TRACK TRADE

Creating a Data Mart for Floating Car Data

COOP-CT-2006-032823
Co-operative Research Projects, FP6

Deliverable 1.1
Data Model and Interfaces

Deliverable lead contractor: DLR

Rene Kelpin, DLR rene.kelpin@dlr.de
Alexander Sohr, DLR alexander.sohr@dlr.de
Martina Umlauft, TUV umlauft@wit.tuwien.ac.at
Petra Brosch, TUV brosch@big.tuwien.ac.at
Gerhard Schimon, WIGEOGIS gs@wigeogis.at
Dieter Pfoser, CTI pfoser@cti.gr

Due data: 30-06-2007
Actual submission date: 09-08-2007

Abstract

Given the overall objective of TRACK&TRADE to create a data mart for FCD and related services, this report details (i) the data collection architecture and prototype and (ii) the TRACK&TRADE Data Model. The creation of a data collection prototype establishes the basis for the data mart by offering data providers a web service interface to deliver their data.

Research Academic Computer Technology Institute, Greece
German Aerospace Agency, Inst. of Traffic Research, Germany
Technical University of Vienna, Business Information Systems Group, Vienna
Talent SA, Greece
Geomatics SA, Greece
Emphasis Telematics, Greece
WiGeoGIS, Austria
Green Way System GmbH, Germany

Table of Contents

1. Introduction

FCD (floating car data) becomes more and more important for traffic management and fleet disposition. A lot of delivery trucks, taxis and public vehicles are equipped with GPS-technology by the manufacturer as a standard. At the same time communication between vehicles and main office becomes easier and cheaper through the use of new communication channels. The quality and reliability of FCD is increasing continuously because of the rising penetration rate of FCD.

So the ground is prepared for new traffic services and applications which are based on FCD. The objective of project TRACK&TRADE is the creation of a data mart for FCD and related services.

This report details initial work towards the realization of the data mart. It (i) details the data mediation architecture and prototype as well as (ii) provides the TRACK&TRADE data format.The creation of a data mediation prototype in Work Package 1 (WP1) establishes the basis for the data mart. This prototype offers data providers the possibility to deliver data to the TRACK&TRADE database via a web service interface.

In order to create a mediation service interface it is necessary to get to know the relevant data, their formats and attributes. Based on the information acquired in the data format survey a data format for TRACK&TRADE had to be defined which has to be able to describe all TRACK&TRADE relevant data. This data format has to contain all attributes necessary to allow (i) data providers to store their data in the TRACK&TRADE database and (ii) service providers to get data (subsets) out of the database to establish new value added services. Using the TRACK&TRADE data format a data mediation interface had to be designed and implemented which supports these accesses in terms of communication, transformation and security.

2. Data Format Survey

During the data format survey a lot of companies, institutes, projects and initiatives have been contacted in order to get detailed data format information. First the focus was on projects and companies using FCD or providing FCD based services to learn about their data format specification. But also other data sources (loop detectors, weather, …) can at least supplement FCD and have been investigated during the survey.

In order to find a common data format for the project TRACK&TRADE it was necessary to get to know the specifics of the data of the project partners as well as available external data. A unified data format has to be able to describe the data without loss of information.

2.1 Existing Data Sources and Formats

The approach was to search for existing FCD-projects and try to get information about the used data formats and standards. Then the standards had to be evaluated whether they are useful and applicable for TRACK&TRADE. The rest of the chapter gives an overview of the formats evaluated in the course of WP1.

2.1.1 DLR FCD for Berlin and Vienna

For the cities of Berlin and Vienna, the project partner DLR receives FCD from taxi fleets, which use a dispatch system manufactured by the company “Austrosoft”[1]. The GPS-equipped taxis send their position and status periodically to the head office for disposition purposes. DLR is connected to the dispatch systems of Berlin and Vienna and gets the anonymized data. The data becomes worthless for taxi dispatch after the end of the trip. After parsing and transformation to a proprietary and xml-based FCD-format (see Listing 1) the data is stored in an internal database. Several project related processing applications (map matching, travel time estimation, other services) are accessing this database.

Listing 1. DLR Austrosoft FCD Sample (German tag names).

```
<RESULT>
  <ERROR>
    <ID>0</ID>
  <ERROR>
  <PARAMETER>
    <GPSFAHRSTATISTIK>
      <FAHRT>
        <ID>5969930</ID>
        <STATUS>70</STATUS>
        <ZEITPUNKT>27.11.2006 17:09:03</ZEITPUNKT>
        <SEKUNDEN>18</SEKUNDEN>
        <SOLLZEIT>160</SOLLZEIT>
        <ABFAHRT>
          <X>13.4584004720</X>
          <Y>52.4583984375</Y>
        </ABFAHRT>
        <ZIEL>
          <X>13.4558329264</X>
          <Y>52.4614827474</Y>
        </ZIEL>
        <FAHRZIEL>
          <X>0.0000000000</X>
          <Y>0.0000000000</Y>
        </FAHRZIEL>
      </FAHRT>
      <FAHRT>
      ...
      </FAHRT>
    </GPSFAHRSTATISTIK>
  </PARAMETER>
</RESULT>
```

The sample rate of the Berlin FCD is 30 seconds, for Vienna it is 1 minute. DLR connects every 5 minutes via FTP to the taxi companies to get the data packages.

2.1.2 EMPHASIS FCD for Athens, Greece

The project partner EMPHASIS collects FCD for Athens, Greece. The data comes from about 100 vehicles (school busses, delivery trucks) with a sample rate of 30 seconds. The specific values that are collected are shown in Table 1.

Table 1. Emphasis FCD.

traTrackID	traVehicleID	traSpeed	traDirection	traDate	traReceived	posX	posY
5733888	10	1	30	13.03.2007 08:06	13.03.2007 08:07	457280	4213710
5733889	10	5	68	13.03.2007 08:07	13.03.2007 08:07	457300	4213760
5733942	10	1	258	13.03.2007 08:07	13.03.2007 08:08	457310	4213750
5734187	10	0	0	13.03.2007 08:08	13.03.2007 08:13	457310	4213750
5734188	10	3	96	13.03.2007 08:12	13.03.2007 08:13	457290	4213760
5734670	10	0	0	13.03.2007 08:12	13.03.2007 08:21	457290	4213760
5734671	10	11	314	13.03.2007 08:20	13.03.2007 08:21	457350	4213820
5734817	10	0	0	13.03.2007 08:20	13.03.2007 08:23	457340	4213820

There are syntactic and semantic differences between the DLR and the EMPHASIS FCD content. Besides the syntactic difference (simple CSV files, differing IDs), the Athens data also includes measured speed and direction with each GPS position sample. Do note that measured speed is provided by the GPS device. This is in contrast to derived speed as can be computing by dividing traveled distance over elapsed time between two position samples.

2.1.3 GREENWAY Loop Data

GREENWAY provides mobile traffic sensing equipment (e.g., for road construction sites) as well as related traffic management services such as adaptive re-routing of vehicles.

The GREENWAY data is not directly comparable to FCD but it can be useful to calibrate algorithms and services using FCD. The data are similar to loop data Vehicles are measured by means of radar sensors as they pass the road construction site. The sensors can detect speed and vehicle type. The GREENWAY application can give an indication of the status of the road which can be free, slightly congested (congestuion warning) or congested (1, 2, 3 respectively). Date and sensor number are encoded in the filename. The coordinates of the sensor are not encoded in the file and only known to operators.

The sensor-supported control system at road construction sites or other temporary road bottlenecks is connected with variable message signs. This helps to reduce traffic congestion and to increase the road safety.

The current format of GREENWAY sensors comes in two similar formats and is organized in rows as shown in Listing 2 (format 1) and Listing 3 (format 2). The second column in format 1 is the ID of the sensor in the local measurement installation.

Listing 2. Greenway Sensor Data, Format 1:
four columns with a filename like "log_v_sens_012_200512.txt"

Timestamp	VEZ	Status	Speed
20.12.2005 11:06:27	4	1	62,9
20.12.2005 11:43:55	3	1	104,0
20.12.2005 12:39:58	3	2	48,8
20.12.2005 12:41:51	3	1	62,5
20.12.2005 15:33:19	3	3	19,8
20.12.2005 15:33:50	3	2	39,6
20.12.2005 15:34:04	3	1	72,3
20.12.2005 15:38:47	3	2	50,5

VEZ … sensor id.

Listing 3. Greenway Sensor Data, Format 2:
three columns with a filename like "log_fz_012_vez005.txt"

Timestamp	Vehicle type	Speed
26.02.2006 11:11:52	Fehlm.	---
26.02.2006 11:12:09	PKW	108
26.02.2006 11:12:13	LKW Anh.	118
26.02.2006 11:12:19	PKW	131
26.02.2006 11:12:19	PKW	131
26.02.2006 11:12:19	PKW	131
26.02.2006 11:12:23	LKW	131
26.02.2006 11:12:38	Dummy	- - -

2.1.4 BMW XFCD

A very extensive format is the BMW XFCD[2] format. It covers not only common FCD data but also a lot of vehicle information like the status of rain sensors, fog lights, the ABS and so on.

BMW uses this data for onboard (in the car) applications and off-board (off the car) processing and evaluation. Figure 1 gives a comprehensive overview of the information XFCD is able to capture. The figure also shows the off-board processing and state detection BMW does with XFCD.

The BMW attributes are taken from the GATS/FCD declaration.

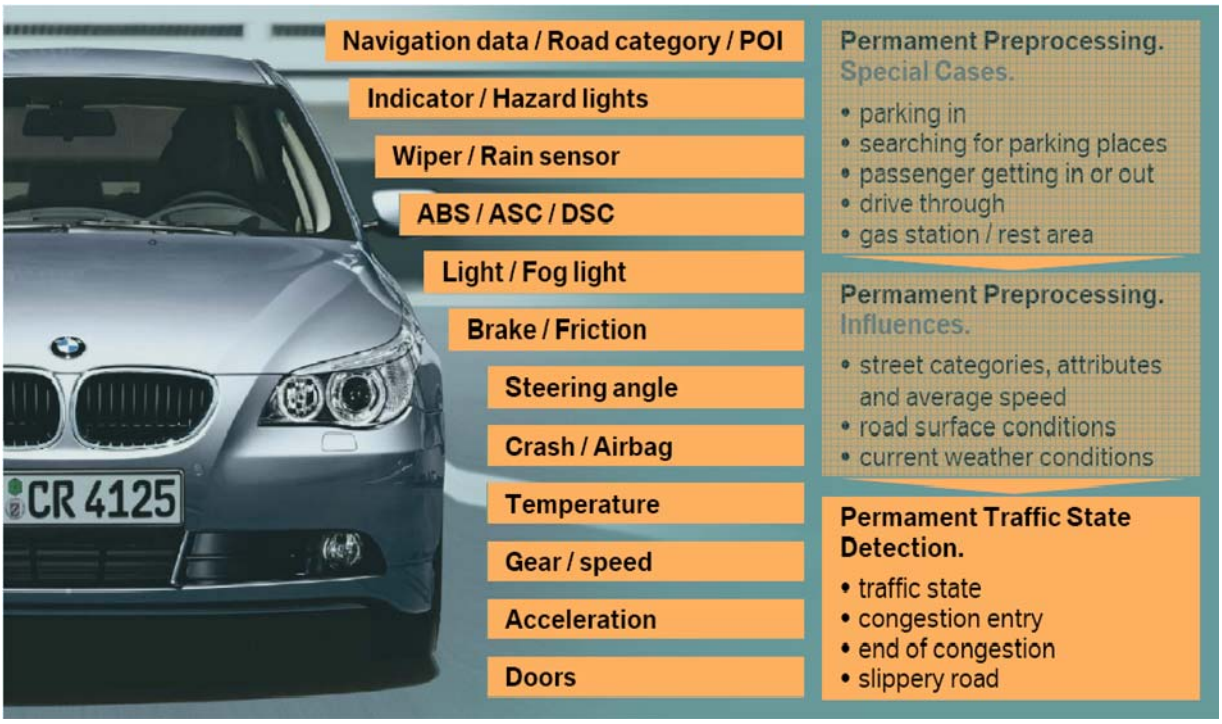


Figure 1. BMW XFCD

2.1.5 GEO Data

Considering the similarity of FCD with geodata, several existing formats that are used for geodata exchange were examined. A specific focus was given to formats used for Web mapping as this domain of user-contributed data is also directly relevant to the success of the TRACK&TRADE project.

The specific data formats that were examined are as follows.

- **GML (Geography Markup Language)** [[8]] is a standard defined by the Open Geospatial Consortium (OGC) to express geographical features. It serves as a modeling language for geographic systems as well as an open interchange format for geographic transactions on the Internet. An example is given in Listing 4.
- **KML (Keyhole Markup Language)** [[9]] is a popular XML-based language used in Google Earth, Google Maps, Google Mobile, ArcGIS Explorer and World Wind. The name stems from Keyhole, Inc, the company which originally develped the format which was acquired by Google in 2004. An example is given in Listing 5.
- **GeoRSS** [[10]] is used to encode locations in RSS feeds. RSS ("Really Simple Syndication") is an XML-based format used to publish frequently updated content such as blog entries, news headlines or podcasts (so called “feeds” or “channels”). In GeoRSS, the content consists of geographical points of interest and other annotations, and the feeds are designed to be consumed by geographic software such as map generators. It can be used to extend RSS version 1.0, 2.0, and Atom. There exist a GML and a simple version of GeoRSS. An example is given in Listing 6.

- **SensorML (Sensor Model Language)** [[11]] is a standard defined by the OGC using XML schema to provide descriptions of sensor systems. It supports a wide range of sensors and functions such as sensor discovery, sensor geolocation, processing of sensor observations, a sensor programming mechanism, and subscription to sensor alerts.

```
<!—Point-Object (srsName specifies CRS [Coordinate Reference System];
    EPSG:4326 ist WGS84) -->
    <gml:Point srsName="urn:ogc:def:crs:EPSG:6.6:4326">
      <gml:pos dimension="2">13.345 48.1234</gml:pos>
    </gml:Point>
<!—Line-Object -->
      <gml:LineString srsName="urn:ogc:def:crs:EPSG:6.6:4326">
      <gml:posList dimension="2">13.34 48.12 13.67 48.45 13.91
48.77</gml:posList>
    </gml:LineString >
<!—Polygon-Object with hole -->
    <gml:Polygon srsName="urn:ogc:def:crs:EPSG:6.6:4326">
      <gml:outerBoundaryIs>
        <gml:LinearRing>
          <gml:posList dimension="2">13.34 48.12 13.67 48.45 13.91 48.77
13.81 48.0 13.34 48.12</gml:posList>
        </gml:LinearRing>
      </gml:outerBoundaryIs>
      <gml:innerBoundaryIs>
        <gml:LinearRing>
          <gml:posList dimension="2">13.5 48.2 13.6 48.3 13.7 48.15 13.5
48.2</gml:posList>
        </gml:LinearRing>
      </gml:innerBoundaryIs>
    </gml:Polygon>
```

Listing 4. Example of GML Objects.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.1">
  <Document>
    <name>Flat Region</name>
    <Region>
      <LatLonAltBox>
        <north>37.430419921875</north>
        <south>37.41943359375</south>
        <east>-122.080078125</east>
        <west>-122.091064453125</west>
      </LatLonAltBox>
      <Lod>
        <minLodPixels>128</minLodPixels>
      </Lod>
    </Region>
    <GroundOverlay>
      <name>Mountain View DOQQ</name>
      <Icon>
        <href>files/image.JPEG</href>
      </Icon>
      <LatLonBox>
        <north>37.430419921875</north>
        <south>37.41943359375</south>
        <east>-122.080078125</east>
        <west>-122.091064453125</west>
      </LatLonBox>
    </GroundOverlay>
  <Document>
</kml>
```

Listing 5. Example of a KML Object.

```
<georss:point>45.256 -110.45</georss:point>
<georss:elev>313</georss:elev>
```

Listing 6. Example of a GeoRSS Object.

Other kinds of geodata formats (e.g. ESRI-Shape, Oracle-Spatial, coordinates in flat tables,…) are not compatible with the desired TRACK&TRADE format and have to be converted by an import-module.

GML offers the possibility of integration into any kind of proprietary or newly defined XML-format. It is also a format with great flexibility for using miscellaneous or complex geo-objects. Therefore, the decision was made to use GML for the definition of geo-objects in the XML-structure of TRACK&TRADE.

2.1.6 Weatherdata

Weatherdata is available in many different proprietary data formats (text-based, table-based,…). An example is shown in Figure 2.

Mostly, the following data are available:

- Weather condition, typically encoded into a status code according to some kind of reference table.
- Maximum and/or minimum temperature.
- Coordinates (Lon/Lat) or a textual representation of the location.
- Timestamp.

Daten in Tabelle 'WETTER' in 'TRACK_AND_TRADE' auf 'SQL1'

id	LAT	LON	SYM	MINVAL	MAXVAL	TIMESTAMP
2250	16,65450878	48,02868318	9	8	12	06.11.2006 11:24:49
2252	16,72577426	48,0001624	9	8	12	06.11.2006 11:24:50
2254	16,71754623	48,06297686	9	8	12	06.11.2006 11:24:50
2256	16,87193857	48,06432876	9	8	12	06.11.2006 11:24:50
2258	16,94703184	48,04135293	9	8	12	06.11.2006 11:24:50
2260	16,92010449	48,00952557	9	8	12	06.11.2006 11:24:50
2262	16,38899977	48,03000099	3	8	11	06.11.2006 11:24:50
2264	16,43423946	47,91950579	5	9	12	06.11.2006 11:24:50
2266	16,33004998	47,88870653	3	6	12	06.11.2006 11:24:50
2268	16,33723211	47,85411831	3	6	12	06.11.2006 11:24:50
2270	16,21799795	48,00728583	3	6	12	06.11.2006 11:24:50
2272	16,28083014	47,99316347	3	6	12	06.11.2006 11:24:50
2274	16,36141143	47,99523503	3	8	11	06.11.2006 11:24:50
2276	16,32768458	47,94837786	3	6	12	06.11.2006 11:24:50
2278	16,26408633	47,93097408	3	6	12	06.11.2006 11:24:50
2280	16,13263333	48,05096008	3	6	12	06.11.2006 11:24:50
2282	16,05677273	48,0529285	3	6	12	06.11.2006 11:24:50
2284	16,23994797	47,95363077	3	6	12	06.11.2006 11:24:50
2286	16,16723199	47,9128886	3	6	12	06.11.2006 11:24:50
2288	16,09129994	47,92162896	3	6	12	06.11.2006 11:24:50
2290	15,96353015	47,972826	5	5	10	06.11.2006 11:24:50
2292	15,98088203	48,0289869	5	5	10	06.11.2006 11:24:50
2294	16,2538451	47,89682415	3	6	12	06.11.2006 11:24:50

Figure 2. Screenshot of Example of a Weather Table.

Weatherdata, which will be used within the TRACK&TRADE project as a means to verify FCD and associated travel times, has to be geocoded. That means that the data must either be made available with coordinates already included by the data provider or there must exist some other means of determining the coordinates when the data is uploaded to the TRACK&TRADE service. In this case the coordinates will be calculated by a converter function implemented within TRACK&TRADE.

2.1.7 TMC

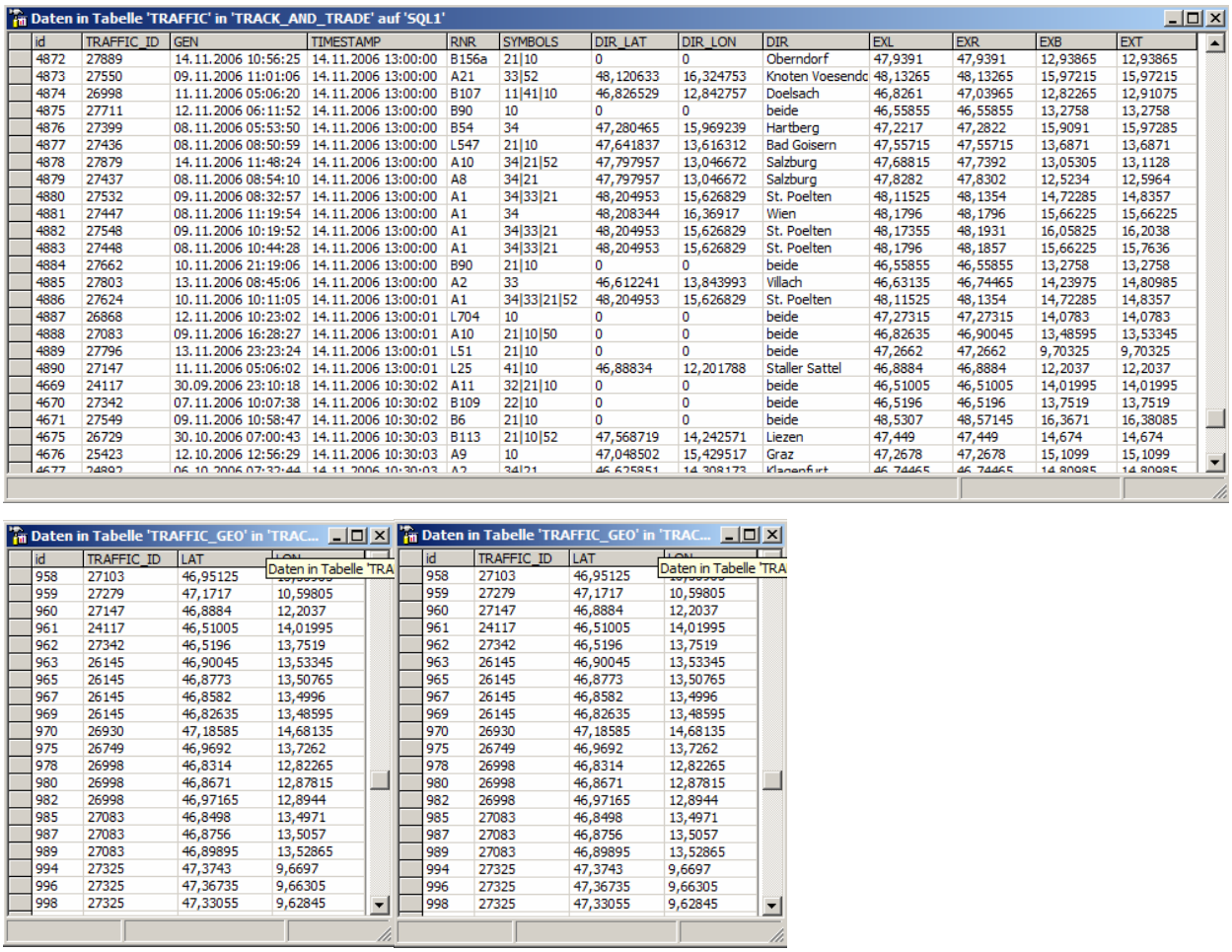
Traffic data is based on traffic events like accidents, road works or traffic jams.

Many European Countries use TMC (Traffic Message Channel) [[7]] to distribute traffic data to the public. Therefore, traffic data is stored in central places for each specific country.

This data can be used to join driving speeds on several road sections with traffic events.

Traffic data is mostly stored with following parameters:

- Initial Timestamp.
- Change/Renewal Timestamp.
- Roadnumber.
- Type of traffic event.
- Driving direction (important for highways!).
- Geographical extent of traffic event.
- Coordinates of event or textual representation of location.



Daten in Tabelle 'TRAFFIC' in 'TRACK_AND_TRADE' auf 'SQL1'

id	TRAFFIC_ID	GEN	TIMESTAMP	RNR	SYMBOLS	DIR_LAT	DIR_LON	DIR	EXL	EXR	EXB	EXT
4872	27889	14.11.2006 10:56:25	14.11.2006 13:00:00	B156a	21|10	0	0	Oberndorf	47,9391	47,9391	12,93865	12,93865
4873	27550	09.11.2006 11:01:06	14.11.2006 13:00:00	A21	33|52	48,120633	16,324753	Knoten Voesendc	48,13265	48,13265	15,97215	15,97215
4874	26998	11.11.2006 05:06:20	14.11.2006 13:00:00	B107	11|41|10	46,826529	12,842757	Doelsach	46,8261	47,03965	12,82265	12,91075
4875	27711	12.11.2006 06:11:52	14.11.2006 13:00:00	B90	10	0	0	beide	46,55855	46,55855	13,2758	13,2758
4876	27399	08.11.2006 05:53:50	14.11.2006 13:00:00	B54	34	47,280465	15,969239	Hartberg	47,2217	47,2822	15,9091	15,97285
4877	27436	08.11.2006 08:50:59	14.11.2006 13:00:00	L547	21|10	47,641837	13,616312	Bad Goisern	47,55715	47,55715	13,6871	13,6871
4878	27879	14.11.2006 11:48:24	14.11.2006 13:00:00	A10	34|21|52	47,797957	13,046672	Salzburg	47,68815	47,7392	13,05305	13,1128
4879	27437	08.11.2006 08:54:10	14.11.2006 13:00:00	A8	34|21	47,797957	13,046672	Salzburg	47,8282	47,8302	12,5234	12,5964
4880	27532	09.11.2006 08:32:57	14.11.2006 13:00:00	A1	34|33|21	48,204953	15,626829	St. Poelten	48,11525	48,1354	14,72285	14,8357
4881	27447	08.11.2006 11:19:54	14.11.2006 13:00:00	A1	34	48,208344	16,36917	Wien	48,1796	48,1796	15,66225	15,66225
4882	27548	09.11.2006 10:19:52	14.11.2006 13:00:00	A1	34|33|21	48,204953	15,626829	St. Poelten	48,17355	48,1931	16,05825	16,2038
4883	27448	08.11.2006 10:44:28	14.11.2006 13:00:00	A1	34|33|21	48,204953	15,626829	St. Poelten	48,1796	48,1857	15,66225	15,7636
4884	27662	10.11.2006 21:19:06	14.11.2006 13:00:00	B90	21|10	0	0	beide	46,55855	46,55855	13,2758	13,2758
4885	27803	13.11.2006 08:45:06	14.11.2006 13:00:00	A2	33	46,612241	13,843993	Villach	46,63135	46,74465	14,23975	14,80985
4886	27624	10.11.2006 10:11:05	14.11.2006 13:00:01	A1	34|33|21|52	48,204953	15,626829	St. Poelten	48,11525	48,1354	14,72285	14,8357
4887	26868	12.11.2006 10:23:02	14.11.2006 13:00:01	L704	10	0	0	beide	47,27315	47,27315	14,0783	14,0783
4888	27083	09.11.2006 16:28:27	14.11.2006 13:00:01	A10	21|10|50	0	0	beide	46,82635	46,90045	13,48595	13,53345
4889	27796	13.11.2006 23:23:24	14.11.2006 13:00:01	L51	21|10	0	0	beide	47,2662	47,2662	9,70325	9,70325
4890	27147	11.11.2006 05:06:02	14.11.2006 13:00:01	L25	41|10	46,88834	12,201788	Staller Sattel	46,8884	46,8884	12,2037	12,2037
4669	24117	30.09.2006 23:10:18	14.11.2006 10:30:02	A11	32|21|10	0	0	beide	46,51005	46,51005	14,01995	14,01995
4670	27342	07.11.2006 10:07:38	14.11.2006 10:30:02	B109	22|10	0	0	beide	46,5196	46,5196	13,7519	13,7519
4671	27549	09.11.2006 10:58:47	14.11.2006 10:30:02	B6	21|10	0	0	beide	48,5307	48,57145	16,3671	16,38085
4675	26729	30.10.2006 07:00:43	14.11.2006 10:30:03	B113	21|10|52	47,568719	14,242571	Liezen	47,449	47,449	14,674	14,674
4676	25423	12.10.2006 12:56:29	14.11.2006 10:30:03	A9	10	47,048502	15,429517	Graz	47,2678	47,2678	15,1099	15,1099
4677	24892	06.10.2006 07:32:44	14.11.2006 10:30:03	A2	34|21	46,625851	14,308173	Klagenfurt	46,74465	46,74465	14,80985	14,80985

Daten in Tabelle 'TRAFFIC_GEO' in 'TRAC...

id	TRAFFIC_ID	LAT	LON
958	27103	46,95125	
959	27279	47,1717	10,59805
960	27147	46,8884	12,2037
961	24117	46,51005	14,01995
962	27342	46,5196	13,7519
963	26145	46,90045	13,53345
965	26145	46,8773	13,50765
967	26145	46,8582	13,4996
969	26145	46,82635	13,48595
970	26930	47,18585	14,68135
975	26749	46,9692	13,7262
978	26998	46,8314	12,82265
980	26998	46,8671	12,87815
982	26998	46,97165	12,8944
985	27083	46,8498	13,4971
987	27083	46,8756	13,5057
989	27083	46,89895	13,52865
994	27325	47,3743	9,6697
996	27325	47,36735	9,66305
998	27325	47,33055	9,62845

Daten in Tabelle 'TRAFFIC_GEO' in 'TRAC...

id	TRAFFIC_ID	LAT	LON
958	27103	46,95125	
959	27279	47,1717	10,59805
960	27147	46,8884	12,2037
961	24117	46,51005	14,01995
962	27342	46,5196	13,7519
963	26145	46,90045	13,53345
965	26145	46,8773	13,50765
967	26145	46,8582	13,4996
969	26145	46,82635	13,48595
970	26930	47,18585	14,68135
975	26749	46,9692	13,7262
978	26998	46,8314	12,82265
980	26998	46,8671	12,87815
982	26998	46,97165	12,8944
985	27083	46,8498	13,4971
987	27083	46,8756	13,5057
989	27083	46,89895	13,52865
994	27325	47,3743	9,6697
996	27325	47,36735	9,66305
998	27325	47,33055	9,62845

Figure 3. Screenshots of Example of Traffic Data Tables.

TMC codes are defined EU-wide by the so called „Alert-C list“ with 1460 entries which defines TMC codes and location entries and is specified in ISO standard 14819, parts 1-3 (see [[12] , [13] , [14]] respectively). Examples are shown in Figure 4 and Figure 5.

Line	Text CEN-English)	Text (German)	Text (Quantifier = Einzahl)	Text (Quantifier = Mehrzahl)	Code	N	Q	T	D	U	C	R
2												
3	**1. LEVEL OF SERVICE**	**1. Verkehrslage**										
4												
5	traffic problem	Verkehrsbehinderung			1			D	1	U	1	A5
6	stationary traffic	(L) Stau			101			D	1	U	1	A1
7	stationary traffic for 1 km	1 km Stau			102			D	1	U	1	A1
8	stationary traffic for 2 km	2 km Stau			103			D	1	U	1	A1
9	stationary traffic for 3 km	3 km Stau			129			D	1	U	1	A1
10	stationary traffic for 4 km	4 km Stau			104			D	1	U	1	A1
11	stationary traffic for 6 km	6 km Stau			105			D	1	U	1	A1
12	stationary traffic for 10 km	10 km Stau			106			D	1	U	1	A1
13	danger of stationary traffic	Staugefahr			130			D	1	U	1	A1
14	queuing traffic (with average speeds Q)	(L) stockender Verkehr			108		4	D	1	U	1	A2
15	queuing traffic for 1 km (with average speeds Q)	1 km stockender Verkehr			109		4	D	1	U	1	A2
16	queuing traffic for 2 km (with average speeds Q)	2 km stockender Verkehr			110		4	D	1	U	1	A2
17	queuing traffic for 3 km (with average speeds Q)	3 km stockender Verkehr			131		4	D	1	U	1	A2
18	queuing traffic for 4 km (with average speeds Q)	4 km stockender Verkehr			111		4	D	1	U	1	A2
19	queuing traffic for 6 km (with average speeds Q)	6 km stockender Verkehr			112		4	D	1	U	1	A2
20	queuing traffic for 10 km (with average speeds Q)	10 km stockender Verkehr			113		4	D	1	U	1	A2
21	danger of queuing traffic (with average speeds Q)	Gefahr von stockendem Verkehr			132		4	D	1	U	1	A2
22	long queues (with average speeds Q)	lange Staus			133		4	D	1	U	1	A7
23	slow traffic (with average speeds Q)	(L) dichter Verkehr			115		4	D	1	U	1	A3
24	slow traffic for 1 km (with average speeds Q)	1 km dichter Verkehr			116		4	D	1	U	1	A3
25	slow traffic for 2 km (with average speeds Q)	2 km dichter Verkehr			117		4	D	1	U	1	A3
26	slow traffic for 3 km (with average speeds Q)	3 km dichter Verkehr			134		4	D	1	U	1	A3
27	slow traffic for 4 km (with average speeds Q)	4 km dichter Verkehr			118		4	D	1	U	1	A3
28	slow traffic for 6 km (with average speeds Q)	6 km dichter Verkehr			119		4	D	1	U	1	A3
29	slow traffic for 10 km (with average speeds Q)	10 km dichter Verkehr			120		4	D	1	U	1	A3
30	heavy traffic (with average speeds Q)	(L) reger Verkehr			122		4	D	1		1	A4
31	traffic heavier than normal (with average speeds Q)	sehr viel dichterer Verkehr als normal			142		4	D	1		1	A1
32	traffic very much heavier than normal (with average speeds Q)	dichterer Verkehr als normal			143		4	D	1		1	A1
33	traffic flowing freely (with average speeds Q)	störungsfreier Verkehr			124		4	(D)	1		1	A5

Figure 4. Example Excerpt of Alert-C List for TMC.

LOCATIONCODE	TYPE	SUBTYPE	ROADNUMBER	ROADNAME	FIRST_NAME	SECOND_NAME	AREA_REFERENCE
1	A3	0			Deutschland		34196
2	A3	0			Frankreich		34196
3	A3	0			Belgien		34196
4	A3	0			Niederlande		34196
5	A3	0			Luxemburg		34196
6	A3	0			Schweiz		34196
7	A3	0			Tschechien		34196
8	A3	0			Polen		34196
9	A3	0			Österreich		34196
10	A3	0			Ungarn		34196
11	A3	0			Italien		34196
12	A3	0			Griechenland		34196
13	A3	0			Dänemark		34196
256	A7	0			Schleswig-Holstein		1
257	A7	0			Hamburg		1
258	A7	0			Niedersachsen		1
259	A7	0			Bremen		1
260	A7	0			Nordrhein-Westfalen		1
261	A7	0			Hessen		1
262	A7	0			Rheinland-Pfalz		1
263	A7	0			Baden-Württemberg		1
264	A7	0			Bayern		1
265	A7	0			Saarland		1
266	A7	0			Berlin		1
267	A7	0			Brandenburg		1
268	A7	0			Mecklenburg-Vorpommern		1
269	A7	0			Sachsen		1
270	A7	0			Sachsen-Anhalt		1
271	A7	0			Thüringen		1
272	A8	0			Arnsberg		260
273	A8	0			Braunschweig		258
274	A8	0			Chemnitz		269
275	A8	0			Reg.-Bez. Darmstadt		261
276	A8	0			Reg.-Bez. Dessau		270
277	A8	0			Detmold		260

Figure 5. Example Excerpt of Location Codes for TMC Data.

2.1.8 Loop Detector Data

Loop data is not directly comparable to FCD, but it was important to work out the necessary attributes to enable the TNT-format to describe this data too.

Loop detector data is obtained by using induction loops immersed into roads that can detect vehicles passing over them. In that it is a simple means to provide traffic counts. Loop detectors are typically used on major roads and intersections (highway ramps). In connection with mathematical traffic models, they provide an overview of the traffic situation in a given road network. The downside of this technology is that it is unreliable (failure) and costly to maintain.

Loop Detector data in the context of FCD can be used for calibration and evaluation of algorithms and for simulation. Thus, the TRACK&TRADE data format has to be able to describe this data as well.

Listing 7. Loop Data Format.

```
Intervallbeginn 22.03.2007 13:44:00
Id|qKFZ|qLKW|vPKW|vLKW|tNetto|Beleg|s|vKFZMittel|
EQ 40W_9,920_A 115 N HFS|5|3|105|87|115|2|12|101|
EQ 40W_9,920_A 115 N UFS1|6|0|115|0|97|2|7|118|
EQ 40W_9,920_A 115 N UFS2|2|0|131|0|254|0|9|128|
```

2.1.9 Cell-based Tracking Data

Cell phones can also be used as FCD sources. There are different research projects and commercial products dealing with cell based tracking data. The cell phone is used to determine the position and as a service agent. The systems are running on established communication networks (e.g., GPRS, UMTS) and need also information similar to FCD (e.g., timestamp, position, ID) – but strongly dependent on the application constraints. Some services use special GPS modules to get the GPS position of the cell phone. Other products use the communication cells to determine the position of the agent.

Unfortunately, until now there is no data sample available. However, the format provides a field to describe cell based tracking data in the TRACK&TRADE format.

2.1.10 Other Formats

During the data format survey the following other formats relevant to FCD have been analyzed:

- **GATS** (Global Automotive Telematics Standard)[3] is a complete architecture including database, protocols and network layers.
- **OTAP** (Open Travel Data Access Protocol)[4] mingles standards from communication technologies and formats. OTAP is an easy access point on the internet where Service Providers can retrieve traffic information from different sources and other information suppliers within a defined area. The data providers make their traffic information available through the internet by using standard technology and common formats.
- **Datex** (Datex2)[5] is a data model with a very complex data dictionary.

- **GTP** (Global Telematic Protocol)[6] is a further stage of GATS with even more complexity and more focussed on protocol means.

2.2 Survey Result

The data providers mentioned above represent the group of institutes, companies and projects who have answered the request for information. Unfortunately a lot of queries (mail, contact forms, phone) were never answered. Some information could be extracted from project and product web sites. However, the information often turned out to be obsolete and unsupported or linked to outdated formats and standards.

Generally speaking, the problem with the formats was the diversity of them, obsolete or proprietary standards, systems and protocols often created only for one special application.

None of these standards have been applicable for the TRACK&TRADE approach. They were too complex and difficult to implement. The formats also varied widely in content, amount of overhead and protocol dependencies. The analysis of the necessary attributes to describe the data on the one hand and of the known standards on the other hand led to the decision to create a customized XML-based, easy to implement and an “as slim as possible” TRACK&TRADE data-format.

The result of the data format survey is two-fold:

a) it showed the need to develop a new format for TRACK&TRADE and

b) it defined the following set of attributes all known data sources are describable with.

At least the following attributes have to be implemented:

- trip/vehicle ID
- geo code origin
- geo code destination
- geo code static
- timestamp
- speed
- status
- direction/angle
- edge id
- count car
- state
- duration target
- duration actual
- event type xFCD
- weather „value“
- mobile cell ID

Based on this set of attributes and with respect to the known data and necessary service functionality the TRACK&TRADE data format scheme described in Section 3 was defined.

3. The TRACK&TRADE Data Model

Based on the data format survey and according to the defined set of attributes a new format was created, which is described in the following.

The TRACK&TRADE Data Model follows the XML schema shown in the UML diagram in Figure 6.

This schema is **used to encode any incoming data from a data provider**. For each data set of OBSERVATIONS the attribute COMPANYID identifies the data provider. A provider can upload several measurement points at once, thereby giving the possibility to accumulate several measurements at the data provider's side and then uploading them all at once to the TRACK&TRADE service according to some upload interval (eg. 5 minutes). Such a set of OBSERVATIONS can consist of one or many separate entries of type OBSERVATION. Each observation is identified by a SRCID, and a timestamp (ACTUAL:TS) and position (POSITION). These values give the time and position when the measurement was made. This is common for *all* data sources.

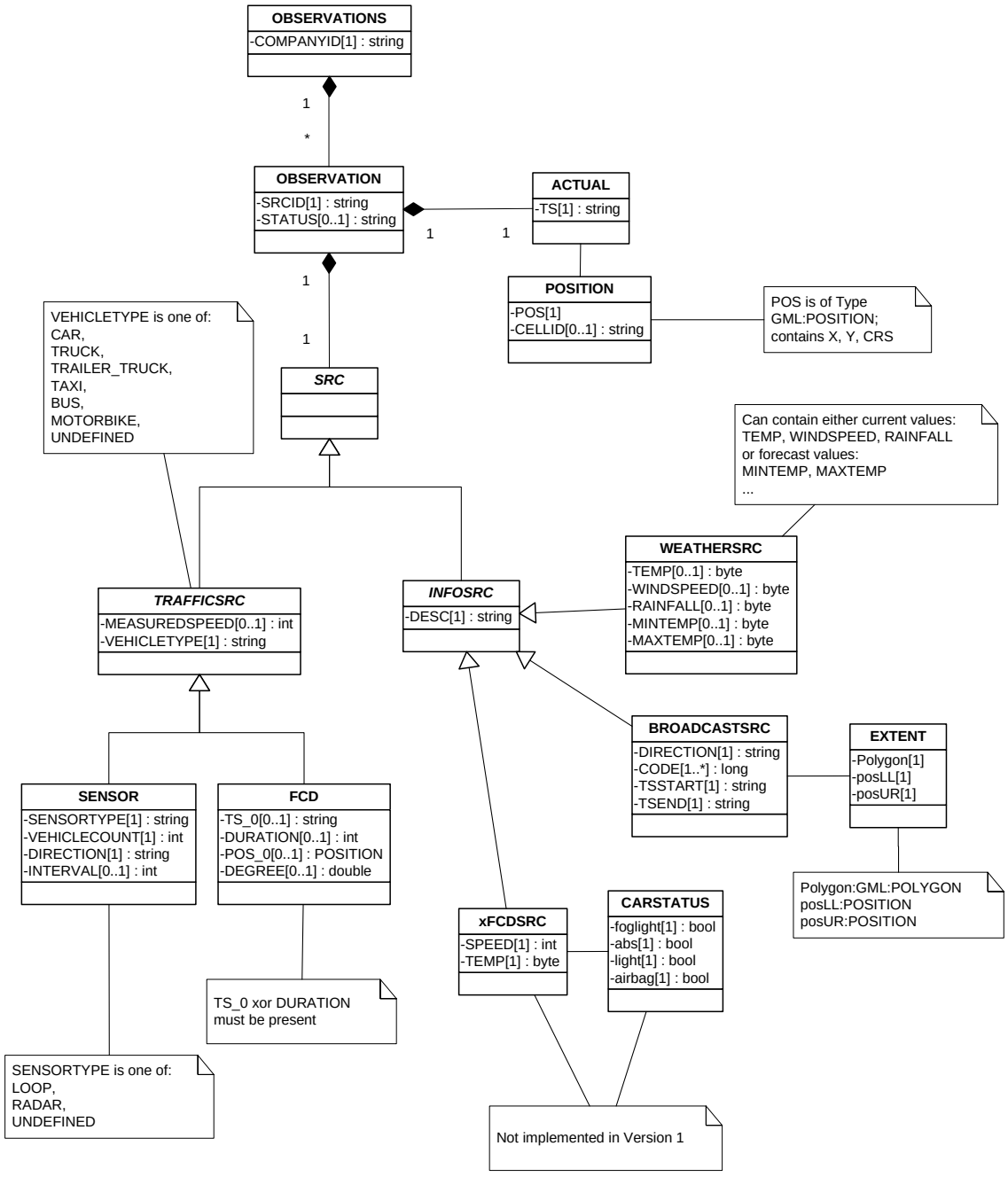


Figure 6. UML Diagram of XML Schema for Track & Trade Data Model.

We further distinguish between several **types of data sources** which have to provide different data:

- TRAFFICSRC: traffic sources can be further divided into SENSOR sources and FCD sources. SENSOR ist used to describe the data provided by all kinds of fixedly installed sensors (such as loop detectors or GREENWAY road construction site sensors). FCD is used to describe floating car data provided by taxis and other vehicle fleets.
- INFOSRC: info sources are all other sources, e.g., weather data providers, broadcast providers (radio or other media intended at a large audience, such as web pages), or car-specific data sources such as the BMW format described in Section 2.1.4.

Each of the fields is described in detail in Table 2. Listing 8 contains the complete XML schema. Detailed examples how data providers (a sensor company, a taxi company) should fill out this format are given in the Appendix.

3.1 Sensor Data

To correctly specify a sensor measurement, a data provider has to fill out the data fields common to all measurements and the fields specified under TRAFFICSRC and SENSOR. Different kinds of sensors are provided for. E.g., some sensors can only count vehicles passing by during a certain time interval but not determine the vehicle type or speed the field. E.g., a counting sensor mounted on top of a traffic light may report x number of vehicles detected in 5 minute intervals. For these cases, VEHICLETYPE must be set to “UNDEFINED”, MEASRUEDSPEED is omitted and only SENSORTYPE, VEHICLECOUNT, DIRECTION (driving direction of the road, eg. towards “Berlin”) and measurement INTERVAL are filled out.

Other types of sensors can detect the vehicle type and speed and report them for each single vehicle instead. For this kind of sensor, the data provider should fill out the common fields and SENSORTYP, as well as MEASUREDSPEED, VEHICLETYPE and DIRECTION for each vehicle reported by the sensor. Since only one vehicle is reported at a time, VEHICLECOUNT must be set to one. As no measurement interval is used, the INTERVAL field should be omitted. Some sensors provide an additional status code which gives an indication of the road status (e.g., free, lightly congested, congested,…). This status code can be reported in the OBSERVATION:STATUS field if available. Table 3 lists the status codes usable within TRACK&TRADE. A detailed description of how a provider of such sensor data should fill out the TRACK&TRADE XML file is given in Appendix A.1.

3.2 Specifying Floating Car Data (FCD) in TRACK&TRADE XML

Floating car data can be provided in several ways in the TRACK&TRADE format. For each measurement taken in a specific car the data provider must fill out a separate **OBSERVATION** and all the common data fields as mentioned above. An FCD measurement is usually defined by either two points in time or one point in time and a timespan (DURATION) and the accompanying positions. From these, the movement and speed of the vehicle can be calculated. The TRACK&TRADE format provides two ways to provide these data: the timestamp ACTUAL:TS and the position POSITON:POS always signify the *end* of the measurement. In the case of a

measurement with two points in time, this refers to the *second* point in time. The first point in time and position must be provided via FCD:TS_0 and FCD:POS_0.

In case of a measurement with one point in time and duration, this means that the measurement has to be provided in the form of the end point in time and a duration that has passed *before* that point in time. For an example of how a data provider using one timestamp and a duration should fill out the TRACK&TRADE XML see Appendix A.2. If the data provider measures vehicle movement by a starting point in time and a duration after that point in time they must convert their starting timestamp into an end timestamp by simply adding the duration before uploading the data to the TRACK&TRADE service.

Note that these two methods can not be used concurrently; iow. *either* two timestamps or a timestamp and a duration must be provided but not both.

Some in-vehicle systems also report the speed and/or direction in degrees as directly measured by the GPS device. These data can be provided via the MEASUREDSPEED and DEGREE fields respectively. If MEASUREDSPEED is provided directly by the GPS device then POS_0 should not be filled out to avoid inconsistencies between truly measured and calculated values.

A status code can be provided via the OBSERVATION:STATUS field. In the case of taxis or fleet vehicles the status can be used to provide additional information on the vehicle. E.g., if a taxi is at its stand waiting for customers its speed will not reflect the traffic conditions on the road. Table 3 lists the status codes usable within TRACK&TRADE.

3.3 Other Information Sources

All other data sources must be specified using **INFOSRC**. An INFOSRC can be either weather data, broadcast data, or in-vehicle data such as the XFCD format specified by BMW. Note that BROADCASTSRC is used to specify traffic data from any media which is “broadcast” by a provider and intended to be consumed by a wide audience, such as radio broadcasts, TMC transmissions, or web sites with traffic information. For a detailed description of the meaning of the fields used for these kinds of sources see Table 2.

Table 2. T&T Format Description.

T&T Format	Content	Note
OBSERVATIONS		Can group a number of observations (measurements) from the same company in one XML file. Eg. several GPS points from one or several taxis, several cars observed by a road sensor, … Each of those measurements is described by an OBSERVATION inside the OBSERVATIONS tag.
OBSERVATIONS:COMPANYID	=<unique company ID>	This is the unique data supplier ID in the T&T system. This ID is assigned by T&T.
OBSERVATION		Describes one concrete observation (measurement). Eg. ***one*** GPS measurement (waypoint) from a taxi, ***one*** vehicle measured by a sensor, or the number of vehicles measured within ***one*** time interval from a counting sensor.
OBSERVATION:SRCID	=<taxi number>, <sensor number>,…	This ID distinguishes the data sources of a data supplier and must be unique within a data supplier company. Eg. There MUST NOT be more than one taxi or sensor from the same supplier.
OBSERVATION:STATUS *(optional)*	=<status code>	Status code according to the list of status codes (see Table 2).
ACTUAL		Describes current time and position of the measurement. A measurement can be defined either by a) two timestamps (start, end) or b) one timestamp (end) and a duration. ACTUAL describes the ***end*** timestamp and associated position as this is when the measurement has finished.
ACTUAL:TS	=<timestamp>	Timestamp (UTC) of the ***end*** of the measurement. Format: *yyyy*-*mm*-*dd*T*hh*:*mm*:*ss* Example: 2006-12-15T09:58:12

ACTUAL:POSITION:POS	=<coordinates in GML>	Position at the ***end*** of the measurement. If the data comes from a sensor which does not move, this is simply the (fixed) position of the sensor. POS is of type GML:POSITON and contains the coordinates and CRS (coordinate reference system); eg. coordinates in WGS84.
ACTUAL:POSITION:CELLID *(optional)*	=<cell ID>	Cell ID if position is measured in a cellular phone network.
SRC *(abstract)*		This class is abstract; iow. it is only used for modeling but never instantiated in the XML file.
TRAFFICSRC *(abstract)*		This class is abstract; iow. it is only used for modeling but never instantiated in the XML file. The elements of TRAFFICSRC actually appear as elements of ist child classes but are listed here for conformity with the UML diagram. Eg. MEASUREDSPEED is described as an element of TRAFFICSRC but will actually be instantiated as either SENSOR:MEASUREDSPEED or FCD:MEASUREDSPEED.
TRAFFICSRC:MEASUREDSPEED (optional: either POS_0 xor MEASUREDSPEED must be present)	=<speed in km per hr>	The measurement can be defined either by a) two positions (start, end), see (POS_0) or b) one position (end) and a measured speed. The end position is always given (see ACTUAL). If POS_0 is not given, MEASUREDSPEED must be present. Speed in kilometers per hour.
TRAFFICSRC:VEHICLETYPE	=<type of vehicle>	One of the following: CAR, TRUCK, TRAILER_TRUCK, TAXI, BUS, MOTORBIKE, UNDEFINED.
SENSOR		Describes data obtained by road sensors.

SENSOR:VEHICLECOUNT	=<number of vehicles observed>	There exist two cases of sensors: a) sensors which report each vehicle with ist speed. These kinds of sensors should report 1 (one) as vehiclecount and report one OBSERVATION per vehicle measured. In this case, SENSOR:INTERVAL should be omitted. b) sensors which measure the number of vehicles passing by in a certain time interval. These sensors should report the number of vehicles measured and MUST also fill out SENSOR:INTERVAL to specify the measurement interval used.
SENSOR:SENSORTYPE	=<type of sensor>	One of the following: LOOP, RADAR, UNDEFINED.
SENSOR:DIRECTION	=<driving direction of observed lane>	Driving direction of the lane which is observed by the sensor; eg. "Berlin".
SENSOR:INTERVAL		This field is for sensors which do not report vehicles and their speed but the number of vehicles in a certain time interval (eg. Sensors mounted on traffic lights in Athens). (see VEHICLECOUNT). Sensors reporting each vehicle as a separate OBSERVATION should omit this value.
FCD		Describes Floating Car Data obtained from GPS-equipped vehicles.
FCD:TS_0 *(optional: either TS_0 xor DURATION must be present)*	=<timestamp>	Timestamp (UTC) of the ***beginning (start)*** of the measurement. Format: *yyyy*-*mm*-*dd*T*hh*:*mm*:*ss* Example: 2006-12-15T09:58:12 A measurement can be defined either by a) two timestamps (start, end) or b) one timestamp (end) and a duration. The end timestamp is always given (see ACTUAL). Therefore, either TS_0 or DURATION but not both (XOR) must be present in FCD.

FCD:DURATION *(optional: either DURATION xor TS_0 must be present)*	=<duration>	Duration in seconds. See description of FCD:TS_0 above.
FCD:POS_0 *(optional: either POS_0 xor MEASUREDSPEED must be present)*	=<coordinates>	Position at the ***start*** of the measurement. The measurement can be defined either by a) two positions (start, end) or b) one position (end) and a measured speed. The end position is always given (see ACTUAL). POS_0 is of type POSITON and contains POS with the coordinates and CRS and optionally a CELLID (see description of ACTUAL:POSITON above).
FCD:DEGREE *(optional)*	=<heading in degrees>	If supplied by GPS system, the current heading of the vehicle in degrees can be provided.
INFOSRC *(abstract)*		This class is abstract; iow. it is only used for modeling but never instantiated in the XML file. See also description of TRAFFICSRC.
INFOSRC:DESC	=<description string>	This is a freeform string containing a human-readable description of the information event.
WEATHERSRC		An information source describing either current weather conditions or forecasts. Either the following current values: TEMP, WINDSPEED, RAINFALL or the following forecast values: MINTEMP, MAXTEMP must be present.
WEATHERSRC:TEMP *(optional: either current or forecast values necessary)*	=<temperature in °C>	Current temperature.
WEATHERSRC:WINDSPEED *(optional: either current or forecast values necessary)*	=<windspeed in km per hr>	Current windspeed.
WEATHERSRC:RAINFALL *(optional: either current or forecast values necessary)*	=<rainfall in mm>	Current rainfall.
WEATHERSRC:MINTEMP *(optional: either current or forecast values necessary)*	=<min temp in °C>	Forecast minimum temperature.
WEATHERSRC:MAXTEMP *(optional: either current or forecast values necessary)*	=<max temp in °C>	Forecast maximum temperature.

BROADCASTSRC		Describes an information source which "broadcasts" traffic events to an audience, eg. traffic information on the radio or on the web.
BROADCASTSRC:DIRECTION	=<driving direction of observed lane>	Driving direction of the lane which is affected by the traffic event; eg. "Berlin".
BROADCASTSRC:CODE	=<message code>	A code[1] which describes the type of the traffic event; eg. 11=traffic jam, 34=road works, etc.
BROADCASTSRC:TSSTART	=<start time>	Start time of traffic event. Can be used by radio stations to determine when to start sending a certain message.
BROADCASTSRC:TSEND	=<end time>	End time of traffic event. Can be used by radio stations to determine when to stop sending a certain message.
BROADCASTSRC:EXTENT	=<geographical area>	The area affected by the traffic message is described giving a GML:POLYGON which is placed inside a rectangle specified by a lower left position (posLL) and an upper right position (posUR).
xFCDSRC		Describes floating car data as provided by some car manufacturers (eg. BMW). Not implemented in version 1 of T&T. xFCDSRC can contain SPEED, (current) TEMP and CARSTATUS which gives information on eg. the condition of foglights, the abs system, lights, or airbags.

1 Codes are defined by the so called „Alert-C list“ with 1460 entries which defines TMC codes and location entries EU-wide; see [[12] , [13] , [14]] as well as Figure 4 and Figure 5.

Table 3. List of Status Codes.

Code	Meaning	Note
1	Free road	From road sensors, eg. GREENWAY
2	Congestion warning (slightly congested)	From road sensors, eg. GREENWAY
3	Congestion	From road sensors, eg. GREENWAY
65	Registered	Taxi codes (as supplied by Austrosoft)
66	Occupied with customer	Taxi codes (as supplied by Austrosoft)
70	Free	Taxi codes (as supplied by Austrosoft)
75	Journey to customer	Taxi codes (as supplied by Austrosoft)
79	Occupied with follow-up job	Taxi codes (as supplied by Austrosoft)
83	At stand	Taxi codes (as supplied by Austrosoft)
90	Occupied with destination	Taxi codes (as supplied by Austrosoft)

Listing 8. T&T XML Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2006 rel. 3 U (http://www.altova.com) by pb
(Vienna University of Technology, Business Informatics) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tnt="http://tnt.trackandtrade.org/schema"
xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://tnt.trackandtrade.org/schema"
elementFormDefault="qualified" attributeFormDefault="qualified">
  <!-- gml import -->
  <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <!-- global types -->
  <xs:complexType name="position">
    <xs:sequence>
      <xs:element ref="gml:pos"/>
      <xs:element name="cellID" type="tnt:cellIdType" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="extent">
    <xs:sequence>
      <xs:element ref="gml:Polygon"/>
      <xs:element name="posLL" type="tnt:position"/>
      <xs:element name="posUR" type="tnt:position"/>
    </xs:sequence>
  </xs:complexType>
  <!-- src -->
  <xs:complexType name="src" abstract="true"/>
  <!-- traffic src -->
  <xs:complexType name="trafficsrc" abstract="true">
    <xs:complexContent>
      <xs:extension base="tnt:src">
        <xs:sequence>
          <xs:element name="measuredspeed" type="xs:integer"
minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="vehicletype">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="CAR"/>
              <xs:enumeration value="TRUCK"/>
              <xs:enumeration value="TRAILER_TRUCK"/>
              <xs:enumeration value="TAXI"/>
              <xs:enumeration value="BUS"/>
              <xs:enumeration value="MOTORBIKE"/>
              <xs:enumeration value="UNDEFINED"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="sensorsrc">
    <xs:complexContent>
      <xs:extension base="tnt:trafficsrc">
        <xs:sequence>
          <xs:element name="interval" type="xs:integer" minOccurs="0"/>
          <xs:element name="vehiclecount" type="xs:integer"/>
          <xs:element name="direction" type="xs:string"/>
```

```
        </xs:sequence>
        <xs:attribute name="sensortype">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="LOOP"/>
              <xs:enumeration value="RADAR"/>
              <xs:enumeration value="UNDEFINED"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="fcdsrc">
    <xs:complexContent>
      <xs:extension base="tnt:trafficsrc">
        <xs:sequence>
          <xs:element name="degree" type="xs:double" minOccurs="0"/>
          <xs:element name="position_0" type="tnt:position"
minOccurs="0"/>
          <xs:choice>
            <xs:element name="ts_0" type="xs:dateTime"/>
            <xs:element name="duration" type="xs:integer"/>
          </xs:choice>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- info src -->
  <xs:complexType name="infosrc" abstract="true">
    <xs:complexContent>
      <xs:extension base="tnt:src">
        <xs:sequence>
          <xs:element name="description" type="xs:string" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:group name="current">
    <xs:sequence>
      <xs:element name="temp" type="xs:byte" minOccurs="0"/>
      <xs:element name="windspeed" type="xs:byte" minOccurs="0"/>
      <xs:element name="rainfall" type="xs:byte" minOccurs="0"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="forecast">
    <xs:sequence>
      <xs:element name="mintemp" type="xs:byte"/>
      <xs:element name="maxtemp" type="xs:byte"/>
    </xs:sequence>
  </xs:group>
  <xs:complexType name="weathersrc">
    <xs:complexContent>
      <xs:extension base="tnt:infosrc">
        <xs:sequence>
          <xs:choice>
            <xs:group ref="tnt:current"/>
            <xs:group ref="tnt:forecast"/>
          </xs:choice>
        </xs:sequence>
```

```
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="broadcastsrc">
    <xs:complexContent>
      <xs:extension base="tnt:infosrc">
        <xs:sequence>
          <xs:element name="direction" type="xs:string"/>
          <xs:element name="code" type="xs:long" maxOccurs="unbounded"/>
          <xs:element name="extent" type="tnt:extent"/>
          <xs:element name="tsstart" type="xs:dateTime"/>
          <xs:element name="tsend" type="xs:dateTime"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- cellID data type -->
  <xs:simpleType name="cellIdType">
    <xs:restriction base="xs:string">
      <xs:pattern value="0x([0-9A-Fa-f]{2}){1,8}"/>
      <!-- cellID: max. 8 byte hex -->
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="observation_entry_type">
    <xs:sequence>
      <xs:element name="status" type="xs:string" minOccurs="0"/>
      <xs:element name="actual">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="ts" type="xs:dateTime"/>
            <xs:element name="position" type="tnt:position"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:choice>
        <xs:element name="weather" type="tnt:weathersrc"/>
        <xs:element name="broadcast" type="tnt:broadcastsrc"/>
        <xs:element name="fcd" type="tnt:fcdsrc"/>
        <xs:element name="sensor" type="tnt:sensorsrc"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="srcID" type="xs:long"/>
  </xs:complexType>
  <!-- root element -->
  <xs:element name="observations">
    <xs:complexType>
      <xs:sequence maxOccurs="unbounded">
        <xs:element name="observation" type="tnt:observation_entry_type"/>
      </xs:sequence>
      <xs:attribute name="companyID"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

4. Data Collection Architecture

Based on the TRACK&TRADE Data Model, a web service to upload data was created (see Figure 7). It allows data providers to connect their tracking data source to the TRACK&TRADE data mart. To achieve a maximum of flexibility and to support providers with different degrees of data conversion knowledge TRACK&TRADE offers **three data collection alternatives** as described in the following. All references are with respect to collection alternatives as shown in Figure 7.

4.1 Adding a new FCD Source Conforming to TNT Data Model (1)

Any device manufacturer can adopt this as the standard for new devices to communicate his data. The connection is established via SOAP/HTTP using the TRACK&TRADE XML Data Model described in Section 3. A template client interface will be provided for accessing the TNT data collection web service.

4.2 Adding Existing Data Sources (2)

Existing data sources can be added in two ways, either (2a) in terms of a Web service wrapper installed at the source (push approach) or (2b) by polling the data from the legacy source.

4.2.1 Existing FCD source using Web service + wrapper (2a)

For any new data source not conforming to the TNT data model (i.e., not falling into category (1) the Web service approach is used to connect the new data source. Connection happens at two levels, at (i) the communication and at (ii) the data model layer.

For the **communication layer**, Web service client templates will be provided that will have to be adapted to the specific needs of the new source. E.g., given that the data is available from a database using SQL queries, the web service layer will encapsulate this access and provide a standard (for the project) Web service interface.

At the **data model layer**, a translator (called “transformer” within TRACK&TRADE, denoted as “T” in Figure 7) will be created for each data source to provide mediation for each new source to the TNT data model. This translator will be on the central server side (NOT on the side of the legacy source).

4.2.2 Legacy data source (2b)

This approach represents the typical method that has been used so far for data collection. The specific interface of the data provider is examined and the legacy source is connected using a custom-built interface. Periodically, the data is collected (polled) from the source, i.e., the data provider has no control over when to deliver his FCD. This approach will be used to collect from those legacy data sources where it is not possible or desirable to install wrapper software on the source system. Examples of such data sources are DLR and EMPHASIS.

An HTTP-Poller has also been implemented, which can grab and parse data off of Web sites with traffic information. The HTTP-Poller as implemented is adapted to Athens Ministry of Public Order traffic information **Web page**, which contains a catalogue of major streets in Athens together with a traffic load description (normal, high, very high). The page is updated every half an hour, from 07:00 to 18:00 on working days. Unfortunately, the page is not very reliable. The HTTP-Poller can be adapted to parse other Web pages with little programming effort,

though one has to take into account that Web pages often change without notification which will require changes in the parser of the HTTP-Poller.

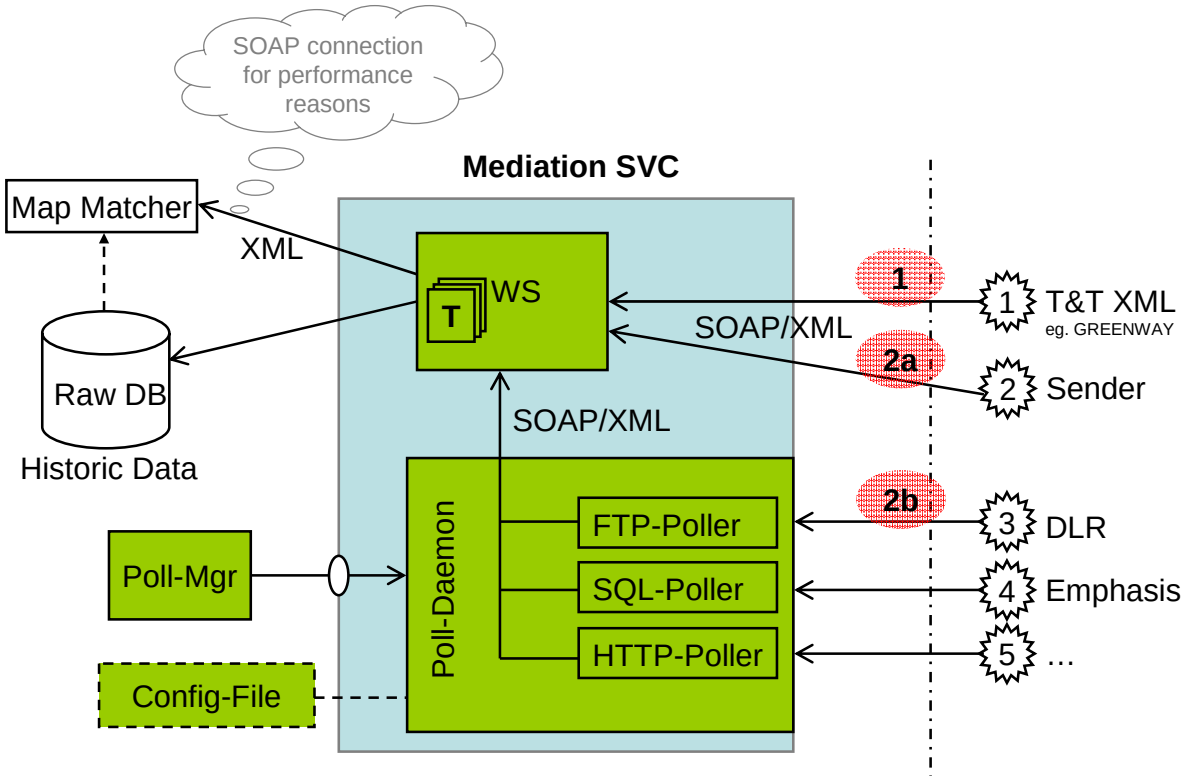


Figure 7. Data Collection Architecture.

4.3 Data Storage

The XML data collected through the various ways (1, 2a, 2b) will ultimately be stored in a database. However, for performance reasons, it is necessary to use main-memory structures for map-matching. Thus, the FCD data is directly fed into the map-matcher via a Web service interface. In parallel, FCD is also stored in the database.

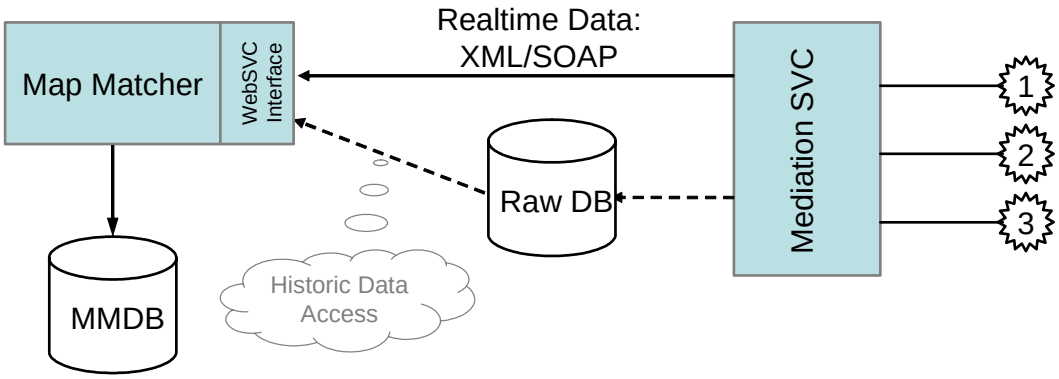


Figure 8. Data Transfer from Mediation Service to Map Matcher.

5. The Data Collection Prototype

This section describes the design, implementation and installation details of the data collection prototype. As described in Section 4 the data collection prototype consists of a Web service and the Poller service.

5.1 Web Service Design – New Data Sources

As shown in Figure 7, the Web Service is responsible for accepting XML and polled input data, transforming it and passing it on to the Map Matcher and historic FCD database.

Following approach (1) adding a new FCD source conforming to TNT data model, Figure 9 shows the UML class diagram of the TRACK&TRADE Web service client and the TRACK&TRADE Web service.

The Web service client implemented in package `org.tnt.client` is provided as a template to potential data providers to make access to the TRACK&TRADE platform easier. As can be seen, the client uses `MediationClientProxy` to access the class `DataMediationWS` which implements the Web service (package `org.tnt.service`).

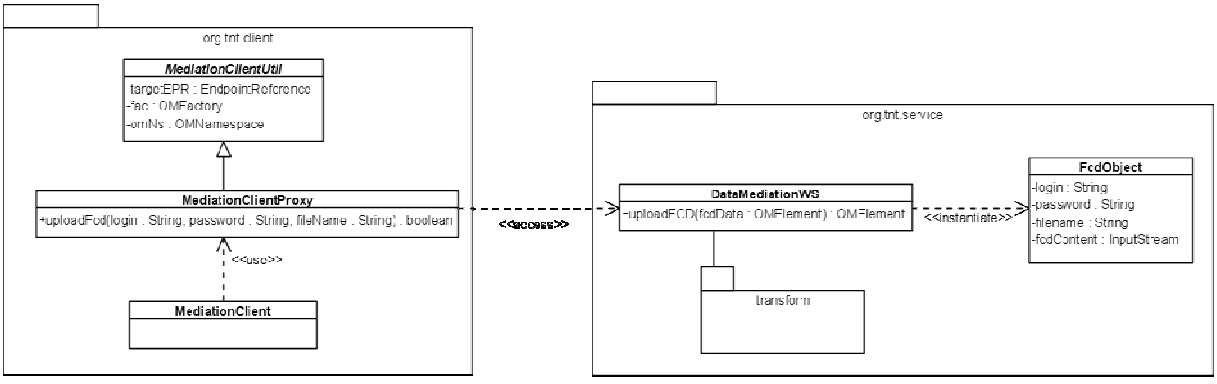


Figure 9. Web Service: Client and Server UML.

The Web service then uses either the classes in package `org.tnt.common.dataformat.schema` (see Figure 10) to handle the TRACK&TRADE

XML format (approach (1)), or the classes in package `org.tnt.service.transform` to translate a non-complient format (approaches (2a) and the result of data polling in approach (2b)). Transformation is described in Section 5.2 The details of data polling of legacy sources are described in Section 5.3. New data sources according to approaches (1) and (2a) may use the client implementation according to the UML diagam shown in Figure 9 which is provided by TRACK&TRADE in Java.

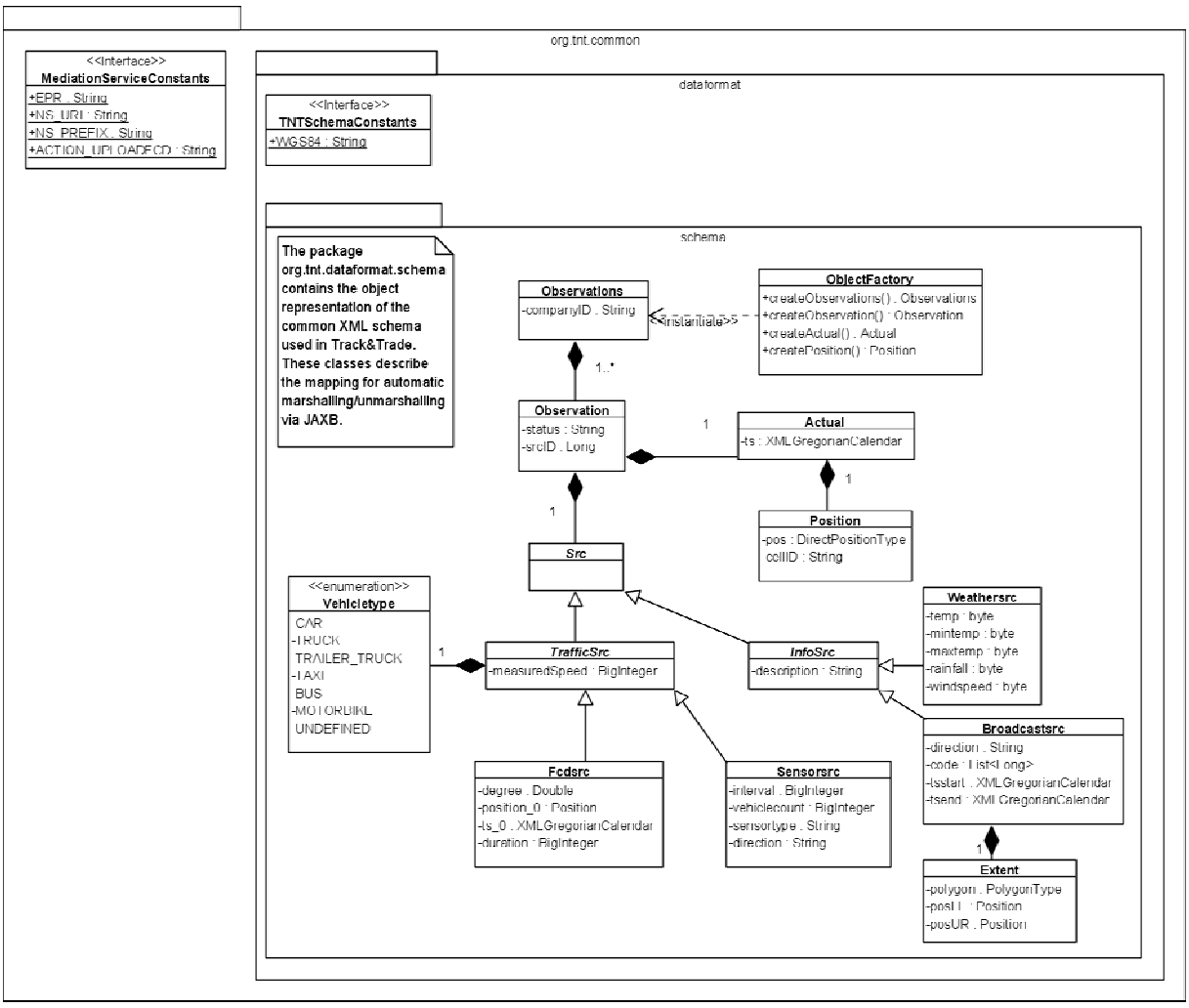


Figure 10. UML of Java Classes to Handle the T&T Format XML.

5.2 Transformation – Existing Data Sources

For approach (2a) and (2b) the data has to be translated from either a raw XML variant directly uploaded to the Web service or a proprietory format polled from the data provider. Once the data has been uploaded or polled, it is transformed into the TRACK&TRADE Format by the classes in package `org.tnt.service.transform`. All transformation classes have to implement the `Transformable` interface. The UML class diagram for the helper classes to handle and transform the XML Format provided by DLR (package `org.tnt.service.transform.dlr.schema`) is shown in Figure 11.

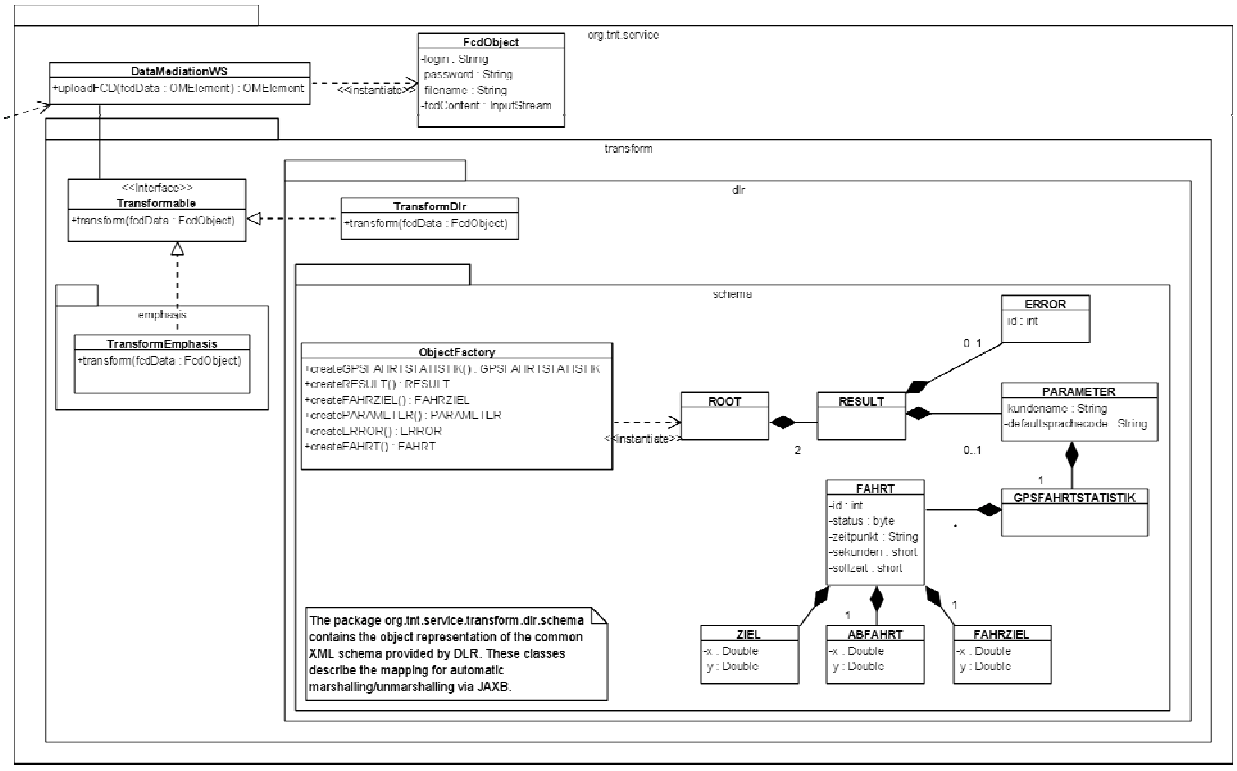


Figure 11. UML of Java Classes to Handle the DLR Data Format.

5.3 Poller Service Design

The Poller service is responsible for the polling of legacy data sources (scenario (2b) in Section 4). The data is actively polled from the data provider with a configurable frequency. This approach is implemented in scenarios where it is not desirable or possible to install or modify software at the data source. In this case it is necessary that there is a possibility to connect to the data source from outside, e.g., by using a service interface. The UML class diagram of the poller service is shown in Figure 12.

For each data provider type a separate class based on the `Poller` class needs to be implemented. This class implements the provider specific details of the access, eg. an FTP access in the case of DLR.

The poller service is separated into two components, the `PollDaemon` and the `PollManager`.

5.3.1 PollDaemon

The daemon is a program that runs in the background, but does the real work. The daemon collects the data from the various data providers by invoking the respective *xyz*`Poller` implementations. It can be configured by a simple config file (see Listing 9 for an example), which defines connection details like the host, protocol, username and password as well as the time interval the query should be performed.

5.3.2 PollManager

The PollManager program is the “remote control” for the PollDaemon. It is implemented as a command-line program and can pass commands from the user to the PollDaemon.

It uses socket connections to communicate with the PollDaemon to start or stop a poll thread or shutdown the process. Therefore, it is not necessary to shut down the daemon to manage or add a new poll thread.

By default port 4406 is used but can be configured differently via a start parameter.

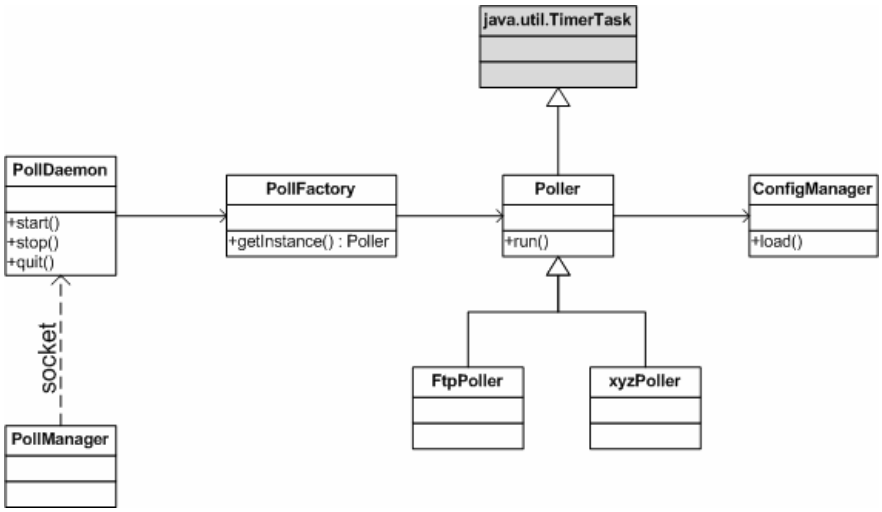


Figure 12. Class Diagram of the Poller Service.

As shown in Figure 12 the implementation is based on the Factory Pattern for simple extension to support new protocols. A specific Poller is derived from the abstract class `Poller` and implements the `run()` method to connect to the server and download the data.

5.3.3 Usage Example

The daemon is started as a background process with the command `nohup java -jar PollDaemon &` in a linux/unix shell.

Commands can be sent to the daemon with the PollManager application. Poll threads are started and stopped using the keywords start or stop in combination with the ID of the data provider, eg.:

```
java –jar PollManager “start dlr”
```

```
java –jar PollManager “stop dlr”
```

The daemon then loads the configuration file (`config.xml`, see Listing 9) which has to be placed in a directory with the same name as the ID of the data provider (e.g., for provider ID "dlr" the directory is named "dlr").

Listing 9. PollDaemon Configuration File (Example for DLR FCD).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <comment>config</comment>
  <entry key="id">dlr</entry>
  <entry key="interval">900000</entry> //poll interval in ms
  <entry key="user">tuv</entry>
  <entry key="password">fcd4tuv</entry>
  <entry key="protocol">FTP</entry>   //the protocol defines the
                                      //specific Poller Instance to use.
                                      //FTP -> FTPPoller
  <entry key="logLevel">INFO</entry> //value = java.util.Logging.Level
  <entry key="host">ftp.dlr.de</entry>
  <entry key="timeout">60000</entry> //connection timeout in ms
  <entry key="notificationEmail">mail@domain.com</entry> //not used
</properties>
```

The daemon is shut down using `java –jar PollManager quit`.

6. Conclusion

A main objective in TRACK&TRADE is to create a trading platform for FCD and supplementary data. The objective of the Work Package 1 (WP1) was the **creation of a data collection system** and the definition of an adequate **Data Model**.

In order to enable data providers to publish their data to an international community in a standardized manner and – based on that data - join a huge number of different data sources in a single data mart it is crucially necessary to know the content and the format of the data. Thus, in the course of the data format survey FCD and supplementary data sources have been analyzed with special interest on the necessary attributes to describe the data.

A format had to be found or designed that is able to describe very different data (e.g., FCD, weather, construction sites data, traffic messages, loop detectors). This has been accomplished with the TRACK&TRADE common data format described in this deliverable. All known data can be described with the common data format without loss of information. The format is as light-weight as possible, XML-based and easy to understand.

A further objective of WP1 was to support data providers to upload their data and to control data base access. No data provider has direct write access to the project database but instead connects via the TRACK&TRADE data collection web service.

The data have to be transformed into the TRACK&TRADE common data format before storing into the data base. To support new and legacy data sources three ways of connecting to the TRACK&TRADE data collection web service and transformation are provided.

- If the data provider is able to transform the data to the TRACK&TRADE data format (e.g., if a new data source is implemented) the data can be uploaded directly to the TRACK&TRADE data collection web service.

- If the data provider is able to install client software on their system they can use sample code (a web service client) provided by TRACK&TRADE to upload their raw data to the data collection web service. The data will then be transformed on the TRACK&TRADE server.
- If the data provider is neither able to transform the data nor to install client software but able to grant access to their system (e.g., by providing a login via the service interface) the data can be polled by T&T. It will then be transformed on the T&T server.

After upload/data polling and transformation (if necessary) the data is passed on to the map matching process and stored in the database by the data collection web service.

The data collection service is already established for the project partners EMPHASIS and DLR. The FCD from Athens, Berlin and Vienna are continuously uploaded to the TRACK&TRADE database.

The completion of the data collection service in the WP1 establishes the basis for the next steps to create the TRACK&TRADE data mart.

References

[1] Austrosoft Weiss Datenverarbeitung Ges.m.b.H. – http://www.austrosoft.net

[2] Extended Floating Car Data – Mobile Incident Detection, Susanne Breitenberger BMW Group – Presentation at ITS World Congress 2005

[3] GATS – Global Automotive Telematics Standard: http://www.ertico.com/en/subprojects/telematics_forum/news_events/news_events.htm

[4] OTAP – Open Travel data Access Protocol: http://www.itsproj.com/otap/OPT

[5] Datex2 – a Standard for the exchange of traffic related data – http://www.datex2.eu

[6] GTP – Global Telematic Protocol - http://www.ertico.com/en/subprojects/telematics_forum/public_documents.htm

[7] TMC – EPISODE - European Pre-operational Implementation Survey On further Development and Evaluation of RDS-TMC; http://www.rds.org.uk/episode/episode.htm

[8] GML – OGC GML - the Geography Markup Language; http://www.opengis.net/gml/

[9] KML – Google KML API Documentation; http://code.google.com/apis/kml/documentation/

[10] GeoRSS – GeoRSS Geographically Encoded Objects for RSS feeds; http://georss.org/

[11] SensorML – Sensor Markup Language at UAH; http://vast.uah.edu/SensorML/

[12] Iso standard 14819-1: Traffic and traveller information (tti) – tti messages via traffic message coding – part 1: Coding protocol for radio data system – traffic message channel (rds-tmc) using alert-c. http://www.iso.org, May 2003.

[13] Iso standard 14819-2: Traffic and traveller information (tti) – tti messages via traffic message coding – part 2: Event and information codes for radio data system –trafficmessage channel (rds-tmc). http://www.iso.org, May 2003.

[14] Iso/ts standard 14819-3: Traffic and traveller information (tti) – tti messages via traffic message coding – part 3: Location referencing for alert-c.http://www.iso.org, June 2000.

A. Appendix: Track and Trade XML Format Examples

This section contains two examples of how to use the TRACK&TRADE format. One is for a company which supplies road sensor data (GREENWAY), one is for a taxi company (DLR).

A.1 Road Sensor Data Example (GREENWAY)

GREENWAY provides sensor data from road construction sites. Vehicles are measured as they pass the road construction site. The sensors can detect speed and vehicle type and the Greenway application can give an indication of the status of the road (Zustand) which can be free, congestion warning (slightly congested) or congested (code 1, 2, 3 respectively). The date and sensor number are encoded in the filename. The coordinates of the sensor are not encoded in the file and only known to operators. The current format of GREENWAY sensors is described in detail in Section 2.1.3.

To transform the GREENWAY format, only the following fields in the TRACK&TRADE -Format must be filled out as shown in Table 4. In addition, some fields in the TRACK&TRADE format which do not exist in the column data of GREENWAY but must be filled out according to Table 5 (see also Table 2 for a detailed description of the meaning of each field. Listing 10 contains an example XML data set containing 2 measurements (of a TRUCK and of a CAR) on the B109 direction Berlin.

Table 4. Mapping of GREENWAY Column Data.

GREENWAY Format	T&T Format	Note
Col 1, timestamp	ACTUAL:TS	
Col 2, vehicle type (VEZ)	TRAFFICSRC:VEHICLETYPE	A mapping table must be used: PKW = CAR, LKW = TRUCK, LKW Anh. = TRAILER_TRUCK, Fehlm, Dummy -> should not be reported to T&T since there is no speed data.
Format 1, Col 4 and Format 2, Col 3, speed	TRAFFICSRC:MEASUREDSPEED	
Format 1, Col 3, status (Zustand)	OBSERVATION:STATUS	Transformed according to the T&T status code list (see Table 2).

Table 5. Additional Data to be Filled out by GREENWAY.

T&T Format	Content	Note
OBSERVATIONS:COMPANYID	=“GREENWAY“	
OBSERVATION:SRCID	=<Sensor Number>	Encoded in the filename. This number must be unique within GREENWAY; iow. there MUST NOT be two GREENWAY sensors which use the same number. Note: Currently, GREENWAY reuses sensor numbers when a measurement installation is moved from one construction site to the next. It is planned to introduce a combined installation and sensor id to make sensor ids unique in the future.
ACTUAL:POSITION	=<coordinates in GML>	Coordinates can not be inferred from file content or name; must be known intrinsically by GREENWAY.
SENSOR:VEHICLECOUNT	=1	Since every observation (each line of the GREENWAY file) should be a separate OBSERVATION in T&T format.
SENSOR:SENSORTYPE	=<type of sensor>	RADAR
SENSOR:DIRECTION	=<driving direction of observed lane>	Eg. "Berlin"
SENSOR:INTERVAL		Should not be filled out by GREENWAY. This field is for sensors which do not report vehicles and their speed but the number of vehicles in a certain time interval (eg. Sensors mounted on traffic lights in Athens).

Listing 10. Example Data Set for GREENWAY.

```
<?xml version="1.0" encoding="UTF-8"?>
<tnt:observations tnt:companyID="GREENWAY" xmlns:tnt="http://tnt.trackandtrade.org/schema"
xmlns:sch="http://www.ascc.net/xml/schematron" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:smil20="http://www.w3.org/2001/SMIL20/"
xmlns:smil20lang="http://www.w3.org/2001/SMIL20/Language"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://tnt.trackandtrade.org/schemaTNT.xsd">
  <tnt:observation tnt:srcID="012">
    <tnt:status>1</tnt:status>
    <tnt:actual>
      <tnt:ts>2006-12-15T09:58:12</tnt:ts>
      <tnt:position>
        <gml:pos srsName="urn:ogc:def:crs:EPSG:6.6:4326">13.498363888888887
52.81587777777777</gml:pos>
      </tnt:position>
    </tnt:actual>
    <tnt:sensor tnt:vehicletype="TRUCK" tnt:sensortype="RADAR">
      <tnt:measuredspeed>63</tnt:measuredspeed>
      <tnt:vehiclecount>1</tnt:vehiclecount>
      <tnt:direction>Berlin</tnt:direction>
    </tnt:sensor>
  </tnt:observation>
  <tnt:observation tnt:srcID="012">
    <tnt:status>1</tnt:status>
    <tnt:actual>
      <tnt:ts>2006-12-15T09:59:52</tnt:ts>
      <tnt:position>
        <gml:pos srsName="urn:ogc:def:crs:EPSG:6.6:4326">13.498363888888887
52.81587777777777</gml:pos>
      </tnt:position>
    </tnt:actual>
    <tnt:sensor tnt:vehicletype="CAR" tnt:sensortype="RADAR">
      <tnt:measuredspeed>104</tnt:measuredspeed>
      <tnt:vehiclecount>1</tnt:vehiclecount>
      <tnt:direction>Berlin</tnt:direction>
    </tnt:sensor>
  </tnt:observation>
</tnt:observations>
```

A.2 Taxi Company Example (DLR)

DLR provides pre-processed taxi data in XML form (for details see Section 2.1.1). For transformation to the TRACK&TRADE data format only the sub-elements of <FAHRT> (current taxi ride) which are listed in Table 6 are necessary. Table 7 lists the additional fields which must be filled out. An example data set for three measurement points is shown in Listing 11. The DLR data is converted to the TRACK&TRADE data format by a transformer developed within TRACK&TRADE.

Table 6. Mapping of DLR Data Format to T&T Data Format.

DLR Format <FAHRT>	T&T Format	Note
ID	OBSERVATION:SRCID	
STATUS	OBSERVATION:STATUS	One to one mapping, no translation of the status code necessary. See also Table 3 for a list of status codes.
ZEITPUNKT *(timestamp)*	ACTUAL:TS	Must be converted to UTC.
SEKUNDEN *(seconds)*	FCD:DURATION	In seconds.
ABFAHRT *(departure)*	FCD:POS_0	Start position.
ZIEL *(arrival)*	ACTUAL:POSITION	End position.

Table 7. Additional Data to be Filled out by DLR.

T&T Format	Content	Note
OBSERVATIONS:COMPANYID	="DLR"	
TRAFFICSRC:VEHICLETYPE	="TAXI"	

Listing 11. Example Data Set for DLR.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<ns1:observations xmlns:ns1="http://tnt.trackandtrade.org/schema"
xmlns:ns2="http://www.opengis.net/gml" xmlns:ns3="http://www.w3.org/1999/xlink"
xmlns:ns4="http://www.w3.org/2001/SMIL20/" xmlns:ns5="http://www.w3.org/2001/SMIL20/Language"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ns1:companyID="dlr"
xsi:schemaLocation="http://tnt.trackandtrade.org/schema TNT_gmlpos.xsd">

    <ns1:observation ns1:srcID="6801012">

        <ns1:status>90</ns1:status>

        <ns1:actual>

            <ns1:ts>2007-07-07T02:45:11+02:00</ns1:ts>

            <ns1:position>

                <ns2:pos srsName="urn:ogc:def:crs:EPSG:6.6:4326">52.4843505859
13.2938659668</ns2:pos>

            </ns1:position>

        </ns1:actual>

        <ns1:fcd>

            <ns1:position_0>

                <ns2:pos srsName="urn:ogc:def:crs:EPSG:6.6:4326">52.488264974
13.303499349</ns2:pos>

            </ns1:position_0>

            <ns1:duration>90</ns1:duration>

        </ns1:fcd>

    </ns1:observation>

    <ns1:observation ns1:srcID="6801076">

        <ns1:status>70</ns1:status>

        <ns1:actual>

            <ns1:ts>2007-07-07T02:45:11+02:00</ns1:ts>

            <ns1:position>

                <ns2:pos srsName="urn:ogc:def:crs:EPSG:6.6:4326">52.519547526
13.4131327311</ns2:pos>

            </ns1:position>

        </ns1:actual>

        <ns1:fcd>

            <ns1:position_0>

                <ns2:pos srsName="urn:ogc:def:crs:EPSG:6.6:4326">52.5218343099
13.4160990397</ns2:pos>

            </ns1:position_0>

            <ns1:duration>20</ns1:duration>

        </ns1:fcd>

    </ns1:observation>

    <ns1:observation ns1:srcID="6801017">

        <ns1:status>70</ns1:status>

        <ns1:actual>

            <ns1:ts>2007-07-07T02:45:11+02:00</ns1:ts>

            <ns1:position>

                <ns2:pos srsName="urn:ogc:def:crs:EPSG:6.6:4326">52.4936523438
13.3097839355</ns2:pos>
```

```
            </ns1:position>
        </ns1:actual>
        <ns1:fcd>
            <ns1:position_0>
                <ns2:pos srsName="urn:ogc:def:crs:EPSG:6.6:4326">52.4950683594
13.3085489909</ns2:pos>
            </ns1:position_0>
            <ns1:duration>20</ns1:duration>
        </ns1:fcd>
    </ns1:observation>
</ns1:observations>
```