Center (NTCC) in the UK, but the application should be adaptable to fit any other DATEX II provider fairly easily.

Application features

- Parse DATEX II files and convert them into the NS FRITS format.
- Runnable as a service that periodically fetches DATEX II and filters out data that has not yet been inserted.
- Mechanisms to detect erroneous content from DATEX II providers, to avoid corrupting the NS FRITS database.
- Ability to fetch compressed data to avoid unnecessary traffic.
- Remove entries in NS FRITS as their validity date expires.

Justification

Today many European countries use the DATEX II standard for publishing information about traffic disturbance, road maintenance work, resting places and road weather. Since this type of data has been identified to be crucial for the NS FRITS system, an application fetching and parsing DATEX II data is needed.

Application design

The application is based on three projects, the NSFRITS-RoadAdministrationConnect, the NSFRITS-DATEXParser, and the NSFRITS-Datamodel. The RoadAdministrationConnect project is responsible of fetching the actual data from the data source. The DATEXParser project is responsible of parsing the data, finding out if it is already inserted, and if not - converting the data into the NS FRITS format. The Datamodel project is finally responsible of inserting the new data into the database.

Fetching the data

The DATEX II data source keeps the data split up into different categories, each containing information about events for that particular subject. Examples of categories are: CurrentRoadworks, CurrentPlanned and UnplannedEvents. Each category consists of two XML files; Content.xml and Metadata.xml. The Content.xml file contains the payload of all events (also called situations) in that category, while the Metadata.xml only contains timestamps of the last time this category was updated with new situations.

The main class responsible of fetching the data is located in the **eu.nsfrits.datexpuller** package and called **DatexConnector.java**. This application consists of an infinite loop that every third minute fetches the Metadata.xml for each category to check if any new DATEX II situations have been added. This is done by keeping a time stamp of each category's latest update time in memory. If the timestamp in the fetched Metadata.xml is the same as the one in memory, no situations have been added since the last update so the application moves on to the next category. If the timestamp instead have a newer value, then one or more situations have been added since the last update, so the new Content.xml file for that category is fetched.

To be able to know what situations that already have been loaded, the application uses a hashmap that maps loaded situations to their NS FRITS InfoObject id. This map is also persisted to a database using a write-through technique to be able to handle application restarts without having to reload all entries. When the application starts a reference to this hashmap is sent to the parser.

Parsing and converting

Each time a new Content.xml file is fetched it is sent to the parsing class, which is responsible of parsing the XML and inserting new objects into the NS FRITS database. The parsing method is based on w3c's DOM (Document Object Model) parser.

When the parser has loaded the XML document, it fetches the id attribute of all situation tags. The ids are then matched against the hashmap described in 0 to see if the situation has already been loaded. If so, that id is skipped. The parser also utilizes an ignore list to skip objects that for some reason are faulty, for example if their validity have expired or have duplicated content with another situation.

If the parser discovers a new situation, it will run its parsing routine on the inner tags of the situation. The parsing routine is written based on examination of the raw data, where the most important types and values are selected. The values are then converted into an NS FRITS InfoObject and added to a list together with its situation. When the parser has gone through each situation, this list is sent to the Data Inserter Module.

Inserting the data

The Data Inserter receives a list of InfoObjects from the parser, for each entry in the list it inserts the object into the database. If the insert is successful, it also adds the situation with the corresponding InfoObject id to the situations-hashmap.

Operating environment

This application is runnable on any system supporting the Java Virtual Machine and JRE 1.6.

Dependencies

The following libraries are needed:

- PostgreSQL JDBC (postgresql-8.4-701.jdbc4.jar)
A JDBC driver for the PostgreSQL database that is being used.
- c3p0 JDBC DataSources/Resource Pool (c3p0-0.9.1.2.jar)
Library for database connection pooling.

The following projects are needed:

- [NS FRITS - Data model](#)

Interfaces and classes including important functions

Classes

A class diagram of the containing classes of the application can be found here: [DATEX Connector classdiagram](#)

Software interfaces and interrelation with other applications

<Functions, parameters - how to set them, where to call the functions from and what parameters to include in the function calls etc. Plus what the output would be if everything is okay>

User interface

This application does not have any user interface, but when running it prints log data onto the standard output.

Fault check, diagnostics and FAQ

<If the application is not doing what it should be doing, what would be some typical diagnostic elements to look out for?>

Appendix 9: NS FRITS system specification – Provider administrator

Application description

The NS FRITS Provider Administrator application is an application that allows creating, listing, updating and deleting of providers and categories. Like the provider interface, the provider administrator is deployed as a web application.

Application features

- Create/delete categories.
- Add descriptions to categories.
- Create/Delete providers
- Set login username/password for providers.
- Set what categories each provider is allowed to create information objects in.
- Create/Read/Update/Delete information objects for each provider.

Justification

This application is useful by the administrators of NS FRITS to be able to add new providers into the system without manually having to edit the database.

Application design

See NS FRITS - Data provider.

Operating environment

See NS FRITS - Data provider.

Dependencies

See NS FRITS - Data provider.

Interfaces and classes including important functions

Classes

<A UML diagram that summaries the classes that constitute the application and how they relate to each other>

External interfaces

The Provider administrator application does not have any external interfaces.

Communication interfaces

The Provider administrator application does not have any communication interfaces.

Software interfaces and interrelation with other applications

<Functions, parameters - how to set them, where to call the functions from and what parameters to include in the function calls etc. Plus what the output would be if everything is okay>

User interface

This section will be summarized in a series of screenshots, each describing use cases of the application.

NS-FRITS - Provider Editor

Welcome to NS-FRITS provider editor!

- Aral
- AVCIS (Crime Hotspots)
- BP DE
- BP UK
- Fordonsgas
- Gothenburg parking company
- Jet
- Max Sweden
- McDonalds Sweden
- National Traffic Control Center
- OKQ8

Welcome to the NS-FRITS Administrator Interface

Select a provider in the menu to the left to edit its name or allowed categories.

Choose Create provider to add a new provider to the NS-FRITS system.

Manage Categories controls which categories are available in NS-FRITS. You can either edit the current ones, or add new ones.

When the provider administrator is started, the user can either select a provider from the menu to the left to start editing its content, choose to create a new provider by the “Create Provider” button or manage the categories in NS FRITS with the “Manage Categories” button.

NS-FRITS - Provider Editor

Welcome to NS-FRITS provider editor!

- Aral
- AVCIS (Crime Hotspots)
- BP DE
- BP UK
- Fordonsgas
- Gothenburg parking company
- Jet
- Max Sweden
- McDonalds Sweden
- National Traffic Control Center
- OKQ8
- Preem Sweden

Modify provider view:

Title: BP DE

Username: bpde

Password:

Select Provider Categories:

☐ Truck Stops
 ☐ Customs Information
 ☐ Traffic Information
 ☒ Petrol Stations
 ☐ Restaurants
 ☐ Secure Parking
 ☐ Crime Hotspots
 ☐ Road Accidents

By selecting a provider, the administrator can setup that particular provider’s username and password. It is also possible to check what categories the provider is allowed to edit. By pressing the “Edit Zones” button, the administrator can view and edit all InfoObjects for that provider.

NS-FRITS - Provider Editor

Welcome to NS-FRITS provider editor!

- Aral
- AVCIS (Crime Hotspots)
- BP DE
- BP UK
- Fordonsgas
- Gothenburg parking company

Provider Title:

New Provider

Username:

newprovider

Password:

.....

The “Create Provider” button allows new providers to be created and added into the NS FRITS system.

NS-FRITS - Provider Editor



Select Category to modify:

- Truck Stops
- Customs Information
- Traffic Information
- Petrol Stations
- Restaurants
- Secure Parking
- Crime Hotspots
- Road Accidents

Add Category

Delete Category

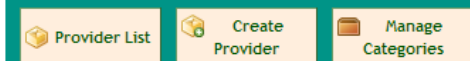
NS-FRITS Category Editor

Select a category in the menu to the left to edit its description. The Add Category button adds a new category, while the Delete Category button removes one.

After a category has been selected, it is possible to edit/view/create/delete the description texts corresponding to that category.

The “Manage Categories” section lists all current categories in the left menu and allows categories to be added and deleted by the “Add Category” and “Delete Category” buttons.

NS-FRITS - Provider Editor



Select Category to modify:

- Truck Stops
- Customs Information
- Traffic Information
- Petrol Stations
- Restaurants
- Secure Parking
- Crime Hotspots
- Road Accidents

Add Category

Delete Category

Category 1

Descriptions:

| Title | Text | Lang | soundURL | Select |
|---------------------|--------------------------------------|------|-------------------------------------|--------------------------|
| Tullinformation | Information om tullstationer | sv | | <input type="checkbox"/> |
| Customs Information | Information about customs procedures | en | http://VTECW405.vcn.ds.volvoo.net:8 | <input type="checkbox"/> |



New Description



Save



Delete



Back

By selecting a category, it is possible to edit the description of that category and add new descriptions in other languages.