ROad **S**afety **ATT**ributes exchange infrastructure in **E**urope

D3.2 - Software modules for data exchange

Version Number: 10
Produced by: NAVTEQ
Due Date: 31-05-2010
Release Date: 12-11-2010

ROSATTE is co-financed by the European Commission – DG INFSO
Contract Number: 213467

Programme Name	7th Framework Programme – Specific Programme Cooperation – Theme 3 "Information and Communication Technologies"
Grant Agreement Number:	213467
Project acronym:	ROSATTE
Project name:	ROad Safety ATTributes exchange infrastructure in Europe
Project start and end:	January 2008 – December 2010 (36 months)
EC Project Officer:	Mr. Emilio Davila-Gonzalez
	E-mail: emilio.davila-gonzalez@ec.europa.eu
Dissemination level[1]:	Public

Title of document:	Software modules for data exchange
Work package:	WP3
Editors:	Ahmed Nasr, Kees Wevers - NAVTEQ (Editors)
Project co-ordinator:	Maxime Flament (ERTICO – ITS Europe)
	Tel: +32 2 400 07 35, fax: +32 2 400 07 01
	E-mail: m.flament@mail.ertico.com

Abstract: This document summarizes the development of reference date exchange software modules/services for implementation at the data provider side and for validation

Keyword list: Software modules, services, data exchange, safety feature, location reference

[1] This is either: Public, restricted to other programme participants, restricted to a group specified by the consortium, confidential

Document control sheet

Editors	Ahmed Nasr, Kees Wevers - NAVTEQ
Work area	WP3
Document title	Software modules for data exchange
Document reference	RST_WP3_D3.2_v10.doc10

Version history:

Version	Date	Main author	Summary of changes
0.1	23-02-2010	Ahmed Nasr (NT)	Document created
0.2	04-03-2010	Ahmed Nasr (NT)	Outline
0.3	07-03-2010	Kees Wevers (NT)	First draft
0.4	20-04-2010	Stephen T'Siobbel (TA) Michael Lanigan (TfL) Dominique Guichon (LR) Nele Dedene (FL) Ramadane Mahiou (AS) Lars Wikström (TR)	Input partners
0.5	10-05-2010	Ahmed Nasr (NT)	Update and format
0.6	14-05-2010	Kees Wevers (NT) Michael Landwehr (PTV)	Input NT Input OBB
0.7	25-05-2010	Ahmed Nasr (NT)	Final editing (peer review version)
0.8	17-06-2010	All partners	Several updates from all partners as respond to the peer reviewer comments
0.9	29-06-2010	Kees Wevers & Ahmed Nasr (NT)	Review and update answers to peer reviewers (Final version)
10	12-11-2010	Ahmed Nasr, Kees Wevers (NT)	inclusion of updates from test sites, addition of paragraphs 5 and 6, finalisation of document

Approval:

	Name	Date
Prepared	Ahmed Nasr, Kees Wevers (NT)	12-11-2010
Reviewed	Jean-Charles Pandazis	12-11-2010
Authorized	Maxime Flament	12-11-2010

Circulation:

Recipient	Date of submission
ROSATTE Consortium	12-11-2010
EC	12-11-2010

Executive summary

This document summarises the general outline of the common and specific software modules used for data exchange throughout the different test sites. The software modules are necessary at the one hand to ensure similarity of data exchange and the other hand to ensure ease of application at each test site individually according to its specific conditions. The main purpose of this document is to provide a description of the efforts made by each of the test sites, as well as a global description of the commonalities and differences.

This document has a direct link to the deliverables D3.1 (Specification of data exchange methods) and D2.2 (Implementation of tools for data maintenance). While D3.1 provides a technical ground to apply common services like the REST architecture, a clear synergy with D2.2 is apparent due to the nature of the two documents aiming at describing applied technologies and solutions for software implementations at the test sites, a fusion between the two document was considered but the consortium decided to keep with the original plan according to the DoW.

The core part of the document provides an overview of two interrelated components of the implementations:

- Commonly used software modules/services (shared technology by all test sites); these include the REST service and the AGORA-C webservice.
- Specific software modules for each test site. The related sections provide details for the different specific software modules, which have been developed according to the needs of each test site.

In annex 1-3, examples of xml schemas, as used in the different test sites, are provided.

Table of contents

Index of tables

Index of figures

1. Project description

1.1 ROSATTE contractual references

ROSATTE is a STREP submitted for the call FP7-ICT-2007-1. The acronym stands *for ROad SAfety ATTributes exchange infrastructure in Europe.* The Grant Agreement number is 213467 and project duration is 30 months, effective from 01 January 2008 until June 2010. It is a contract with the European Commission, DG INFSO.

The EC Project Officer is:

Emilio Davila-Gonzalez
EUROPEAN COMMISSION
DG INFSO - G04
Office: BU31 - 5/41
B - 1049 Brussels
Tel: +32 2 298 76 12
E-mail: emilio.davila-gonzalez@ec.europa.eu

1.2 Project objectives and scope

The ROSATTE project intends to develop the enabling infrastructure and supporting tools that will ensure Europe-wide access to safety road attributes with a focus on incremental updates. This infrastructure is intended to facilitate a continuous supply of such data at a high and steady level of quality from the parties that administer and control the attributes to third parties, and thereby to help maintain near-permanent correctness of such data for use in road safety applications. In addition, the infrastructure will serve administrative internal functions at data providers, which will in turn be beneficial for the system of safety road attributes as a whole. Improved and more extended availability of up-to-date safety road attributes is expected to result in improved and extended functionality of driving assistance systems, and thereby to contribute to more efficient road operations and increased traffic safety.

The flow of data that is addressed in ROSATTE may be seen as a data chain, which is depicted in the upper part of Figure 1. Public road administrations and other road operators, which are seen as the most efficient and reliable source for update information, are at the beginning of the chain. Processes to define, install and change safety road attributes, like issuing of regulations and work orders, are not included in the scope of the project. ROSATTE looks only at the outcome of these processes, and especially at incremental changes in attributes. For this, it will, on the data provider side, study database storage of attribute information and update mechanisms, and methods for extracting the information, both complete data sets and incremental updates.

The extracted information is the input for the data exchange infrastructure, which is the topic of this document. On the user side, the project will study data integration, especially at providers of digital map databases for navigation systems and other driving assistance applications. The data chain from map providers to in-vehicle systems, which is depicted in the lower part of Figure 1, is not part of the project scope. Of this chain, especially the part representing incremental updating of the in-vehicle map database at regular, short

intervals, will benefit from data chain that ROSATTE intends to realize while at the same time providing a rationale for the project.

Setting up this chain has a clear benefit for public authorities and road operators through its potential contribution to improving road traffic safety, while giving the industry the opportunity to improve the quality of map databases used in in-vehicle systems, and enabling new safety applications that need map data with Europe-wide complete and up-to-date coverage of road safety attributes.

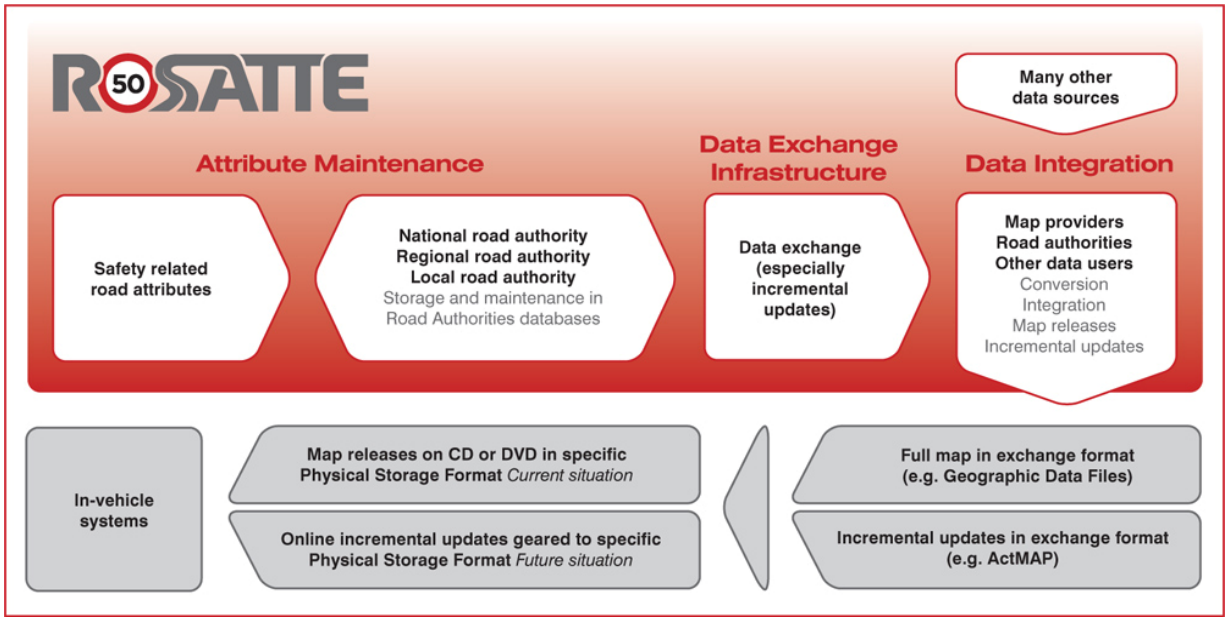


Figure 1 - The scope of the ROSATTE project

1.3 ROSATTE project deliverables available to date

Since the start of the ROSATTE project in 2008, a number of reports have been finalized. Those reports labelled for public dissemination are available via the ROSATTE project website (www.rosatte.eu). The table below gives a comprehensive overview of available reports to date.

D1.1 State of the Art
Describes the current road authorities and infrastructure operator's situation with respect to how safety relevant data is stored, exchanged and updated.
D1.2 Requirements and Overall Architecture
Defines the project scope, user, user requirements and derived system requirements. It also gives suggestions on information model and a high-level system structure.
D2.1 Conceptual specification on how to establish a data store
Presents a conceptual description of the data maintenance operations with regards to functional view (use cases), process view, information view, and component view.
D3.1 Specification of Data Exchange Methods

Presents the exchange specification consisting of a data content specification, a physical exchange format specification, (to be completed) and a service specification.
D7.2 Brochure
The project brochure is to promote the project results in liaison with major events.

Table 1 - Overview of available project deliverables

The project deliverables in Table 1 are available for download at the following link:
http://www.ertico.com/en/activities/safemobility/rosatte/publications/publications.htm

2. Introduction

2.1 Objectives of WP3

The objectives of WP3 are:

- To enable the automatic and timely exchange of safety attributes between road authorities and potential users of such data.
- To provide, test and validate reference implementations for the exchange of data

2.2 Purpose of document

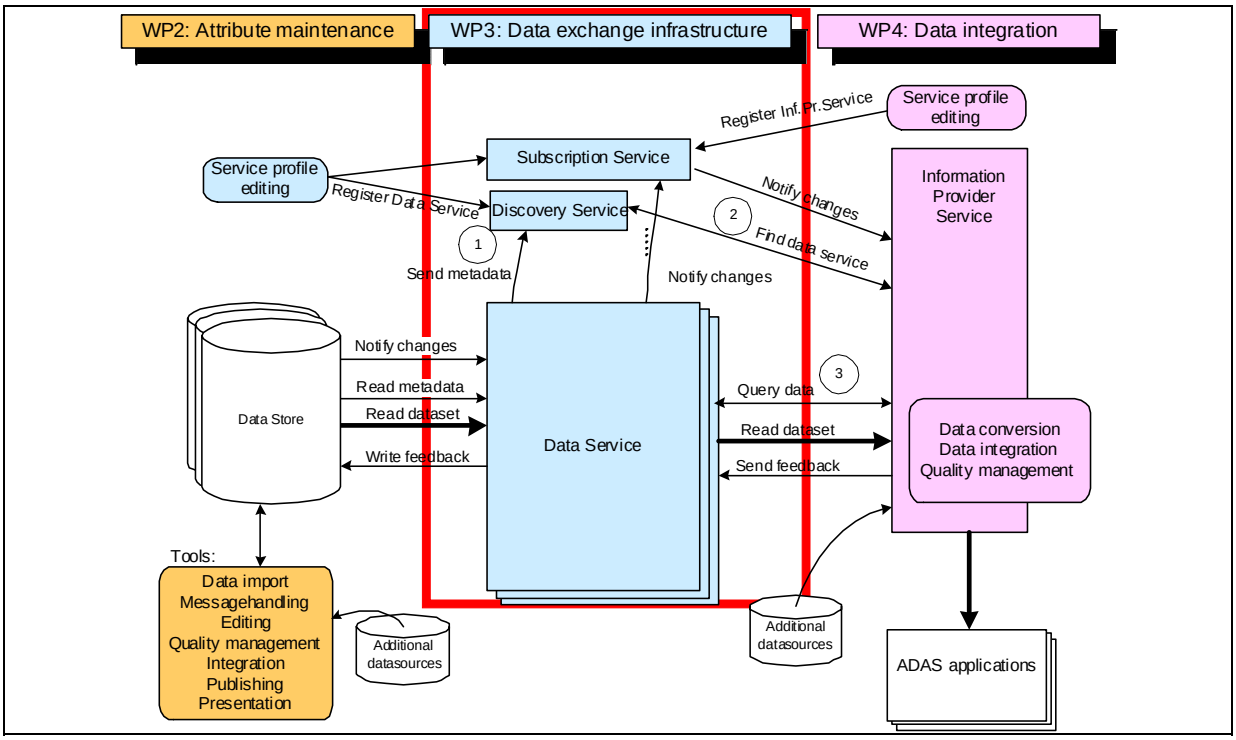


Figure 2 - Overview of work packages and data flow – scope of WP3 in red box

Figure 2 shows the components and the data flow in the ROSATTE project (from D1.2).

- WP 2 is concerned with data maintenance tools, i.e. processes involved to enter and maintain data in the data store. The implementation work in this domain in subject of the deliverable D2.2.
- WP 3 is concerned with the extraction of data from the data base and the transfer of data to the information provider (map maker) including the location reference transfer. The implementation work in this domain in subject of the deliverable D3.1 and the present deliverable (D3.2).
- WP4 is concerned with the integration of the received data into the information providers' database. The implementation work in this domain in subject of the deliverables D4.1 and D4.2.

The objective of this document is to summarize the development of reference data exchange software modules/services which have been implemented at the data provider side and have been used for validation of the ROSATTE data chain.

In addition, the document will elaborate the adaptation of the developed software modules to the specific environments of each of the test sites, and the implementation and testing of the modules.

This document fulfils the stated objectives by identifying two different categories of the implementations:

- Commonly used software modules (shared by all test sites);
- Specific software modules for each test site.

A common structure was used when requesting the test sites for their respective contributions for this deliverable. It should be taken into account that the main purpose of this document is to provide a description of the efforts made by each of the test sites, as well as a global description of the commonalities and differences according to certain characteristics.

As this document describes the solutions for software implementation, which have been implemented in each test site, there is a direct synergy with the D2.2 deliverable.

2.3 Structure of document

This document is structured in two major sections. First the commonly used software modules/services are described in chapter 3. These include the REST service. Next to this, also the location reference webservice implementations of the map makers are presented, which are needed for encoding and decoding of the AGORA-C location references which are part of the update information.

In chapter 4, for each test site the specifics of the implemented software module are summarised.

For each test site, each implemented software module/service is described using the following key elements:

- Purpose of the software module/service (especially which safety feature is targeted)
- General description
- Component diagram
- Screen shots

Chapter 5 provides a summary for definitions and acronyms used in the document.

3. Commonly used software/services

3.1 REST service

The REST architecture and schema is defined in D3.1 is considered as a common service that is further developed and applied in each test site separately. Different implementations are explained under each test site under paragraph 4 (test site specific software).

3.2 NT AGORA-C online webservice

3.2.1 Introduction

The existing basic version of the NAVTEQ AGORA-C webservice for encoding and decoding locations was substantially improved and extended for use in the ROSATTE project. The service a non-commercial service, and is only intended for R&D purposes. The service has two different components: the http service, and the visualisation service. The service is proprietary, and a user name and password are required to access the service.
Figure 3 shows The implementation of NAVTEQ's REST service with AGORA-C service and further connection to the Test sites's services.

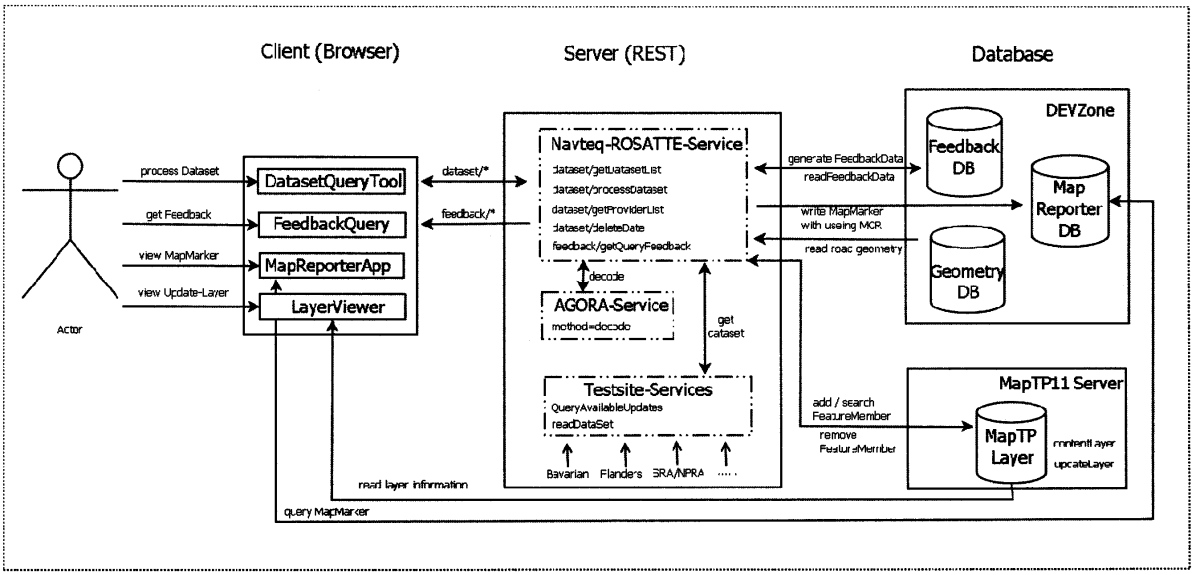


Figure 3 – NAVTEQ's REST and AGORAC-C implementation

3.2.2 The http service

The http service defines a specific link format for access of the service, which enables to include in the link specific information for either the encoding of a chosen location, or the decoding of an existing AGORA-C code string. For encoding, a location is defined in terms of one or more (consecutive) NAVTEQ link-IDs with additional attributes, especially offsets and direction. The service answers with providing the AGORA-C code string. Similarly, for

decoding, an AGORA-C code string is packaged in a string according the prescribed format, and the service will return a set of (one or more) NAVTEQ link-IDs, offsets and direction. The http service is in the first place meant for automation from within a program.

The service can address AGORA-C versions 1.0, 2.5 and 3.0. Version 1.0 refers to the original version as defined in the AGORA-C specification document of 6 April 2005[2], version 3.0 to the version as defined in ISO standard 17572 Part 3[3]. Version 2.5 is an intermediate version. A list of valid versions can be requested from the server. The service operates with different available map versions. A list of available map versions can be requested from the server. Both AGORA-C version and map version are included in the string that is sent to the server.

The following link provides an example encoding request for AGORA-C version 3.0 and map version Q409, together with the AGORA-C code that is returned:

http://rtmdemo.navteq.com/ame/db/agora-webservice/AgoraWebService?method=encode&mapversion=Q409&agoraversion=3.0&startoffset=0.3&endoffset=0.2&edges=542841337;-545820742;-545820740;-542837896;-542837923

ATYAMAAyACAGBBUAxCz+GbAeeQDbCBbPQASAAAMDVEVQBBUAxCz+GUoeeOybCADPQASAAAMDSkVT

The next link provides an example decoding request for the same map version, and using the result of the above encoding, together with the returned result:

http://rtmdemo.navteq.com/ame/db/agora-webservice/AgoraWebService?method=decode&mapversion=Q409&agorastring=ATYAJQAyABAGBBUAxCz+GbAeeQDbCBbPQASAAAMDVEVQBBUAxCz+GUoeeOybCADPQASAAAMDSkVT

agoraversion=3.0
startoffset=0.3216951490711576
endoffset=0.20282604849495003
locationdirection=aligned
edge=+542841337
edge=-545820742
edge=-545820740
edge=-542837896
edge=-542837923

3.2.3 The visualisation service

The visualisation service with visual map interface enables decoding and visualisation of AGORA-C code strings via manual insertion or xml file upload. In addition, it enables manual encoding of a specific point or linear location by clicking a point (for a point location), or a series of two or more points (for a linear location) in the map on the screen.

[2] Hendriks, T., K. Wevers, M. Hessling and H.-W. Pfeiffer, "Specification of the AGORA-C on-the-fly location referencing method", version 1.0, 06 April 2005.

[3] ISO, "Intelligent Transport System (ITS) - Location referencing for geographic databases - Part 3: Dynamic location references (dynamic profile)", ISO/FDIS 17572-3:2008, 05-09-2008, International Organization for Standardization (ISO), Geneva.

Furthermore, the service can display speed limit categories, allows zooming in to an inserted latitude/longitude position. The visualisation service is useful for testing, validation and inspection activities.

Figure 4 shows the NAVTEQ's Rosatte portal interface used as a visualisation tool for the AGORA-C web service and Figure 5 shows a snap shot of an update layer after being decoded.

Figure 4 – NAVTEQ's Rosatte portal interface

Figure 5 – NAVTEQ's update layer

3.3 TA AGORA-C online webservice

The Tele Atlas R&D AGORA-C webservice was set up in order to provide the functionality to encode map locations to AGORA-C binary codes and to decode AGORA-C binary codes to map locations based on the Tele Atlas MultiNet Europe data. A more detailed description can be found in deliverable D4.2. The high level characteristics of the AGORA-C webservice:

- supports encoding/decoding of point and line locations,
- supports different Multinet products. A request to the AGORA-C webservice allows the user to define the source map version of a list of defined IDs or the target map version used to decode an AGORA-C binary code. (e.g. Europe 2009.02),
- All encoding results are expressed in the AGORA-C physical binary format and encoded as BASE64 code to ensure a user-readable form,
- supports different versions of the en- and decoder physical binary format (e.g. "3.0" as defined in the published International Standard (ISO17572:2008)).

The webservice is available at the following URL via HTTPS POST:
https://innovationcenter.teleatlas.com/LocationTranslation/Translation

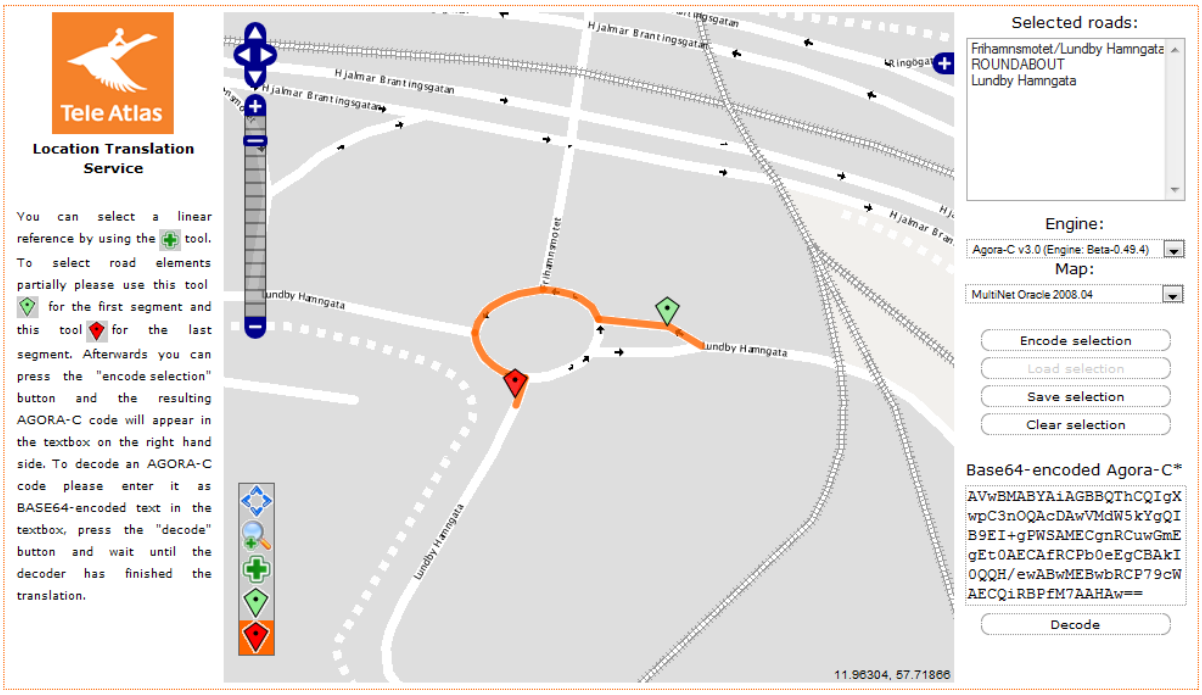


Figure 6 – Screenshot of a demo application using TeleAtlas web-service

See example in annex

The XML structure of the request is based on the following schemas:
https://innovationcenter.teleatlas.com/LocationTranslation/schemas/ta/LocationTranslation.xsd

See example in annex

https://innovationcenter.teleatlas.com/LocationTranslation/schemas/ta/ADTLT.xsd

See example in annex

4. Test site specific software

4.1 Test site ASFA (France)

4.1.1 Summary of used software/services

ASFA test site technical architecture:

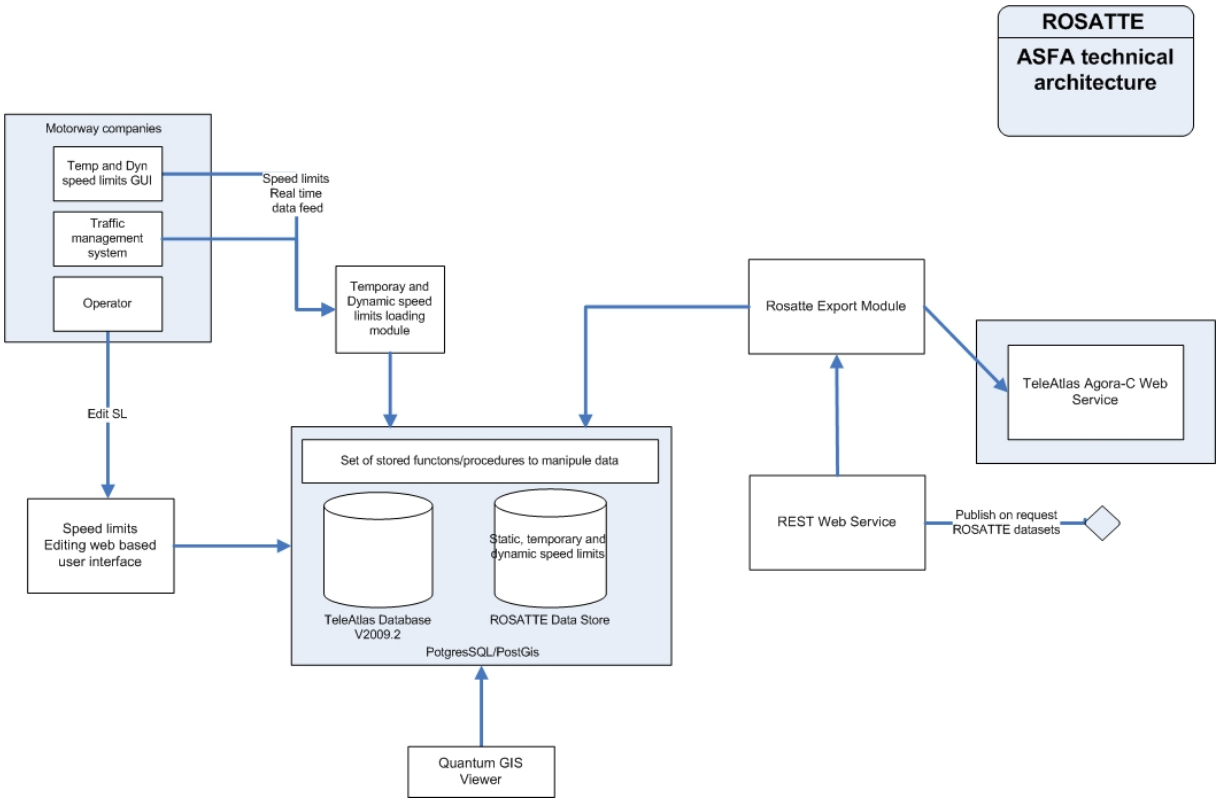


Figure 7 - ASFA test site technical architecture

ROSATTE data store

The data store has been built on a PostgreSQL database with integrated Postgis spatial cartridge. The name of the PostgreSQL database for the management of speed limits is *"vitesselimite"*.
The Teleatlas 2009.02 road database has been installed on PostGreSQL.

2 modules are involved in the data exchange mechanism:

1 – Export module

This module prepares data for publishing by doing the following action:

- extracts TeleAtlas map Ids for each linear location to feed the TeleAtlas AGORA-C webservice (see 4.1.2 for detailed description)
- encode linear location in AGORA-C by requesting the Tele Atlas AGORA-C webservice see 4.1.3 for detailed description)

- exports datasets in ROSATTE XML format (see 4.1.4 for detailed description)

2 - Data publishing module

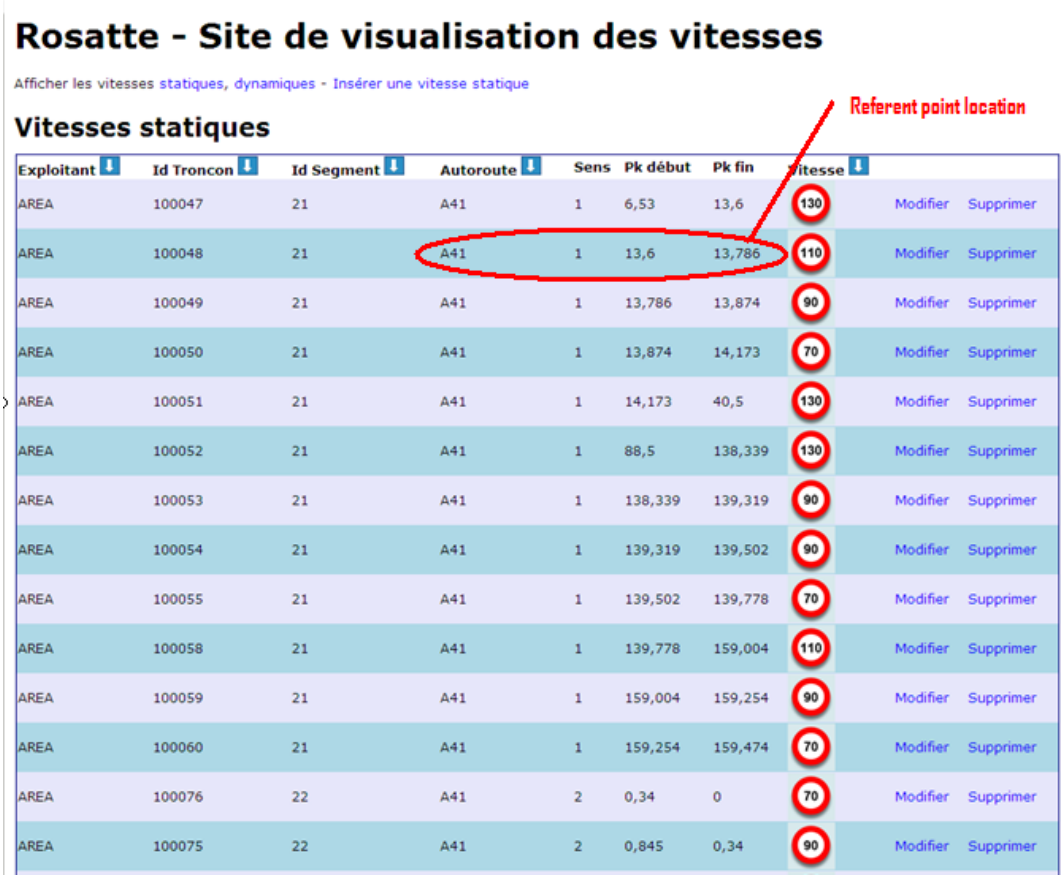
- This module is the REST webservice that publishes ROSATTE datasets (see 4.1.5 for detailed description)

These modules and the attached services will be described in the following sections.

4.1.2 Export module /extract TA map ids

Data are inserted in the ROSATTE data store through a web based user interface:

Figure 8 – Web-based ROSATTE data store at ASFA



Exploitant	Id Troncon	Id Segment	Autoroute	Sens	Pk début	Pk fin	Vitesse		
AREA	100047	21	A41	1	6,53	13,6	130	Modifier	Supprimer
AREA	100048	21	A41	1	13,6	13,786	110	Modifier	Supprimer
AREA	100049	21	A41	1	13,786	13,874	90	Modifier	Supprimer
AREA	100050	21	A41	1	13,874	14,173	70	Modifier	Supprimer
AREA	100051	21	A41	1	14,173	40,5	130	Modifier	Supprimer
AREA	100052	21	A41	1	88,5	138,339	130	Modifier	Supprimer
AREA	100053	21	A41	1	138,339	139,319	90	Modifier	Supprimer
AREA	100054	21	A41	1	139,319	139,502	90	Modifier	Supprimer
AREA	100055	21	A41	1	139,502	139,778	70	Modifier	Supprimer
AREA	100058	21	A41	1	139,778	159,004	110	Modifier	Supprimer
AREA	100059	21	A41	1	159,004	159,254	90	Modifier	Supprimer
AREA	100060	21	A41	1	159,254	159,474	70	Modifier	Supprimer
AREA	100076	22	A41	2	0,34	0	70	Modifier	Supprimer
AREA	100075	22	A41	2	0,845	0,34	90	Modifier	Supprimer

Figure 9 – Speed visualisation at ROSATTE web-based data store

The location system used by French motorway companies is very specific: a location is defined by a road name, a direction, a start point (kilometric road point) and an end point (kilometric road point). Such location cannot be understood by the map providers. A location conversion module has been implemented to be able to provide locations in the expected system. The service runs for each location the following service:

at_routepksens_2_xy: convert French motorway companies locations to geographic locations (WGS84) and extract the related Tele Atlas map Ids.

Implementation:

- PostgreSQL stored function *at_routepksens_2_xy*
- Result in 2 PostgresSQL tables *routepksens2xy_result* and *routepksens2ta_result*
- Call for a given location the stored function *at_routepksens_2_xy* and access to the results through a .Net Framework 3.5 C# DLL : *convertPRToTA.dll*

The extracted TeleAtlas map ids will be used as inputs to the TeleAtlas webservice to encode the locations in AGORA-C.

4.1.3 Export module / encode linear location in AGORA-C

The Tele Atlas AGORA-C webservice was set up in order to provide the functionality to encode map locations to AGORA-C binary codes and to decode AGORA-C binary codes to map locations based on the Tele Atlas MultiNet Europe data. ASFA test site uses version 2009.02 of Tele Atlas database.

The export module calls the AGORA-C TA webservice for every location stored in the ROSATTE database.

Implementation:
- Load the locations (TA map Ids) from the database and request the TA AGORA-C webservice through a .Net Framework 3.5 C# DLL: *agoraCEncoding.dll*

Example of input sent to the TA webservice:

See example in annex

Example of AGORA-C string returned by the TA webservice:

ATYBMAAyAiAGBBUUxCwED4EgZaVPSIJ7yUAAAQNBNDMEDAvIRAfY/zbNAAABAQQIB8kIAKRLHmA=

4.1.4 Export module / export datasets in ROSATTE XML format

The last operation of the module is the export of datasets in ROSATTE XML format. A data set is export when requesting by a client such as map providers.
ROSATTE XML format is defined by a XSD schema (rosatte.xsd) available on ROSATTE web site.

Implementation:
Load the safety attributes information from the database and generate a dataset through a .Net Framework 3.5 C# DLL: export_rosatte.dll

Example of ROSATTE export:

See example in annex

The export module generates ready to be published datasets.

4.1.5 Data publishing module / REST webservice

ASFA test site safety attributes datasets are published through a REST webservice: *restServicePublishing*

Environment:
- .Net Framework 3.5 WCF RESTfull webservice: *restServicePublishing*
- Microsoft IIS web server

Base url:
/restServicePublishing/restService.svc

Implemented methods:

1. Get the list of available datasets:
   Download/queryDataSets?lastValidDataSetID=

2. Get the list of available datasets since the last one retrieved:
   Download/queryDataSets?lastValidDataSetID=value

3. Get a dataset identified by a dataset ID:
   Download/readDataSet?DataSetID=value

4. Send a feedback:
   feedback/sendFeedback/{datasetId}/{feedbackId}

Example of requests:

Request the last available dataset:
http://dev-paris.autoroutes-trafic.fr/restServicePublishing/restService.svc/Download/queryDataSets?lastValidDataSetID=

Request the list of available datasets since the last valid dataset Id specified in the Url:
http://dev-paris.autoroutes-trafic.fr/restServicePublishing/restService.svc/Download/queryDataSets?lastValidDataSetID=9cde05a7-2a5d-4bc4-aa79-e68c557ebeb5

Request a specific dataset using its Id:
http://dev-paris.autoroutes-trafic.fr/restServicePublishing/restService.svc/Download/readDataSet?dataSetID=9cde05a7-2a5d-4bc4-aa79-e68c557ebeb8

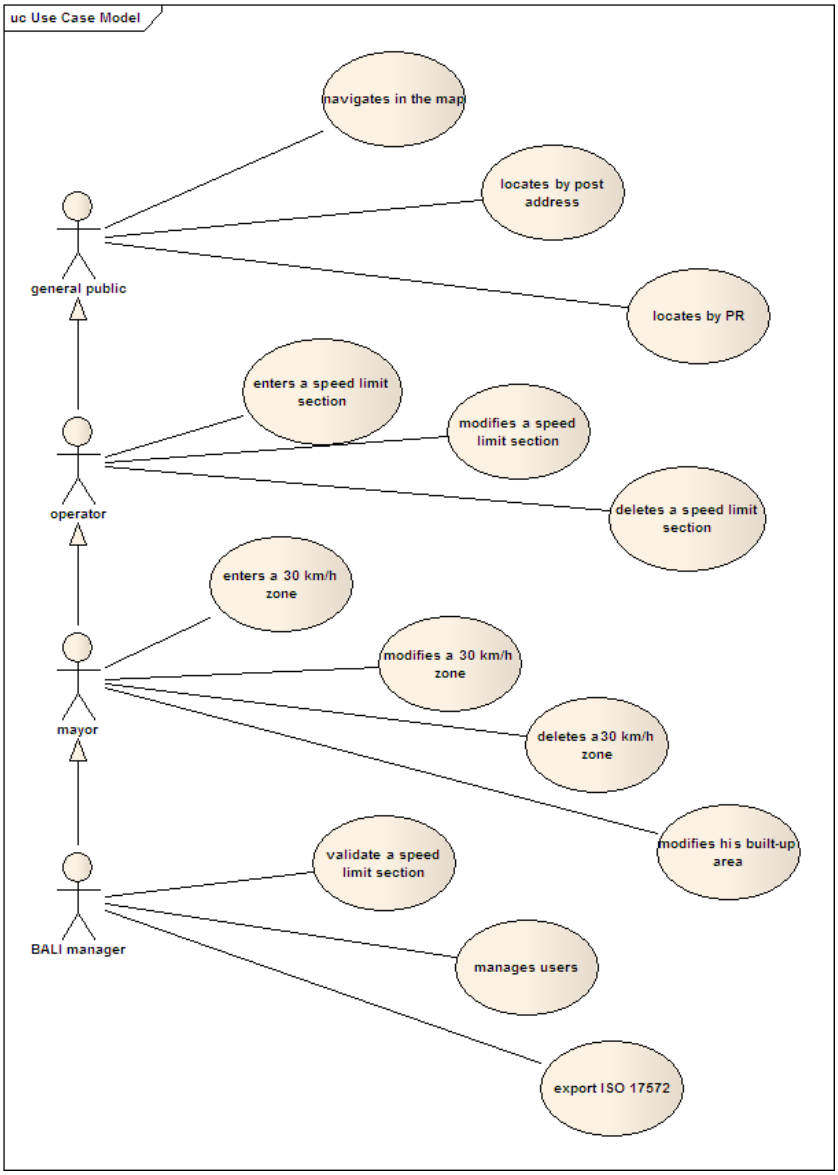
REST webservice always returns data in XML format compliant with the ROSATTE XSD.

4.2 Test site BALI (France)

4.2.1 *Summary of used software/services*

The system (functional architecture) is composed of the following unit:

- a process for initialization processing of a support road database and other background data,
- a process for initialization of the BALI road sign database,
- an internet application for consultation of the speed limit database,
- an extranet application for updating the BALI database,
- an extranet application intended for the validation of speed limit section entries,
- an application for the administration of users,
- an exportation module of data intended for the data producers.



PR : Point of Reference = distance marker

Figure 10 – BALI UML model example

Technical architecture:

- ESRI solutions : ArcGIS Server
- Oracle DB

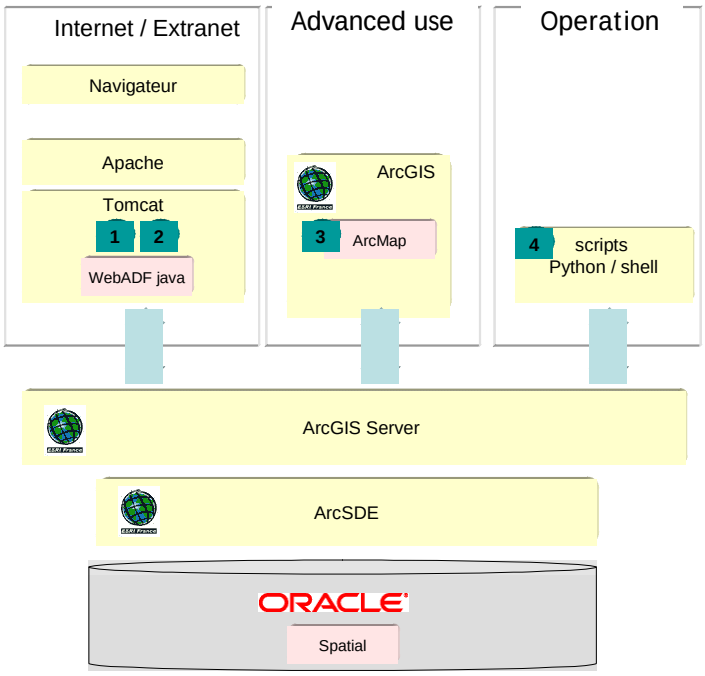


Figure 11 – BALI technical architecture

4.2.2 *Initialization of BALI data base*

From the data support database, the Ministry wishes to initialize a specific speed limit database. This database, which is associated with the road database, is essentially made up as follows:
- road signs (as required),
- maximum speed zone ("zone 30", built-up area, etc.),
-segments constructed on the highway grid indicating the speed limit (linear by dynamic segmentation).

These data do not exist currently, either at the Ministry or with the data suppliers. The highway grids currently incorporated into the vehicle-mounted navigation software contain the speed-limit information, but this information is partial and approximate, to the extent that it is not indispensable to the orientation process.

It is therefore necessary to initialize this data by processing from the support database and creating road signs in accordance with the following elements:
- the classification of the road and the type of road (single carriageway and dual carriageway), which indicates the generic speed limit,
- The indicated speed limit (if it is different from the generic speed limit, the driver having necessarily been warned by a sign),
- the continuity of the graph: it can be assumed that a single sign indicates the speed limit for a set of adjacent sections without crossings, having the same characteristics.

The aim of this function is to use the supporting data to initialize the business data.
For each road link, the application uses its functional classification to check whether the associated speed limit value is different from the generic speed value that would normally be expected.

Allocation rule for generic speeds: the generic speed is deducted from the supporting data (road class)

Motorway:	130 km/h
Quasi-motorway:	110 km/h
Dual carriageway:	110 km/h
Single carriageway:	90 km/h
Slip road:	90 km/h
Unpaved road:	90 km/h

Other values: Segment ignored
And the built-up areas (with speed limit 50 km/h) are updated by importing all the built-up area data in the supporting data.

4.2.3 Extranet application for speed limit section

The local authorities responsible for managing the roads are required to publish a traffic order for each sign change. The objective is to encourage these municipalities to publish the geographical position signs erected with the order.
An extranet application allows a positioning in a user-friendly environment.
The application includes the following features:

- Consultation of a map presenting the highway grid and the speed limits (in the form of a colour scheme)
- A location function using the postal address and by PR (point of reference),
- A function for entry / modification / deletion of a speed limit section,

This function takes account of a geographical restriction that is applied in accordance with the connected user and his jurisdiction.

Tool bar:

Speed section

Figure 12 – Editing tools snapshot

Buttons for the **speed limit section** (accessible only by the local authorities and administrator):

- Creation of a speed limit section: runs a wizard as described above,
- Modification of a speed limit section: a speed limit section must be selected. Runs the creation wizard, with the values associated with the selected section,
- Deletion of a speed limit section: a speed limit section must be selected. Deletes the selected section after a confirmation message.

The following screen illustrates the first stage for the creation of a speed limit section (by cartographic coordinates, reference point, address, and interesting point)

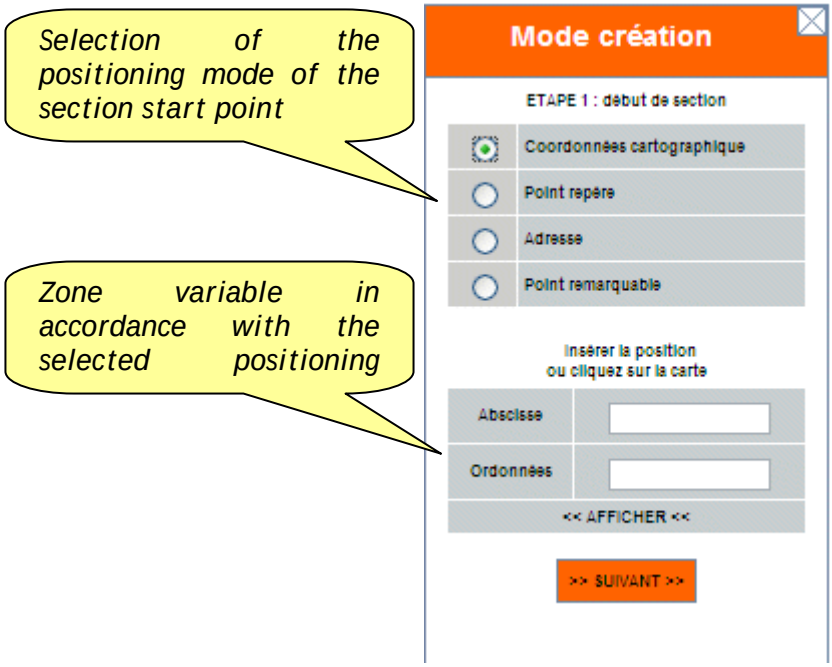


Figure 13 – Creation of speed limit section at BALI

4.2.4 *Extranet application for speed limit area*

The application also includes a function for entry / modification / deletion of a speed-limit zone. This function takes account of a geographical restriction that is applied in accordance with the connected user and his jurisdiction.

Tool bar:

Figure 14 – Editing tools for speed limit area

Buttons for the **zones** (accessible only by the local authorities and administrator):

- Creation of a zone: used to digitize a 30 km/h zone within the built-up area,
- Modification of a zone: used to modify a 30 km/h zone or the built-up area,
- Deletion of a zone: used to delete a 30 km/h zone.

Figure 15 – Built-up area creation

To creation of a 30 km/m zone and a 50 km/h zone involves the following rules:
The creation of a 30 km/h zone is effected within a built-up area.
Only the Maires and the Préfets have authority to create 30 km/h zones.
The Maires are able create 30 km/h zones only within their own built-up areas.
The LROP is responsible for validation of a 30 km/h zone.

The user can create a 30 km/h zone within a built-up area by clicking on .
The cursor changes to allow positioning of the apices of the polygon outlining the 30 km/h zone.
A window opens in parallel, to allow entry of the coordinates of each apex.
The 30 km/h zones cannot have either holes or islands.
When the user double-clicks, the polygon is closed, and the user can enter the attributes.

An apex is added automatically at each intersection of the polygon of the zone with a road.

4.2.5 *Software module for exporting data*

The export of data comprises 2 parts:

- The data themselves and their location
- The metadata.

Data export is inspired by the specifications of the EuroRoads project. The location part is inspired by that of the draft standard ISO DIS 17572.
The XML document resulting from the export will include all the speed limitation sections of the BALI base. This data extraction will be proposed for the whole county. The data are calculated upon request at each export.

The BALI application will be based on a method in order to integrate the XML Agora exportation process. The aim of this export operation is to generate an XML document according to the XML scheme specified in part 3 of the ISO standard, namely "Intelligent Transport System (ITS)—Location Referencing for Geographic databases" (ISO/DIS 17572-3).

The XML document resulting from the export process will include all the speed limit sections of the BALI database. This data extraction will be proposed for the whole county. The data will be calculated on demand at each export event.

Referencing of the location is effected by the creation of an ordered list of significant points that can be used to re-create the location on a transmission network. These points are called "*Core Points*" or *CP.*

In the context of the BALI project, the objects to be referenced in the database are the speed limit sections.
Each speed limit section will therefore be modeled by a sequenced list of these points, obeying the following rules:

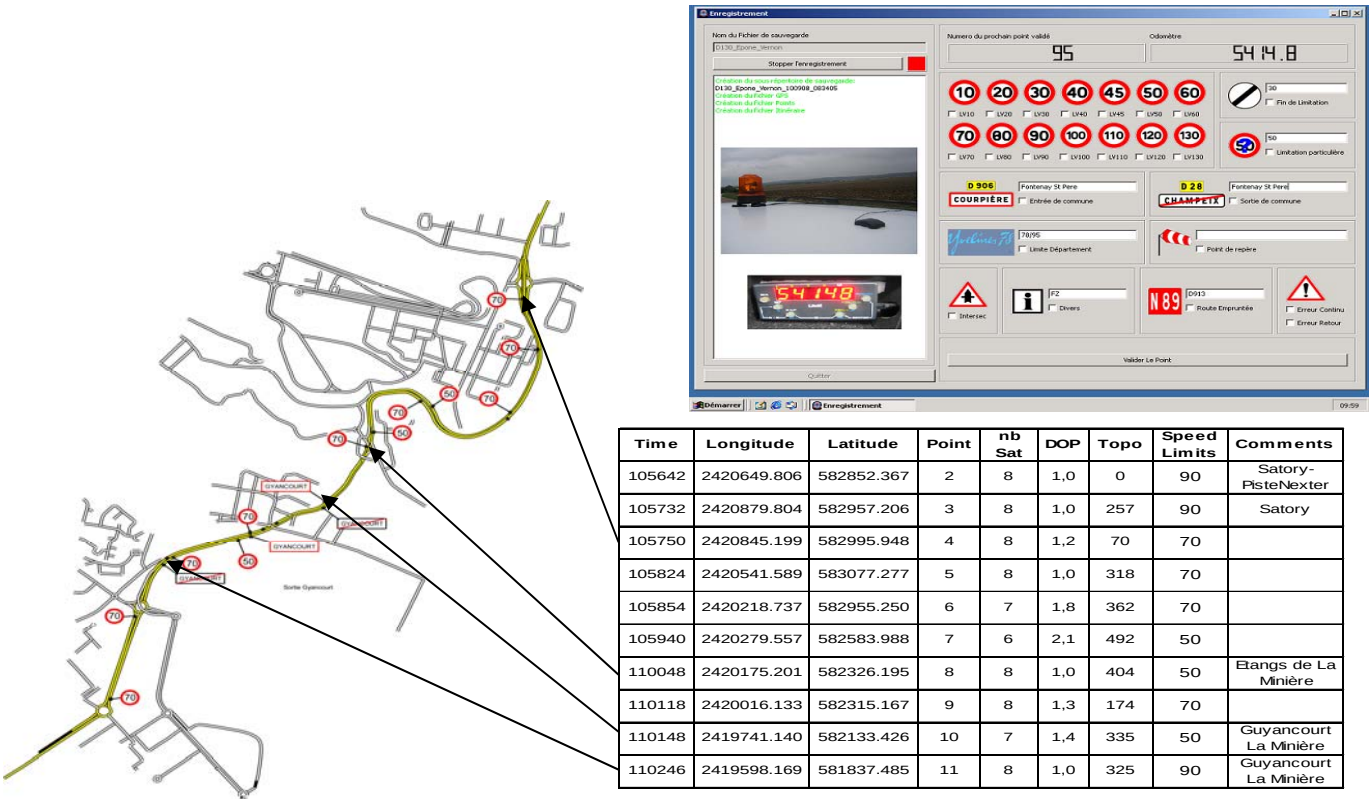
- The list starts with a CP relating to the start of the speed limit section.
- The list ends with a CP relating to the end of the speed limit section.
- Each point on the list is in direct relation to the next point.
- The list necessarily starts with a specific CP named Intersection Point IP (that of the section start). These point represent intersections where the Road Section Signature changes.

BALI service offers a direct access to the BALI database for downloading a full data export in XML format.

No REST services have implemented for the BALI application since it is not complying with French ministry's policy for ICT (only SOAP is accepted). The produced files will be available through a simple http interface.

4.2.6 *Software tool for checking on the field*

A mobile equipment with a simple application is used to collect the geographical position of the signs in the field. This data is used to update the central database ("map-matching" in order to reconstruct the travelled route, linear referencing of the identified signs, propagation of information about the network…).



Time	Longitude	Latitude	Point	nb Sat	DOP	Topo	Speed Limits	Comments
105642	2420649.806	582852.367	2	8	1,0	0	90	Satory-PisteNexter
105732	2420879.804	582957.206	3	8	1,0	257	90	Satory
105750	2420845.199	582995.948	4	8	1,2	70	70	
105824	2420541.589	583077.277	5	8	1,0	318	70	
105854	2420218.737	582955.250	6	7	1,8	362	70	
105940	2420279.557	582583.988	7	6	2,1	492	50	
110048	2420175.201	582326.195	8	8	1,0	404	50	Etangs de La Minière
110118	2420016.133	582315.167	9	8	1,3	174	70	
110148	2419741.140	582133.426	10	7	1,4	335	50	Guyancourt La Minière
110246	2419598.169	581837.485	11	8	1,0	325	90	Guyancourt La Minière

Figure 16 – Software tool for checking on the field

4.3 Test site Bavaria (OBB)

4.3.1 Summary of used software/services

The data maintenance system implemented in TS Bavaria has been described in Deliverable 'D2.2 Implementations of tools for demonstration of data maintenance and access in different test beds' in detail (see Figure 17). From data maintenance (editing operations) all safety features are stored in the ROSATTE SafetyFeature Database. For the data exchange via the ROSATTE interface now several modules are implemented.

- SafetyFeature Referencing Module: It calls a remote service to generate an AGORA reference for the safety feature location and includes this location string in the SF data base. This AGORA Service is provided by NAVTEQ. As an alternative an OpenLR encoding service developed by PTV is implemented, allowing to use and evaluate OpenLR in the context of ROSATTE as map-based location referencing method.
- Snapshot Generator: In regular intervals all currently existing safety feature objects are assigned to a snapshot, representing a current (full) data set of Safety features.
- Diff-Builder: this module generates a data set representing the updates between two snapshots;
- REST Service: It offers online access for downloading safety features as specified by the ROSATTE data exchange specification (D3.1)

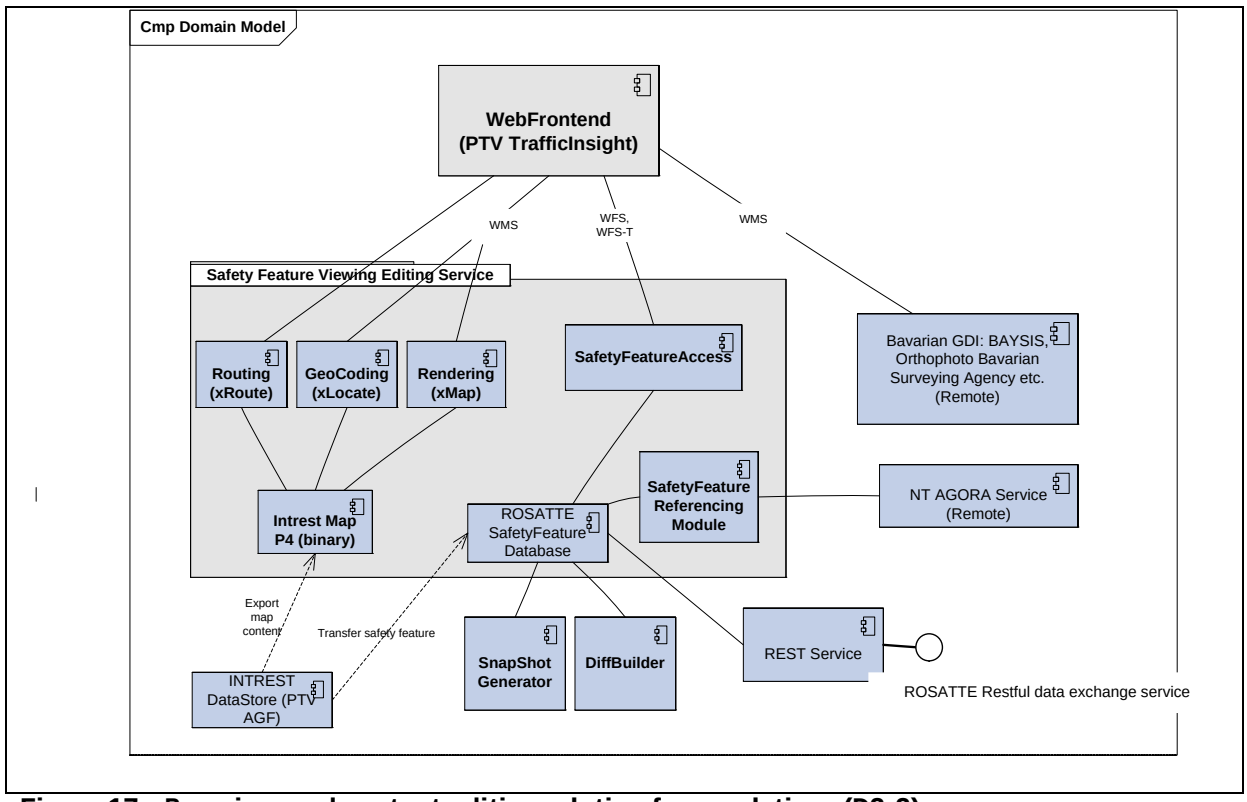


Figure 17 - Bavarian road content editing solution for regulations (D2.2)

In the following, these modules which ensure the data exchange according to ROSATTE are described.

4.3.2 Safety feature referencing service

In the ROSATTE database, safety features hold different referencing information, namely their geometry (GML linestring) and a logical reference to the road network (Permanent Link Ids to the INTREST network). For transmission via ROSATTE, AGORA referencing is required. For this purpose, the remote AGORA service provided by NAVTEQ is used.

In order to decouple the AGORA coding from the data maintenance by the user (e.g. if service is unavailable or generates errors) the following approach is used.

A time-scheduled process (cron job, once a day, at 23:30 hours) is searching the database for safety features without AGORA references. For these safety features, the internal logical reference (Permanent link Ids from the INTREST network) is then transformed into the link references to the corresponding NAVTEQ network by a matching/look-up table in the database. If the matching from INTREST network[4] to NAVTEQ network fails, an error code (including the "bad" INTREST network ids) is inserted as string in the AGORA attribute for later manual debugging. In an alternative system setup, an OpenLR encoder is used, which uses the INTREST network directly for encoding (hence, no matching/look-up table is needed).For these NAVTEQ link references, the remote NAVTEQ-AGORA service is called via http to generate codes. Following cases can be discerned:

- If a valid map-based location reference is returned it is inserted in DB as attribute to the respective safety feature.
- If the remote map-based location referencing service is unavailable, no code is inserted, i.e. AGORA encoding will be requested the next time the job runs.
- In case the remote map-based location referencing service generates errors then these error codes are inserted as string in the AGORA attribute.
- Where the safety feature references no linear features but 'network zones' a specific error ("ERROR:_...") is inserted instead of the AGORA location string. Such zones references can currently not be transmitted via the ROSATTE interface, since they cannot be described in one single map-based location referencing code[5].

The following error code texts are inserted in the AGORA string attribute in case of problems:

- ERROR: Zones are currently not supported.
- ERROR: Start and/or end offset of the segment are invalid.
- ERROR: Mapping from INTREST id xxx to NAVTEQ id failed.
- ERROR: Parsing of INTREST id xxx from ptv_geometry of simplified_safety_feature with id yyy failed.
- ERROR: Encoding failed: (Wrong Parameter), EncodeRequest=xxx, Exception=yyy

These error codes are self-explanatory and allow a visual examination in the ROSATTE xml data container and to understand the problems encountered when map-based location referencing encoding fails.
Currently no way is foreseen to inform the data maintenance operator on map-based location referencing errors, since the map-based location referencing coding has only

[4] A summary of the INTREST system can be found in ROSATTE Deliverable D1.1 State of the art, Version 1.0, 1. September 2008. In Chapter 4.10.2.5,

[5] A change request to the ROSATTE data exchange specification is pending, where a safety feature shall be allowed to hold 0…n (linear) AGORA references, instead of currently 0..1. A zone would then be described by a set of linear extents, for which AGORA references can be generated by the available encoders.

relevance for the transfer to third parties, not to his own operations. All safety features with a map-based location referencing string attribute that is not empty will be transmitted to the map makers.
Obviously, only those safety features that hold valid map-based location referencing references (and not error codes) as content to the map-based location referencing string attribute may be ultimately decoded and integrated in the target database on the map maker. Hence, the information on the safety features with encoding errors as represented in the map-based location referencing string may be fed back by Map providers to the data maintenance side through the feedback service.

4.3.3 Snapshot generator

The ROSATTE database in the Bavarian data maintenance system uses the concept of snapshots, mainly to allow the generation of updates from differences between two subsequent snapshots.
Safety features are held in the database pertaining to snapshots which are generated in regular intervals (cron job e.g. once a day). A snapshot represents all safety features with non empty AGORA references assigned to at the moment of the snapshot freeze. In practice these snapshots correspond to a tagging of a safety feature to a snapshot.

A second intermediate representation of the safety features holds the current state of the safety features. The editing operations from the data maintenance frontend interact directly with this representation of the safety features.

- If a safety feature is added to the ROSATTE data store, it is inserted as new object in this intermediate representation.
- If a safety feature is modified, its last state of the SF is represented in the intermediate representation
- If a safety feature is deleted by the operator it is deleted in the intermediate representation.

With the generation of a new snapshot, the intermediate representation is blocked for further editing. The objects from the intermediate representation are transferred to the main database representation and tagged as pertaining to the latest snapshot:

- New safety features are inserted as new objects in the main database and tagged as part of the new snapshot.
- Modified safety features are inserted in their latest, modified state as new (additional) objects in the database and tagged as part of the new snapshot
- Deleted safety features are of course not transferred to the main database. But their last representation in the main database which pertains to older snapshots remains unchanged in the database.
- Unchanged objects remain unchanged in the main database except that they are also tagged as pertaining to the new snapshot. They then may hold references/tags to 2 or more (previous, subsequent) snapshots.

The concept used here between the intermediate representation of safety features (for editing operations) and the main database resembles the one known from data synchronization processes such as rsync used in IT networks.

After this process, the intermediate representation is initialized with the content of the latest snapshot and accessible again for data maintenance.

4.3.4 *Diff-Builder*

This component generates differences between two subsequent snapshots from ID and attribute comparison. It thereby identifies new/modify/delete updates as required by the ROSATTE interface.

4.3.5 *REST service:*

This component implements the RESTful data exchange service as required by the ROSATTE data exchange specification to allow online access to safety feature data sets, either as full representation or as update.

- In case a full representation is requested, the latest snapshot of safety features is made available in the ROSATTE xml data container. (No possibility is offered to receive older snapshots).
- In case updates are requested, the REST service supplies access to 0..n data sets which hold the changes from one snapshot to the next snapshot, dating back to the last change data set that the requesting client has received.

Internally, the Java-objects which hold the snapshot or update data sets automatically generate the xml data containers for download in the REST service (JAXB, JAXRS, see Figure 18)

For the Bavarian test site, changes can therefore be only received in sets between fixed snapshots and not between arbitrary states. The recipient needs to process these sets in their original order, otherwise the resulting data set may be inconsistent.

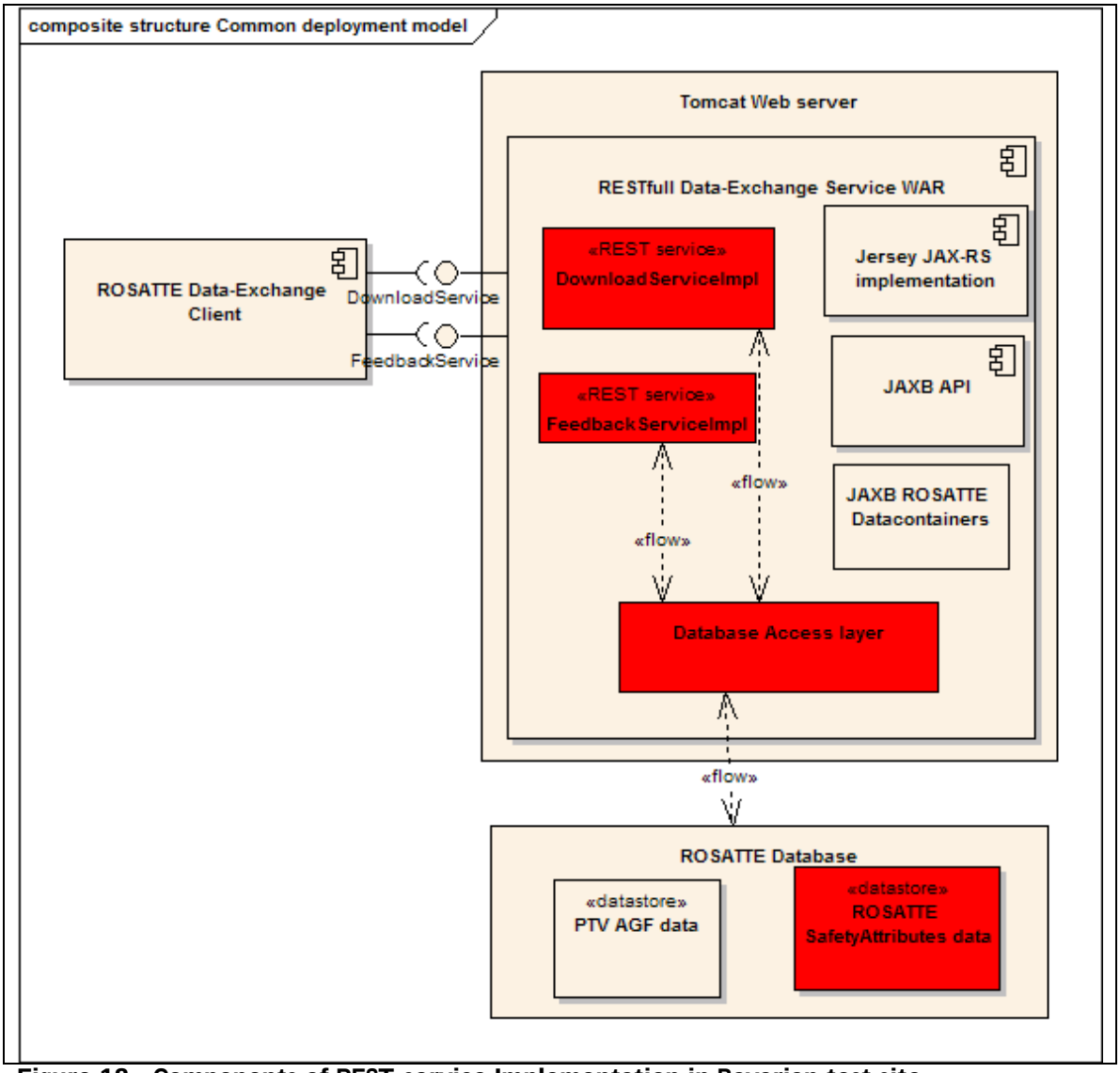


Figure 18 - Components of REST-service Implementation in Bavarian test site

ROSATTE REST for TS Bavaria which uses AGORA as on the fly referencing is accessible under http://80.146.239.164/data-exchange-service. The REST service of the mirror system which uses OpenLR as on-fly referencing encoding is accessible under http://80.146.239.153/data-exchange-service

Examples are:

Request list of all existing (update) Datasets for the last dataset with identifier(5,1281910205630):	http://80.146.239.164/data-exchange-service/download/queryDataSets?lastValidDataSetID='5,1281910205630
Request list of all existed update Datasets	http://80.146.239.164//data-exchange-service/download/queryDataSets
Request data of the Update-Dataset(5,1281910205630):	http://80.146.239.164/data-exchange-service/download/readDataSet?dataSetID='5,1281910205630'
Request list all existed feedbacks for the update dataset (5,1281910205630):	http://80.146.239.164/data-exchange-service/feedback/queryFeedbacks?dataSetID='5,1281910205630'

4.4 Test site Flanders (FL)

4.4.1 Summary of used software/services

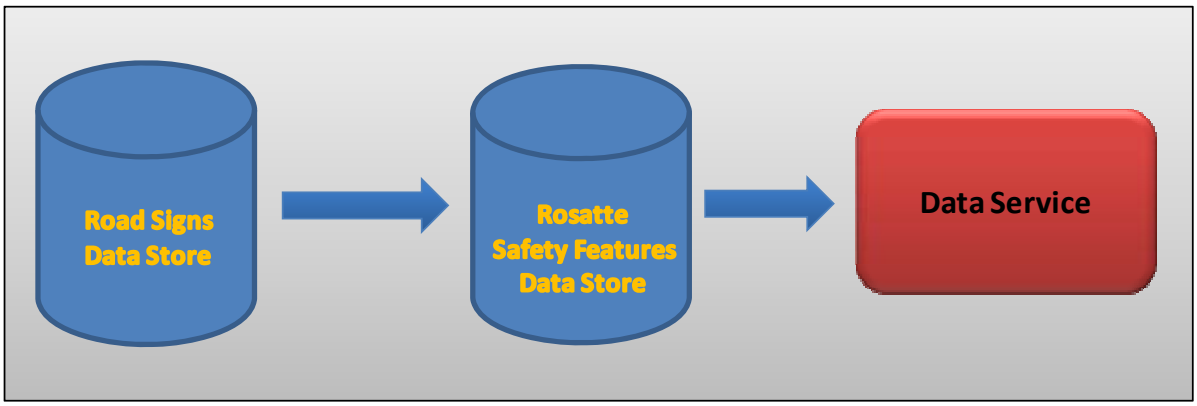


Figure 19 - High level architecture for the Flanders test site

- Road Signs Data Store: contains the locations and attributes of all road signs in Flanders (highways, regional and municipal roads). This data store is an existing data store (not part of the ROSATTE project) which is due to be completed at the end of August 2010. It serves as the basis for extracting ROSATTE Safety Features, both snapshots and updates as the data store evolves over time.
- ROSATTE Safety Features Data Store: Safety Features extracted from the Road Signs Data Store are stored here in xml format according to the ROSATTE specification. These include snapshots and updates.
- Data Service: extracts information from the ROSATTE Safety Features Data Store and delivers it to the data/information providers.

Safety Features are extracted from the position and attributes of road signs. Two major parts can be distinguished in this extraction process:

- Safety Feature attribute content: from the attributes of a particular road sign or a combination of road signs, a particular Safety Feature is extracted – if applicable since not all road sign / combination of road signs have Safety Feature content.
  A xml-file which contains mappings from the code-attribute of a road sign / road signs combination to a particular Safety Feature, is used to extract Safety Features.
  A Safety Feature will be extracted for a road sign / road signs combination if its code-attribute is found in this xml-file.
  All Safety Features described in the ROSATTE specification are targeted this way.
- Safety Feature Location Reference : the geographic location of the road sign / road signs combination in the Road Signs Data Store is a X,Y coordinate and a road axis-id to which the road sign / road signs combination is referenced. The X,Y coordinate is projected on this road axis and then AGORA-C encoded to obtain a location reference. This implies that all Safety Feature Location References are point location references.

4.4.2 *Software modules/service*

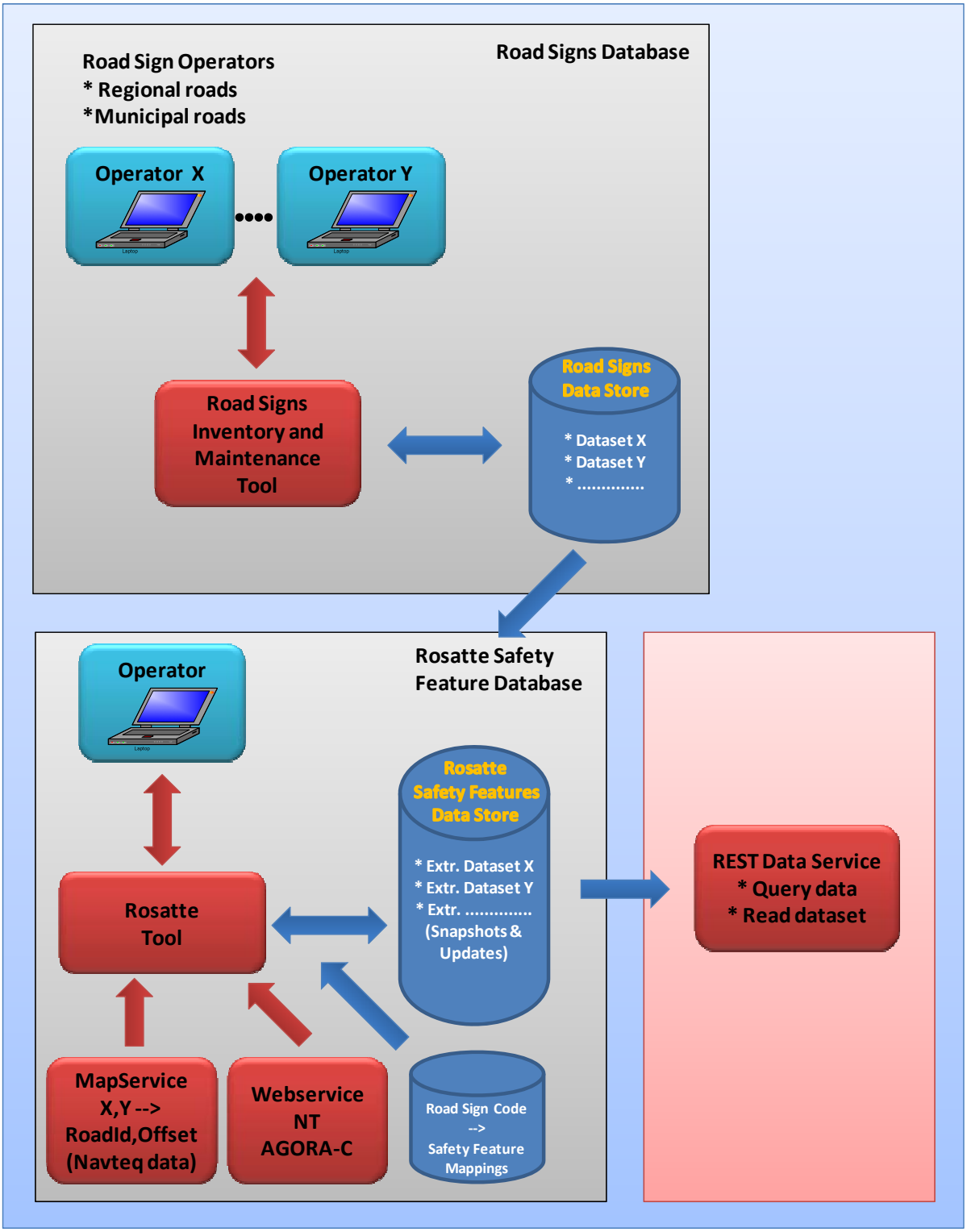


Figure 20 - Overview of software modules and services

4.4.2.1 Road signs database (not part of the ROSATTE project)

The road signs database is maintained by different operators each responsible for a certain area and either regional roads or municipal roads. All road signs are stored in a central database subdivided in datasets that represent this area and road type distinction.
The Road Signs Inventory / Maintenance tool allows the operator to import an inventory of road signs and to maintain this inventory over time.
Although the Road Signs Database itself is not part of the ROSATTE project, the up to dateness and correctness of ROSATTE Safety Feature Database is directly coupled with the up to dateness and correctness of this Road Signs Database. This Road Signs Database is the basis from which the ROSATTE safety features will be extracted.

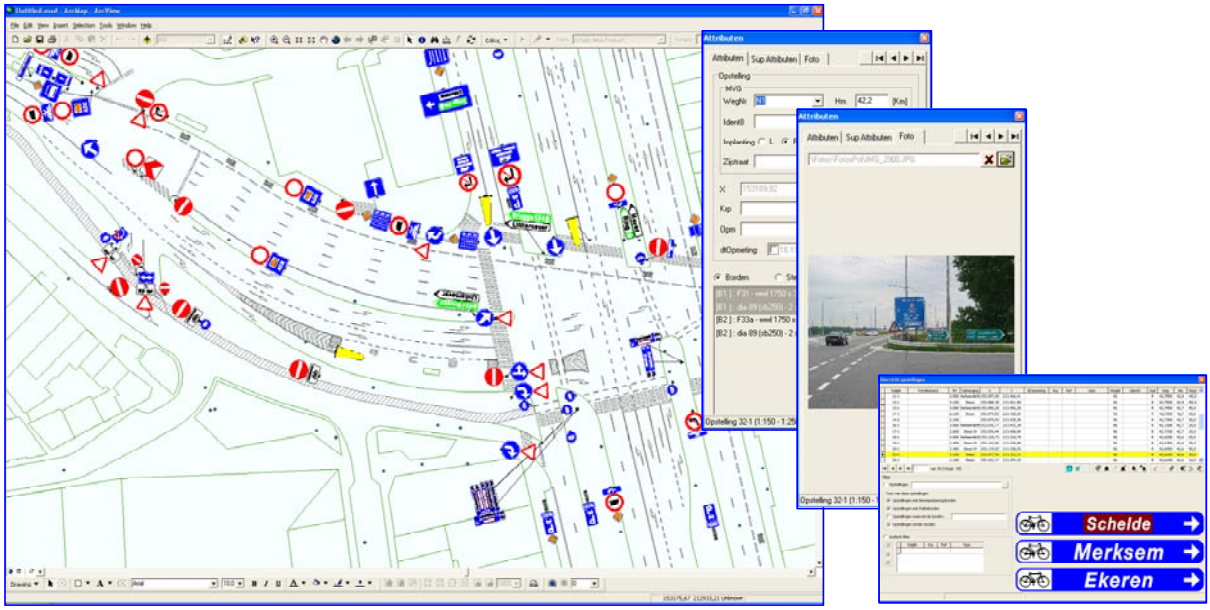


Figure 21 - Snapshot of the road signs database

The central database is Oracle or SQL Server-based. The operators use a client to maintain the database. The client is either GIS-based (ArcGIS) with a custom extension for intranet-based operators, or it is a custom written windows based client based on ArcGIS server technology and served to the operators over the internet using Citrix technology.

4.4.2.2 ROSATTE safety feature database

The ROSATTE Safety Feature Database is maintained by a single operator with the use of the ROSATTE tool. Following components can be distinguished:

- The database itself is Oracle or SQL Server-based. It contains snapshots and updates in xml-format according to the ROSATTE specification. Snapshots and updates are available for each dataset as it exists in the Road Signs Database. So the safety feature datasets are based of the same subdivision into areas and regional/municipal roads as in the Road Signs Database.
- The database is maintained using the ROSATTE tool :
    - This is a windows-based client that connects to both the Road Signs Database and the ROSATTE Safety Features Database.

- It uses an xml file – containing a mapping of road signs / road signs combinations to a corresponding safety feature – for extracting Safety Features attributes from the Road Signs Database.
- Using a ArcGIS-MapService and NAVTEQ data, the X,Y location attribute information of a road sign is translated to a road-id / offset value pair. From this, the location can be AGORA-C encoded using the NAVTEQ AGORA-C webservice. Note that this way, location referencing for a Safety Feature is point-based.
- The tool writes snapshot or update information to the ROSATTE Safety Features Database on a per dataset basis.

An operator initially selects a number of datasets in the Road Signs Database and generates a snapshot Safety Feature dataset for each.
Changes in the datasets of the Road Signs Database do not automatically result in a update being generated for the corresponding Safety Feature Dataset. It is up to the operator to generate from time to time updates for each of the datasets for which initial snapshots were generated. The results are client defined update intervals.

Screenshots:

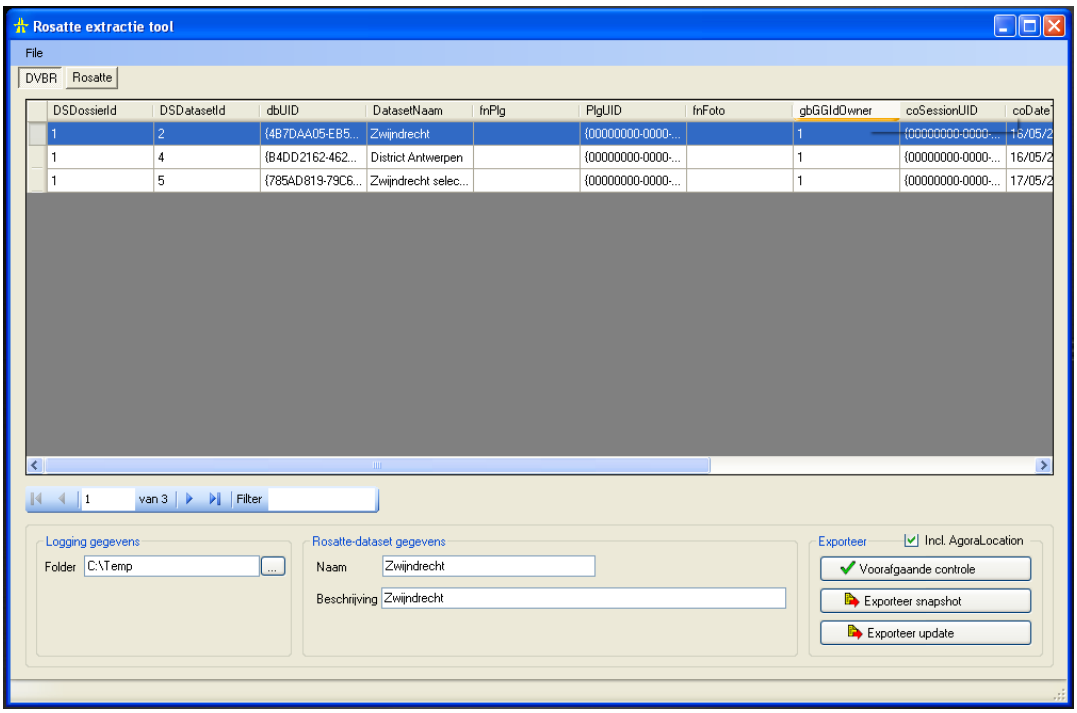


Figure 22 - UI overview of road signs database and functions for extracting safety feature

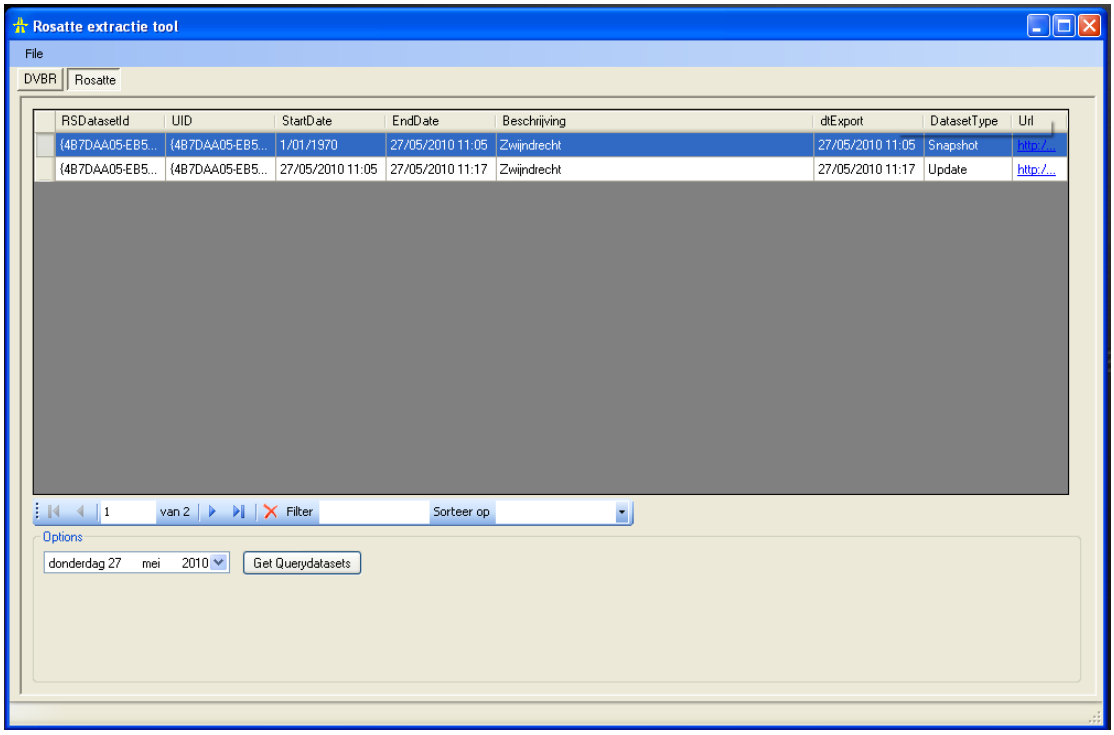


Figure 23 - UI showing overview of safety feature datasets and administrative functions

4.4.2.3 Data service

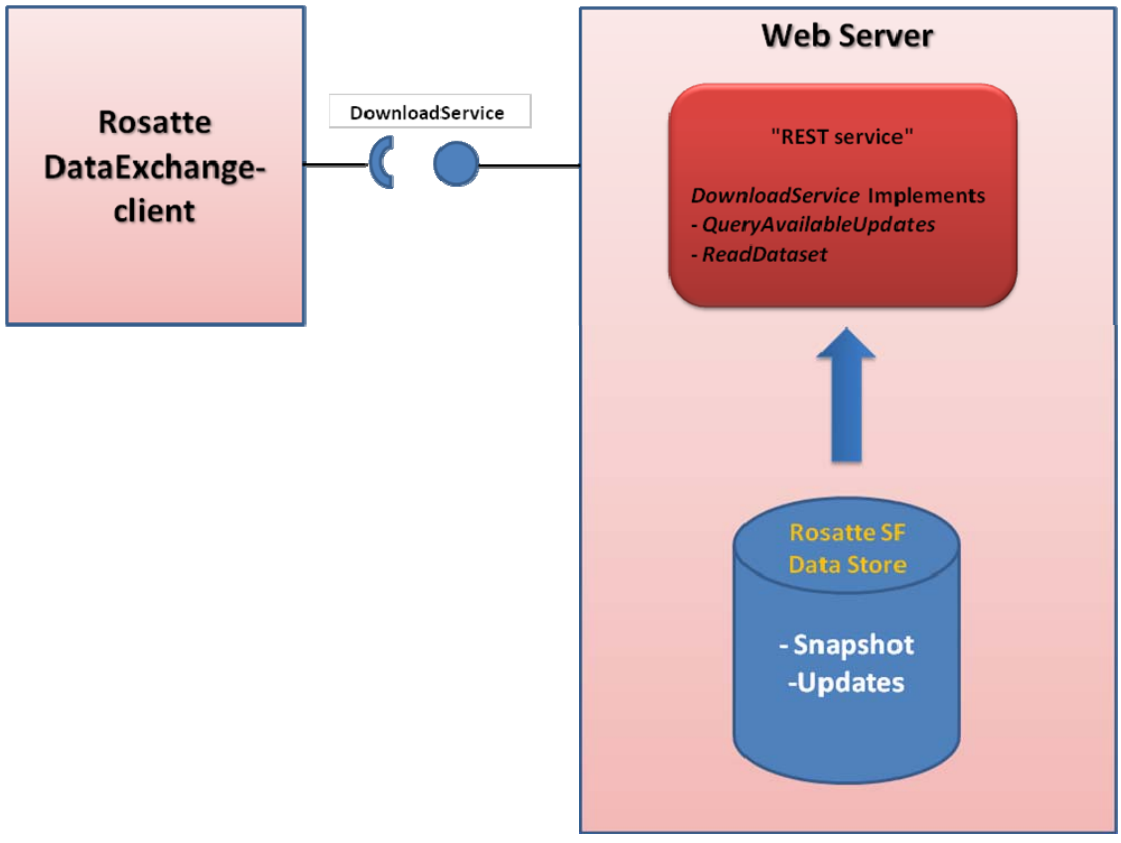


Figure 24 – REST service implementation in the Flanders test site

For downloading ROSATTE Safety Feature Datasets, a REST service is available for a ROSATTE DataExchange-client implementation. The REST service implements the IDownloadService interface only. It supports the methods *QueryAvailableUpdates* and *ReadDataset*.

The *QueryAvailableUpdates* method returns a *RosatteRestDatasetRefList* containing a list of ROSATTE Safeture Feature Datasets (both snaphot and updates).

From this list, the client chooses a dataset which can be subsequently be downloaded from the web server using REST and the URI supplied in the list.

4.5 Test site Sweden/Norway (SRA, NPRA)

4.5.1 Summary of used software/services

This description of the Swedish and Norwegian test sites starts where the ROSATTE data store has been created. A corresponding description on how this is done is found in the deliverable D2.2.

The ROSATTE data store contains all necessary data about safety features, including an update log that chronologically and per safety feature describes the update events (INSERT, UPDATE, DELETE) that occurred. Also each safety feature already contains the encoded AGORA location as an attribute. The data store uses a generic structure approach for safety features which makes the implementation type independent and therefore easily extendable.

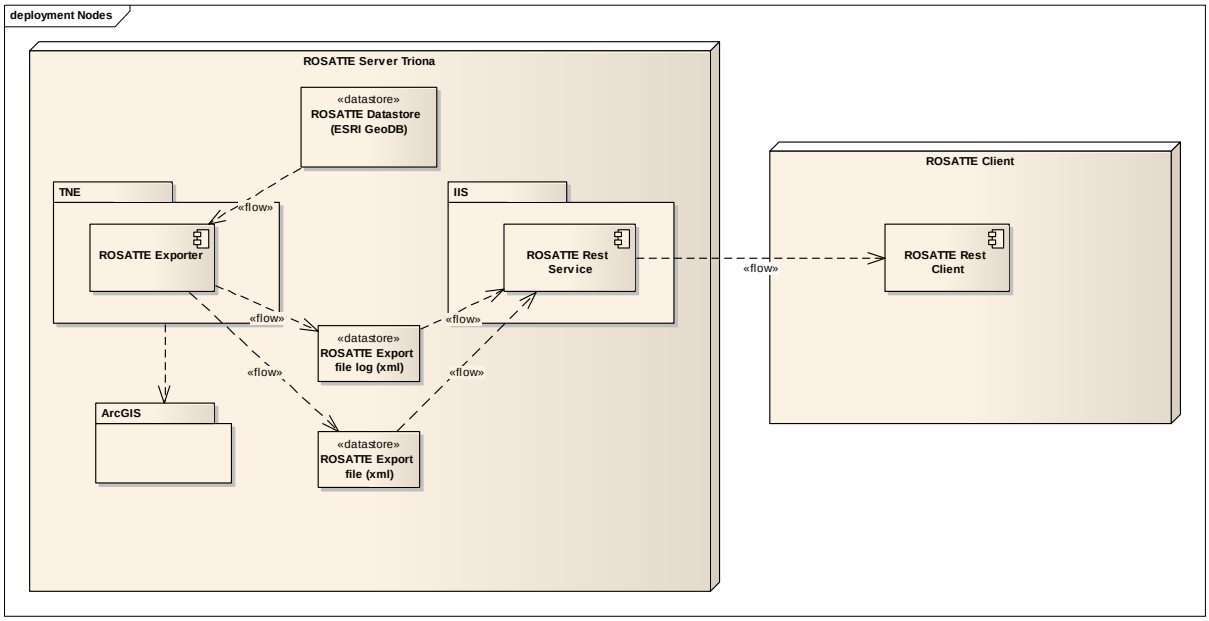


Figure 25 - Component overview ROSATTE REST service SE/NO

4.5.2 ROSATTE server

The ROSATTE server is a Windows machine (Windows XP or 7) with TNE® (*Transport Network Engine*®) and ArcGIS software. ArcSDE or Access is used for the ROSATTE data store. IIS is used for serving the REST service.

4.5.3 ROSATTE exporter

The ROSATTE Exporter is implemented as a command line executable (using C#/.NET). The executable is scheduled to execute automatically with a certain frequency. The purpose of the component is to produce a new ROSATTE xml update dataset for each execution. The executable itself makes sure that the updates in each new update dataset starts where the previous dataset ended.

All available types of safety features are exported at the same time (there is currently no option to select subsets of the data in the store). A list of the currently available safety feature types is described in D2.2.

The ROSATTE Exporter also maintains a log of all datasets that have been exported. This log is later used by the REST service in order to respond to the *QueryDataSets* request from a client.

The executable also has an option to produce an initial dataset. This dataset contains all safety features available in the ROSATTE data store at the time of execution and each safety feature is exported as "Add" updates.

4.5.4 ROSATTE export file/ROSATTE export file log

The ROSATTE Export file is an xml document according to the ROSATTE Exchange format specification in D3.1 (*ROSATTE.xsd*). Each file contains the updates that occurred since the last export.

The ROSATTE Export file log is an xml file that for contains a sequential list of all exported datasets according to the listing below.

ROSATTE Export file log example:
See example in annex

4.5.5 ROSATTE REST service

The REST Service has been implemented using ASP.NET and C#. No framework support for implementing REST Services from wadl definitions (*Data-exchange.wadl.xml*) has been used. Instead, a simple *IHttpHandler* was implemented for all server requests to *http://rosatte-no.triona.se/ROSATTEDownload/download/...*

The request *http://rosatte-no.triona.se/ROSATTEDownload/download/querydatasets* triggers the service to read the ROSATTE Export file log (above) and produce an xml output according to the ROSATTE schema (*ROSATTE-rest.xsd*). A request qualified with a lastValidDataSetID parameter produces an xml output starting with the dataset next in succession from the specified dataset.

The request to download a certain dataset simply returns the content of that dataset as the response.

4.5.6 ArcGIS/TNE

TNE® and ArcGIS are used as basic components for data manipulation in the Swedish and Norwegian solution for ROSATTE.

ArcGIS Desktop standard software may be used to view the content of the ROSATTE data store. Also shape files containing the corresponding data as the ROSATTE export files may be created (manually) in ArcMap. These shape files are used in the validation process. Data is stored using either ArcSDE or Access (for personal geodatabases). With the amount of data used in the Swedish and Norwegian test sites, a personal geodatabase has proven to be sufficient and also practical from a testing point-of-view. If the ROSATTE data store is going to be used in a WMS context, the data may easily be migrated into an ArcSDE and/or PostGRE/PostGIS or any database engine with OGC-/ISO-compliant spatial support.

TNE® is an extension for ArcGIS, created by Triona, for managing Transport Network data.

4.5.7 AGORA encoding

The encoding of AGORA locations actually occurs in the process where safety features are created and stored in the ROSATTE data store (which has been described in D2.2).

However, since it is a fundamental part of the solution we describe the process of AGORA encoding also here.

Both the Swedish and the Norwegian national road databases depend heavily on dynamic segmentation, where linear referencing is used to describe the location (in relation to the road network) for various data elements. All the different types of data, such as speed limits, turn restrictions, height restrictions, road numbers and names, functional road class etc are stored separately. To be able to encode an AGORA location, the network together with the attribution used in AGORA is dynamically segmented into a network where homogeneous (i.e. each attribute has a single value) segments are created.

Each data element of interest for ROSATTE (e.g. speed limit, traffic sign etc) has a location in the form of a linear reference relative to the reference network. This location can, by mapping it to the dynamically segmented network, be transformed into an AGORA location.

The process of dynamic segmentation and generation of location points for AGORA is briefly described in the figure below:

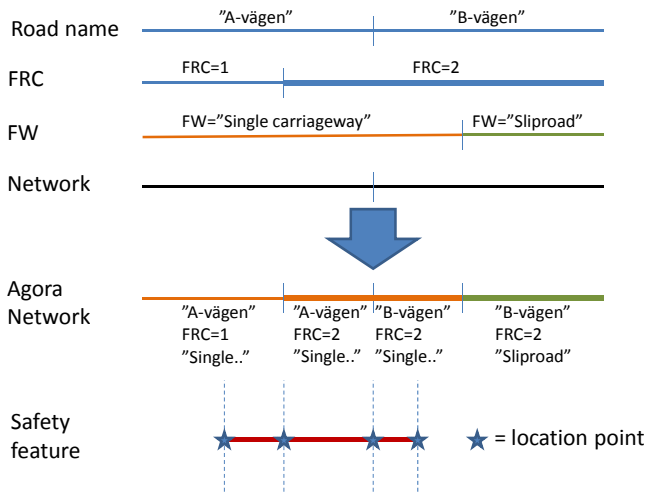


Figure 26 - Dynamic segmentation

The reference network and the various types of attribution (Road name, FRC and FRW in the example above) are segmented independently in the source data store. Each attribution type uses linear referencing mechanisms to record its location with regards to the reference network. The dynamic segmentation process creates a new network which is segmented in such a way that each resulting segment becomes homogeneous with regards to all participating attributes. In order to create the "AGORA Network", a dynamic segmentation process which collects all attributes of interest for AGORA is set up and executed. This AGORA Network can later be used in the process of encoding AGORA locations where any linear reference (with regards to the reference network) can be mapped to a location in the AGORA Network.

To be able to reuse as much code and logic as possible, a plug-in architecture was created for the Swedish and Norwegian test sites where the encoding engine is a component which is used for both test sites, but the specifics for each test site is implemented in a plug-in which is unique for each test site. The encoder that shall be instantiated is specified in a configuration file for the executable. The principle is explained in the figure below:

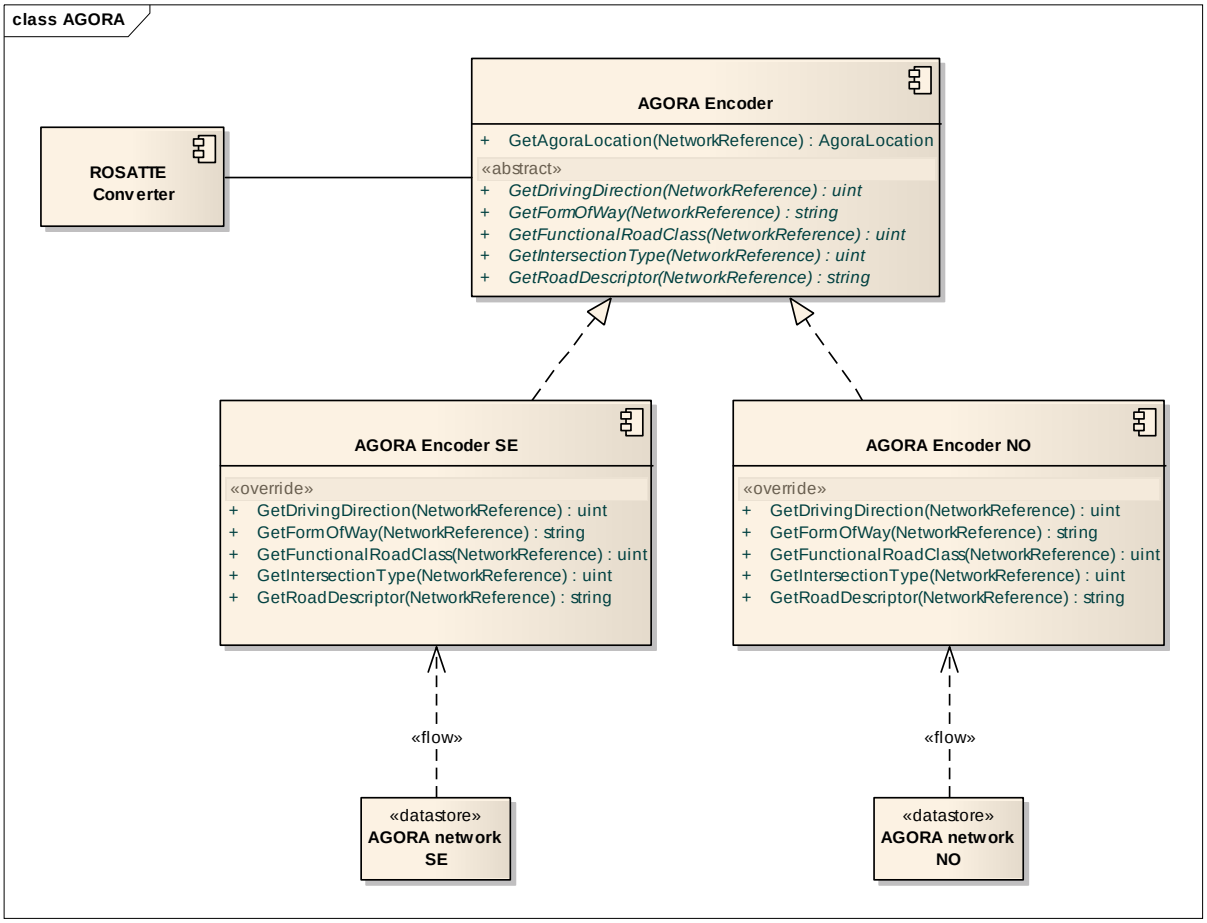


Figure 27 - AGORA components

4.6 Test site London (TFL)

4.6.1 Summary of used software/services

Introduction

The DSLM software is a set of MapBasic applications developed to enable both the on-site surveying and recording, and the in-office update and maintenance of speed limit signs. The applications allow for automatically colour-coding the roads covered by the signs in order to produce the comprehensive speed limit map (DSLM) of the London area.

A full survey of the area has already been undertaken, the speed limit signs plotted onto the map and the relevant colour-coding has been applied. The London Boroughs regularly make changes to the speed limits on their roads through the publication of Traffic Management Orders (TMOs) and in doing so introduce new, alter or remove existing speed limit signs. These changes need to be reflected on the speed limit map and so on-site surveys are carried out periodically and any alterations recorded and transferred onto the master system where speed limits applicable to each length of road are updated automatically. A full audit trail of all alterations made to the system is kept in order to track and monitor the changes.

The application

The application has two specific functions and they are referred to as the 'Server Application' and the 'Laptop Application'. The server application comprises the entire road link network with the terminal and repeater signs for the Greater London area overlaid onto Ordnance Survey Aerial Photography data, all stored and managed on an Oracle Spatial database (migration to the Oracle database has been stalled by budget cuts. It is expected to have this done shortly but in the meantime the software is the same, the data is just managed through a MapBasic Application). This is treated as the master copy where all changes to speed limit data are made, whether directly onto the system or through automatic updates from collected survey data.

The laptop application is loaded onto one or more laptop or tablet computers and typically holds a subset of the data. They are taken out on site where the TMO and signing information is checked for accuracy. Any changes made on site can be subsequently loaded into the server system which is then automatically updated.

A number of tools have been provided to allow the user to position new flags indicating an area where a TMO speed limit change has been introduced, position new terminal signs and repeater signs, reposition any of these signs or remove them from the database according to the information held within the associated TMO and add in field notes. Changes can also be made to the other attributes of the sign, such as the speed limit it is enforcing, the sign condition and its status as either a physical sign location on street that is present or absent, or a sign required by the system to denote a possible change of speed limit due to change of road type – a "fix-it" sign.

Terminal signs hold two sets of speed limit values, Primary and Secondary. The primary speed limit indicates the speed limit which will apply from the sign. The secondary indicates the speed limit up to the point at which the sign is positioned.

Terminal signs are positioned on a road link at the point where a change of speed limit occurs and so the primary and secondary values are usually different. These values are then used to assign the speed limits to the road link that they are situated on; the road link also being colour-coded according to the speed limit. An example of the terminal sign symbol is set out in table 2 below.

Repeater signs are positioned along road links at regular intervals in order to re-enforce the speed limits. Their primary and secondary speed limit values are normally the same. They are not used to colour-code the road links. An example of the Repeater Sign symbol is set out in table 2 below.

Field notes are positioned in locations where there has been a difficulty to survey due to road or building works, or where the situation of the signs on street creates a problem with the colour coding. A photograph can be attached to the field note to give the user a clearer idea of the situation on street. An example of the Field Note symbol is set out in table 2 below

Name	Symbol
Terminal Sign	
Repeater Sign	
Field Note	

Table 2 - Sign and field note symbols

4.6.2 *Software module/service*

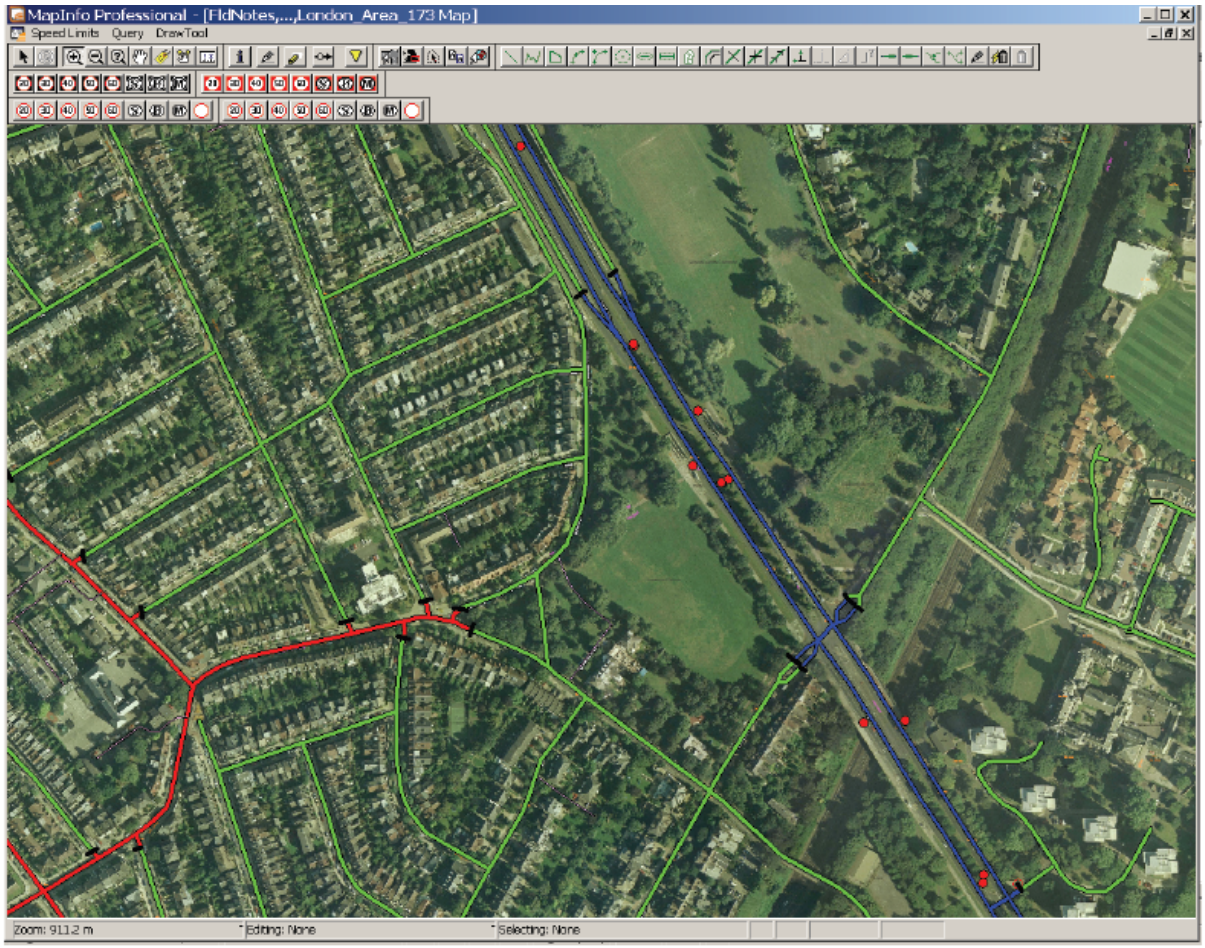
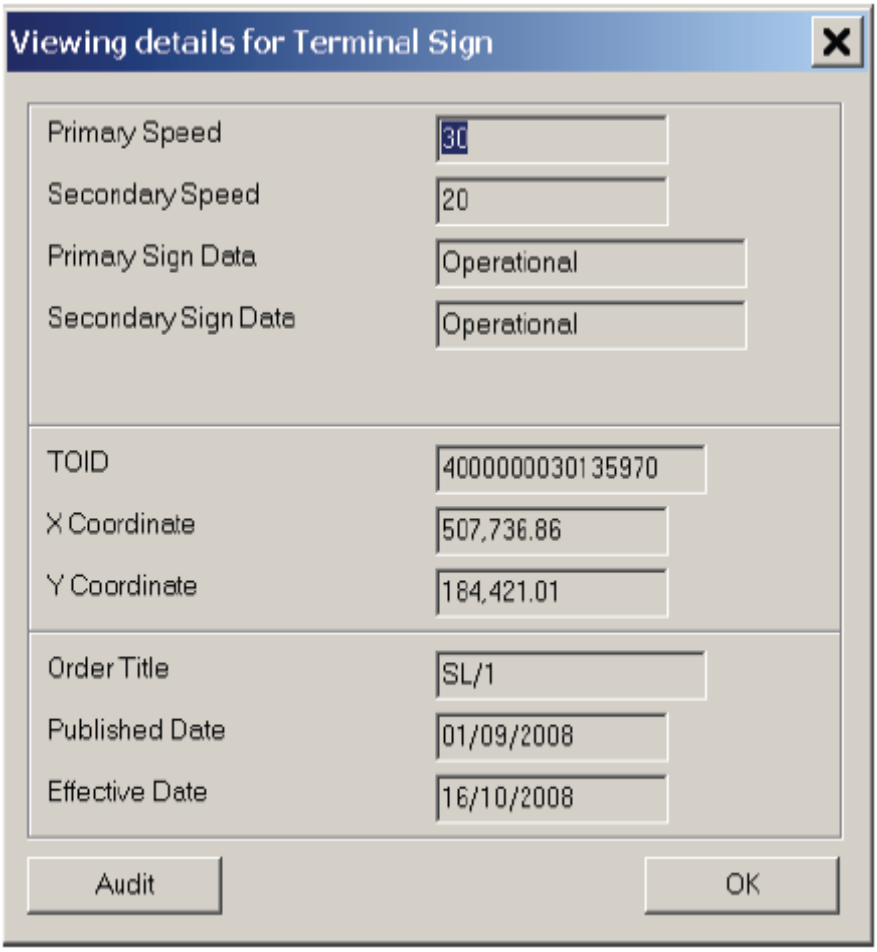


Figure 28 - Opening screen for server/laptop application

On starting the application you will first be prompted to enter your name (used for auditing purposes) and then be presented with a map displaying the roads, signs and a number of menu and tool button options across the top of the screen. The map will be centred on the same position and at the same scale as when the system was last used.

Information about a particular sign can be shown by selecting the Sign Information ('Info') button and clicking on the required symbol. 
For a terminal (or repeater) sign the following screen will be shown:



The fields within the Sign Information Screen and the information provided are:

- **Primary speed**: The speed applied to the road from the sign location
- **Secondary speed**: The speed applied to the road up to the point of the sign location
- **Primary sign data**: The condition/sign status of the primary sign face
- **Secondary sign data**: The condition/sign status of the secondary sign face
- **TOID**: The Ordnance Survey Topographical Identifier value of the road link that the sign relates to
- **X/Y coordinates**: The geographical location of the sign
- **Order Title**: The name of the TMO to which the speed limit is referenced
- **Published date**: The date that the TMO was published
- **Effective date**: The date from which the TMO (and therefore the speed limit) is effective

Figure 29 - Sign information screen

Terminal signs may be added to the system by selecting the Terminal Sign button according to its Primary speed limit value. These buttons are represented by the appropriate speed limit with a *black* background as shown below.

The last three symbols, S, D and M refer to National Speed Limits and refer specifically to National limits on:

- S = Single Carriageway
- D = Dual Carriageway
- M = Motorway

Repeater sign button are represented by the appropriate speed limit with a *red* background.

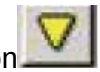
Figure 30 - Terminal sign buttons

Figure 31 - Repeater sign buttons

When a new sign is added, where it bisects the TOID (road link) it will split it and populate either side according to the Primary and Secondary sign values. See below.

Figure 32 - Ingleton Street before (white) / after (red) addition of 20mph sign (red)

Figure 33 - New field note screen

Having chosen the field note button  , the operator will then click on the location in the map he wishes to append the field note to.

This screen (Figure 32), allows you to attach a photo and enter a description for the reason of the field note. Field notes can be edited, moved or removed in the same way as terminal and repeater signs.

To plot a Traffic Order (TMO) point on the map, select the Plot Order button  and click on the map in the vicinity of the speed limit change as advertised in the order. The screen below will be displayed and the operator should fill in as much information as possible that would assist the surveyor in checking signs

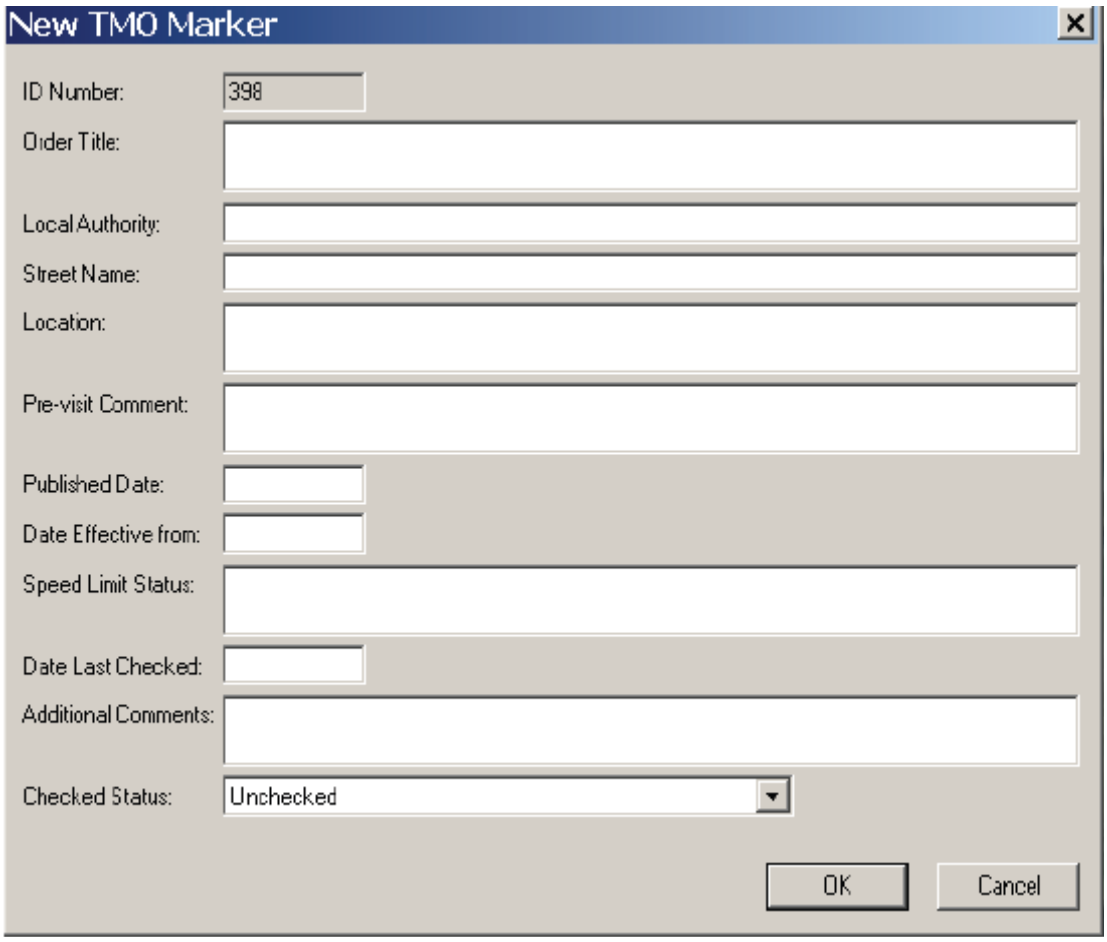


Figure 34 - Mapping a new TMO marker

4.6.3 REST service implementation

In order to exchange data from the TfL database, an XML extract of the data will be integrated with an on-line webservice for ad-hoc downloading by third parties. The XML extract will be as defined in the ROSATTE specification document. The service will allow extracts of the map at specific "versions" as defined in the ROSATTE data exchange methods document.
Essentially it would be a GET/POST request with appropriate parameters (i.e. version control) that will return an XML data stream (i.e. XML file) i.e. ultimately a REST request. This has not yet been implemented due to budget cuts. It is expected that this will be set up in the future, but in the meantime we can provide full data and periodic updates on request by sending the XML by email or ftp.

5. Commonalities and differences

In Table 3 below an overview is provided concerning commonalities and differences between the test sites for several relevant characteristics. The most distinct variation occurs in the location referencing method used: four different methods. These concern binary AGORA-C, xml AGORA-C, OpenLR and a specific BALI xml encoding. The Flanders test site only provides point features, all other test sites provide both point and linear features (and corresponding location encoding). All test sites use test site specific legacy software, on top of which the exchange infrastructure is implemented. All test sites except the BALI test site implemented the exchanges infrastructure according to the ROSATTE specification. Flanders provides traffic sign information, while all other test sites provide information concerning the regulation.

Criteria	ASFA	BALI	Bavaria	Flanders	Norway/Sweden	London
location referencing method	binary AGORA-C	specific BALI xml	OpenLR	binary AGORA-C	xml AGORA-C	OpenLR
encoding type (point/line)	line	line	line	point	line	line
test site specific legacy software	yes	yes	yes	yes	yes	yes
exchange format	ROSATTE	test site specific	ROSATTE	ROSATTE	ROSATTE	ROSATTE
feature type	regulation	regulation	regulation	sign information	regulation	regulation

Table 3 - Test site commonalities and differences for relevant characteristics

6. Conclusions

No significant cooperation existed between the different test sites concerning the development or harmonisation of software modules. This is especially due to the different nature and characteristics of the sites, and the resulting differences in the implementations. Differences concern e.g. legacy systems in place, types of data available, data formats used, software tools used for internal development, and data capturing methods in use). This limited cooperation and coordination should not be considered as a shortcoming, but as an advantage. Use of a variety of tools and approaches is likely to provide valuable information concerning strong and weak points, and for best practices. In addition, also in a future roll out such difference will exist. What is important is the common exchange format to which all outputs should be converted.

Definitions and acronyms

6.1 Definitions

Term	Definition
Traffic sign	Signs (e.g. speed limit signs) which are put up by road maintenance operators as a manifestation of a traffic regulation for driver information. Traffic signs, by their nature are point objects. To describe a line or an area regulation, several traffic signs may be put up for clarity of information to the drivers. Traffic signs data are often maintained as a part of separate data bases by road maintenance authorities in order to more efficiently manage road (side) equipment. While for certain signs, the content is close in meaning to the corresponding 'safety attribute', transformation is needed to create a safety attribute together with a correct location description. E.g. several traffic signs (repeated speed limit signs) may need to be analysed to define the location/extent of the corresponding safety attribute (speed restriction for several kilometres along a road).
Safety Feature / attribute	Feature/attribute in a digital road database which describes the content of a traffic regulation. To be useful, each safety attribute along a road must be paired by the description of its location. The location may be a point, a linear or an area location. To describe the location of a safety features/attributes it can be 'attached' to the road network by (logical) reference to the road database objects in order to clarify their location. Alternatively, a direct location description by coordinates is often used (geo-reference). Its details (as well as the location information) may be directly derived from a traffic regulation (or it could hold a reference to the regulation at its origin). Alternatively, its details (and location information) may be captured by field survey, or from databases including traffic signs. Note: In ROSATTE data stores at enacting authorities or in the digital maps of the information- or map providers, ROSATTE data may be represented either as separate features associated with locations at the road network or as attribution of the road network itself. The term "Safety attribute" in documents D1. and D2.1 refer to both the stored data at either end of the exchange and the exchanged data itself. The term Safety feature in D3.1 refers specifically to the representation of the data which is being exchanged in ROSATTE using the exchange specification. Therefore the terms may therefore be viewed as synonyms since it is the same real world entities that are being represented.
Linear Referencing	The Linear Reference System (LRS, also called Linear Referencing System) is a reference system in which features are localized by a measure along a linear element. Each feature is localized by either a point known as a "milepoint" or a linear event ("segment"). The system is designed so that if a segment of a route is changed only those milepoints on the changed segment need to be updated. (From Wikipedia). In the above classification, LRS is an indirect location referencing technique, which –in the case of a road network – uses 'routes' (a directed chain of segments) as 'aggregated' linear objects, in reference to which a linear location is described either by distance measures (from km 115 to km 120 in reference to the start point at km 0) or percentages (from 55% to 58% of the route length).

Term	Definition
AGORA	is one on-the-fly location referencing method, which is made reference to throughout this document. AGORA requires a digital network description that includes (1) geometric road information, (2) which has a topology to allow routing functions and (3) which contains certain road attributes, such as 'form of way' and 'functional road class'. AGORA was initially developed in an EU funded research project and has evolved into an ISO standard.
Interface	An interface is a gateway to the functionality that a component exposes to other components or external systems.
Service	A service is a software system running on its own, not relying on user input, used by external components.
Dataset	A dataset is an identifiable collection of data.
Metadata	Metadata is data about data, information making it possible to discover available data types and structures, quality parameters, geographic coverage etc., without reading the actual datasets.
Component	A component is the whole or part of a software system, seen from the outside as one unit.

6.2 Acronyms

Acronym	Definition
BALI	BAse de données des LImites de vitesse (Speed limit database, French initiative)
Base64	The term Base64 refers to a specific MIME content transfer encoding. It is also used as a generic term for any similar encoding scheme that encodes binary data by treating it numerically and translating it into a base 64 representation. [en.wikipedia.org]
DG INFSO	Directorate-General for Information Society and Media (European Commission).
GIS	Geographic Information System is any system that captures, stores, analyzes, manages, and presents data that are linked to location. [en.wikipedia.org]
REST	Representational State Transfer
ROSATTE	Road Safety Attribute Exchange Infrastructure
UML	Unified Modelling Language [www.omg.org/uml]
WP	Work Package
XML	Extensible Markup Language: is a set of rules for encoding documents electronically. It is defined in the XML 1.0 Specification produced by the W3C and several other related specifications; all are fee-free open standards. [en.wikipedia.org]

Annex 1: TA XMLs

Example XML feeds

LocationTranslation.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.location.com/locationtranslation"
xmlns:adtlt="http://www.location.com/adtlt"
xmlns:lt="http://www.location.com/locationtranslation"
xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" version="0.1">

   <import namespace="http://www.location.com/adtlt" schemaLocation="ADTLT.xsd"/>
   <!-- ############## Top level elements ############## -->
   <element name="LocationTranslationRequest" type="lt:LocationTranslationRequestType"/>
   <complexType name="LocationTranslationRequestType">
      <sequence>
         <element ref="lt:LocationTranslationQuery" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="version" use="required" fixed="1.0.0"/>
   </complexType>
   <element name="LocationTranslationResponse" type="lt:LocationTranslationResponseType"/>
   <complexType name="LocationTranslationResponseType">
      <sequence>
         <element ref="lt:LocationTranslationAnswer" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="version" use="required" fixed="1.0.0"/>
   </complexType>
   <!-- ############ Second level elements ########### -->
   <element name="LocationTranslationQuery">
      <complexType>
         <complexContent>
            <extension base="lt:LocationTranslationQueryType">
               <attribute name="vendor" type="adtlt:MapVendorType" use="optional"/>
            </extension>
         </complexContent>
      </complexType>
   </element>
   <complexType name="LocationTranslationQueryType">
      <sequence>
         <element ref="adtlt:Location"/>
               <element name="TranslationOutput" type="adtlt:TranslationOutputType"
maxOccurs="unbounded"/>
      </sequence>
   </complexType>
   <element name="LocationTranslationAnswer" type="lt:LocationTranslationAnswerType"/>
   <complexType name="LocationTranslationAnswerType">
      <sequence maxOccurs="unbounded">
         <choice>
            <element ref="adtlt:Location" minOccurs="1" maxOccurs="1"/>
            <element name="Error" type="adtlt:Error" minOccurs="1" maxOccurs="1"/>
         </choice>
      </sequence>
      <attribute name="release" type="string" use="optional"/>
   </complexType>
</schema>
```

ADTLT.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns:gml="http://www.opengis.net/gml" xmlns:adtlt="http://www.location.com/adtlt"
xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.location.com/adtlt"
elementFormDefault="qualified" version="1.4">
                    <import namespace="http://www.opengis.net/gml"
schemaLocation="../gml/2.1.2/geometry.xsd"/>
  <!-- ############ Abstract datatypes for location translation############ -->
  <element name="TranslationOutput" type="adtlt:TranslationOutputType"/>
  <complexType name="TranslationOutputType">
    <attribute name="format" type="adtlt:TranslationOutputFormatType" use="required"/>
    <attribute name="version" type="string" use="required"/>
  </complexType>
  <element name="Location" type="adtlt:LocationType"/>
  <complexType name="LocationType">
    <choice>
      <element ref="gml:LineString"/>
      <element ref="adtlt:MapIdString"/>
      <element ref="adtlt:LocationReference"/>
      <element ref="adtlt:TMCLocation"/>
      <element name="GenericFormat" type="string"/>
    </choice>
    <attribute name="locationQuality" use="optional">
      <simpleType>
        <restriction base="double">
          <minInclusive value="0"/>
          <maxInclusive value="1"/>
        </restriction>
      </simpleType>
    </attribute>
    <attribute name="release" type="string" use="optional"/>
  </complexType>
  <element name="LocationReference" type="adtlt:LocationReferenceType"/>
  <complexType name="LocationReferenceType">
    <simpleContent>
      <extension base="base64Binary">
        <attribute name="format" type="adtlt:LocationReferenceFormatType" use="required"/>
        <attribute name="version" type="string" use="required"/>
      </extension>
    </simpleContent>
  </complexType>
  <element name="MapIdString" type="adtlt:MapIdStringType"/>
  <complexType name="MapIdStringType">
    <simpleContent>
      <extension base="string">
        <attribute name="delimiter" type="token" use="optional" default=" "/>
        <attribute name="startOffset" use="optional" default="0.0">
          <simpleType>
            <restriction base="float">
              <minInclusive value="0.0"/>
              <maxInclusive value="1.0"/>
            </restriction>
          </simpleType>
        </attribute>
        <attribute name="endOffset" use="optional" default="0.0">
          <simpleType>
```

```
                <restriction base="float">
                    <minInclusive value="0.0"/>
                    <maxInclusive value="1.0"/>
                </restriction>
            </simpleType>
        </attribute>
    </extension>
  </simpleContent>
 </complexType>
 <simpleType name="TranslationOutputFormatType">
    <restriction base="string">
        <enumeration value="AGORA-C"/>
        <enumeration value="OPENLR"/>
        <enumeration value="MAP-ID-STRING"/>
        <enumeration value="GML-LINESTRING"/>
        <enumeration value="GENERIC-FORMAT"/>
    </restriction>
 </simpleType>
 <simpleType name="LocationReferenceFormatType">
    <restriction base="string">
        <enumeration value="AGORA-C"/>
        <enumeration value="OPENLR"/>
    </restriction>
 </simpleType>
 <simpleType name="MapVendorType">
    <restriction base="string">
        <enumeration value="NAVTEQ"/>
        <enumeration value="TELE ATLAS"/>
        <enumeration value="SRA"/>
        <enumeration value="OBB"/>
    </restriction>
 </simpleType>
 <simpleType name="ResponseType">
    <restriction base="string">
        <enumeration value="SUCCESSFUL"/>
        <enumeration value="UNSUCCESSFUL"/>
    </restriction>
 </simpleType>
 <element name="TMCLocation" type="adtlt:TMCLocationType"/>
 <complexType name="TMCLocationType">
    <attribute name="delimiter" type="token"/>
    <attribute name="tmctableversion" type="string"/>
 </complexType>
 <simpleType name="Error">
    <restriction base="string"/>
 </simpleType>
</schema>
```

Example request (Map ID String to AGORA-C binary):

```
<?xml version="1.0" encoding="UTF-8"?>
<LocationTranslationRequest
        xmlns="http://www.location.com/locationtranslation"
        xmlns:adtlt="http://www.location.com/adtlt"
        version="1.0.0">
  <LocationTranslationQuery>
  <adtlt:Location xmlns:adtlt="http://www.location.com/adtlt" release="2009.02">
```

```
 <adtIt:MapIdString delimiter="," startOffset="0.3"
endOffset="0.7">103800013651562,103800013671562,103800025826586, -
103800013550562,103800013551562</adtIt:MapIdString>
 </adtIt:Location>
<TranslationOutput format="AGORA-C" version="3.0"/>
</LocationTranslationQuery>
</LocationTranslationRequest>
```

Annex 2: ASFA XML

Example of input sent to the TA webservice:

```
<?xml version="1.0" encoding="UTF-8"?>
<LocationTranslationRequest
        xmlns="http://www.location.com/locationtranslation"
        xmlns:adtlt="http://www.location.com/adtlt"
        version="1.0.0">
 <LocationTranslationQuery>
 <adtlt:Location xmlns:adtlt="http://www.location.com/adtlt" release="2009.02">
 <adtlt:MapIdString delimiter="," startOffset="0.3" endOffset="0.7">-152500039804489,-
152500039840062,-152500039840075,-152500039923006,-152500038549019</adtlt:MapIdString>
 </adtlt:Location>
<TranslationOutput format="AGORA-C" version="3.0"/>
</LocationTranslationQuery>
</LocationTranslationRequest>
```

Example of ROSATTE export:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rst:ROSATTESafetyFeatureDataset                                                    gml:id="i0"
    xmlns:rst="http://www.ertico.com/en/subprojects/rosatte/rst"
    xmlns:gml="http://www.opengis.net/gml"      xmlns:gmd="http://www.isotc211.org/2005/gmd"
    xmlns:gco="http://www.isotc211.org/2005/gco"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.ertico.com/en/subprojects/rosatte/rst .\\ROSATTE.xsd">
<gml:featureMember>
<rst:GenericSafetyFeature gml:id="d3751a7c-2164-4e2a-9ffe-43f19e44313a">
<rst:id>
<rst:SafetyFeatureId>
 <rst:providerId>ASFA-AREA</rst:providerId>
 <rst:id>100060</rst:id>
    </rst:SafetyFeatureId>
    </rst:id>
<rst:locationReference>
<rst:AgoraLocationString gml:id="0af9e179-2772-4e52-bad5-82ed559e5336">

    <rst:base64String>AS8BMAArAiAGBBMShCQEeScgw1OtQAABAQRBNDEwBAgH0Qj35oIIGAQHBtEI35
    6HYA==</rst:base64String>
 <rst:agoraBinaryVersion>3.0</rst:agoraBinaryVersion>
    </rst:AgoraLocationString>
    </rst:locationReference>
 <rst:validFrom>2010-01-12</rst:validFrom>
<rst:UpdateInfo>
 <rst:type />
    </rst:UpdateInfo>
 <rst:source>Regulation</rst:source>
<rst:encodedGeometry>
<gml:Point gml:id="9212ad24-e35d-4bd4-bdeb-558c8c800384" srsName="start">
 <gml:coordinates>6,29027576974344 46,0722758320229</gml:coordinates>
    </gml:Point>
<gml:Point gml:id="d5fda844-e267-4acc-b0f8-f766c79f991b" srsName="end">
 <gml:coordinates>6,28964641446151 46,0704588752577</gml:coordinates>
    </gml:Point>
    </rst:encodedGeometry>
 <rst:type>SpeedLimit</rst:type>
```

```
<rst:properties>
<rst:SafetyFeaturePropertyValue>
 <rst:type>MaximumSpeedLimit</rst:type>
<rst:propertyValue>
 <gml:measure uom="kmph">70</gml:measure>
    </rst:propertyValue>
    </rst:SafetyFeaturePropertyValue>
    </rst:properties>
    </rst:GenericSafetyFeature>
  </gml:featureMember>
 <gml:featureMember>
 <rst:GenericSafetyFeature gml:id="06673588-3b4e-4bd3-8efa-e33aedef2446">
 <rst:id>
 <rst:SafetyFeatureId>
 <rst:providerId>ASFA-COFIROUTE</rst:providerId>
 <rst:id>100169</rst:id>
    </rst:SafetyFeatureId>
    </rst:id>
 <rst:locationReference>
 <rst:AgoraLocationString gml:id="8c7b3f7e-8051-4b52-ab26-d6e9dc638921">
      <rst:base64String>ATUBMAAxAiAGBBYVxCwAPDUhJWemSIFxy0AAAQEDQTEwBBQTyE77z/1UnWD
      NQAADAQNBMTD/QA==</rst:base64String>
 <rst:agoraBinaryVersion>3.0</rst:agoraBinaryVersion>
    </rst:AgoraLocationString>
    </rst:locationReference>
 <rst:validFrom>2010-04-12</rst:validFrom>
<rst:updateInfo>
<rst:UpdateInfo>
 <rst:type>Remove</rst:type>
    </rst:UpdateInfo>
    </rst:updateInfo>
 <rst:source>Regulation</rst:source>
<rst:encodedGeometry>
<gml:Point gml:id="f47a29ff-51bd-4162-a1db-3538c47f3e25" srsName="start">
 <gml:coordinates>0,330043034015408 46,6113623038599</gml:coordinates>
    </gml:Point>
<gml:Point gml:id="e9b9e17f-796e-4b84-9d9f-af1237eb68e1" srsName="end">
 <gml:coordinates>0,307886324163768 46,5971554133767</gml:coordinates>
    </gml:Point>
    </rst:encodedGeometry>
 <rst:type>SpeedLimit</rst:type>
<rst:properties>
<rst:SafetyFeaturePropertyValue>
 <rst:type>MaximumSpeedLimit</rst:type>
<rst:propertyValue>
 <gml:measure uom="kmph">130</gml:measure>
    </rst:propertyValue>
    </rst:SafetyFeaturePropertyValue>
    </rst:properties>
    </rst:GenericSafetyFeature>
    </gml:featureMember>
<rst:metadata>
 <rst:datasetId>9cde05a7-2a5d-4bc4-aa79-e68c557ebeb8</rst:datasetId>
 <rst:datasetCreationTime>2010-04-12T17:00:31</rst:datasetCreationTime>
</rst:metadata>
<rst:type>Update</rst:type>
</rst:ROSATTESafetyFeatureDataset>
```

Annex 3: (SRA, NPRA) XML

ROSATTE Export file log example:

```
<?xml version="1.0" encoding="utf-8"?>
<ROSATTEExportLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <ExportedFilesInfo>
  <ExportedROSATTEFileInfo>
   <DataSetId>48b489af-4be0-40ab-b84d-2a699708f940</DataSetId>
   <ExportedDate>2010-02-25T13:50:05.9665014+01:00</ExportedDate>
   <Complete>true</Complete>
   <FileName>c:\ROSATTEData\Export2010-03-18\RosatteExport_100225_0150.xml</FileName>
  </ExportedROSATTEFileInfo>
  <ExportedROSATTEFileInfo>
   <DataSetId>2f87da37-82c3-4f0e-851e-5f7e9b3a282c</DataSetId>
   <ExportedDate>2010-02-25T14:02:23.9166+01:00</ExportedDate>
   <Complete>false</Complete>
   <FileName>c:\ROSATTEData\Export2010-03-18\RosatteExport_100225_0202.xml</FileName>
  </ExportedROSATTEFileInfo>
  <ExportedROSATTEFileInfo>
   <DataSetId>7af71268-ca95-47f0-b5e8-99acaf0a228c</DataSetId>
   <ExportedDate>2010-03-19T10:24:18.5495+01:00</ExportedDate>
   <Complete>false</Complete>
   <FileName>c:\ROSATTEData\Export2010-03-18\RosatteExport_100319_1024.xml</FileName>
  </ExportedROSATTEFileInfo>
 </ExportedFilesInfo>
</ROSATTEExportLog>
```