## DELIVERABLE

### D2.1 - Report on Data Infrastructure Architecture and Implementation Plan

<table>
<thead>
<tr>
<th>Dissemination level</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of Document</td>
<td>Report</td>
</tr>
<tr>
<td>Contractual date of delivery</td>
<td>M12 - 30/04/2016</td>
</tr>
<tr>
<td>Actual Date of Delivery</td>
<td>M12 - 30/04/2016</td>
</tr>
<tr>
<td>Deliverable Number</td>
<td>D2.1</td>
</tr>
<tr>
<td>Deliverable Name</td>
<td>Report on Data Infrastructure Architecture and Implementation Plan</td>
</tr>
<tr>
<td>Deliverable Leader</td>
<td>NISSA</td>
</tr>
<tr>
<td>Work package(s)</td>
<td>WP2</td>
</tr>
<tr>
<td>Status &amp; version</td>
<td>V2.0 – Final (revised)</td>
</tr>
<tr>
<td>Number of pages</td>
<td>89</td>
</tr>
<tr>
<td>WP contributing to the deliverable</td>
<td>WP2</td>
</tr>
<tr>
<td>WP / Task responsible</td>
<td>T2.1</td>
</tr>
<tr>
<td>Coordinator (name / contact)</td>
<td>INTRASOFT</td>
</tr>
<tr>
<td>Other Contributors</td>
<td>JSI, UNINOVA, ICCS, EP, ADRIA, LPP, BCC, LS, AIT, UoW, TIS, TREDIT, FLU, KAPSCH, UoA</td>
</tr>
<tr>
<td>EC Project Officer</td>
<td>Mauritsch Walter</td>
</tr>
<tr>
<td>Keywords:</td>
<td>Data sources, Data Infrastructure, Implementation Plan</td>
</tr>
<tr>
<td>Abstract (few lines):</td>
<td>This deliverable is the basis for the development of the data infrastructure that will enable a continuous feeding of the Platform with the data from various sources. This process will encompass the gathering (availability) of</td>
</tr>
</tbody>
</table>
data from data sources and its preparation (cleaning) for the further processing. Detailed description of data sources in pilots is provided, indicating the complexity of the formats, ways of the access and the potential usage. The deliverable details the conceptual architecture of the data infrastructure and outlines the plan for the implementation of the envisioned data infrastructure.

<table>
<thead>
<tr>
<th>Document History</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Version</strong></td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.7</td>
</tr>
<tr>
<td>0.7</td>
</tr>
<tr>
<td>0.8</td>
</tr>
<tr>
<td>0.9</td>
</tr>
<tr>
<td>0.99</td>
</tr>
<tr>
<td>1.0</td>
</tr>
<tr>
<td>1.5</td>
</tr>
<tr>
<td>2.0</td>
</tr>
</tbody>
</table>

**Deliverable resubmission**

<table>
<thead>
<tr>
<th>EC Recommendation</th>
<th>Actions Performed / Explanations</th>
<th>Section in the doc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>The necessity of the discussion on pages 17-18 regarding updating frequency is not clear. These types of parameters should be programmable, configured.</td>
<td>This data feed is provided by a 3rd party &amp; therefore configuration of the frequency of the readings by us is not possible. Furthermore, the reading intervals vary for the different sensors and that is why a frequency analysis was performed.</td>
<td>The explanation is on the column on the left. Only a few updates on page 16.</td>
</tr>
<tr>
<td>Most relevant question is what are the functional needs and requirements.</td>
<td>Despite the fact that we cannot configure the frequency of data retrievals for this feed, the current arrangement meets our needs. For most of the sensors, we get data every 5 mins. This is sufficient for the operation of the forecasting engine, which has been designed to provide short-term forecasts for an 1-hour horizon using 5-min steps.</td>
<td>Same as above</td>
</tr>
</tbody>
</table>
Table of Contents

Deliverable resubmission........................................................................................................... 2

Executive summary....................................................................................................................... 7

1 Introduction.................................................................................................................................. 8

   1.1 Objectives of the Deliverable................................................................................................. 8

   1.2 Structure of the Deliverable .................................................................................................... 8

2 Analysis of available data sources.............................................................................................. 9

   2.1 Proactive improvement of transport systems quality and efficiency (West Midlands pilot) .... 9

      2.1.1 Short description................................................................................................................. 9

      2.1.2 Data sources (with focus on speed and volume)................................................................. 9

      2.1.3 Existing infrastructure, APIs and protocols (with focus on scale and reliability) ..... 26

   2.2 Proactive improvement of transport systems quality and efficiency (Pilot Austria) ....... 28

   2.3 Proactive improvement of transport systems quality and efficiency (Pilot Ljubljana) .... 29

      2.3.1 Short description................................................................................................................. 29

      2.3.2 Data sources (with focus on speed and volume)................................................................. 29

      2.3.3 Existing infrastructure, APIs and protocols...................................................................... 29

      2.3.4 Proposed infrastructure..................................................................................................... 30

   2.4 Proactive charging schemes for freight transport (Portuguese Pilot).................................... 30

      2.4.1 Short description................................................................................................................. 30

      2.4.2 Data sources (with focus on speed and volume)................................................................. 31

      2.4.3 Existing infrastructure, APIs and protocols (with focus on scale and reliability) ..... 54

   2.5 Integrated Car2X communication platform........................................................................... 55

      2.5.1 Short description................................................................................................................. 55

      2.5.2 Data sources (with focus on speed and volume)................................................................. 56

   2.6 General data sources ............................................................................................................ 60

      2.6.1 Short description................................................................................................................. 60

      2.6.2 Data sources....................................................................................................................... 60

3 Proposed data infrastructure – conceptual design................................................................. 73
3.1 Introduction – the need for Edge processing .......................................................... 73
3.2 The benefits of Edge processing ............................................................................. 74
3.3 Edge processing - elements ................................................................................... 74
  3.3.1 Layers .............................................................................................................. 75
  3.3.2 Edge cloud ...................................................................................................... 77
3.4 Data infrastructure: Conceptual architecture ......................................................... 78
3.5 Data retrievers, watchdog ...................................................................................... 79
3.6 Support for standard big data paradigm ............................................................... 80
4 Implementation plan ................................................................................................. 82
5 Conclusion .................................................................................................................. 89

Table of Figures

Figure 1: Sample Flow Sensor Reading – JSON Format ................................................... 9
Figure 2: Sample Average Speed Sensor Reading – JSON Format ................................. 10
Figure 3: BCC Data Streaming and Processing Adaptor .................................................. 11
Figure 4: Sample Transformed Flow Reading – JSON Format ....................................... 11
Figure 5: Sample Transformed Average Speed Reading – JSON Format ....................... 12
Figure 6: Number of Updated Flow Measurements per 2-Minute Request during the Experimental Period (7 Days) .................................................... 13
Figure 7: Total Number of Updated Flow Measurements during the Experimental Period (7 Days) .......................................................... 13
Figure 8: Number of Updated Average Speed Measurements per 2-Minute Request during the Experimental Period (7 Days) .................................................... 14
Figure 9: Total Number of Updated Average Speed Measurements during the Experimental Period (7 Days) .......................................................... 14
Figure 10: Number of Unique Sensors Used to Update Each Measurement Type ............ 15
Figure 11: Frequency of Sensor Updates for Flow Measurements ............................... 16
Figure 12: Frequency of Sensor Updates for Average Speed Measurements ............... 17
Figure 13: Frequency of Sensor Updates for Travel Time Measurements .................... 17
Figure 14: Histogram (a) and Boxplot (b) of the Updating Time Intervals of BCC Sensor Types .......................................................... 18
Figure 15: Sample Tweet in JSON Format ................................................................. 23
Figure 16: Cumulative Number of Tweets as Streamed from Three Bounding Boxes ......................................................................................... 24
Figure 17: Frequency of 1000-Tweet Time Intervals in UK Stream (A), and 100-Tweet time intervals in West Midlands Stream (b) .......................................................... 25
Figure 18: Pilot’s Highways .......................................................................................... 31
Figure 19: Pilot’s Highway Counters Provided by Infraestruturas de Portugal ............... 32
Figure 20: Database Dump with Counter Data ............................................................ 33
Figure 21: IP Counter Sensors Adaptor ........................................................................ 35
Figure 22: Sample Transformed Counter Metadata – BSON Format ............................ 36
Figure 23: Sample Transformed Counter Reading – BSON Format ............................. 36
Figure 24: Data Quality Test Sample .......................................................................... 37
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MEASUREMENT TYPES, STREAMING INTERVALS, AND NO OF RECORDS CONSIDERED</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>MEASUREMENTS VOLUME AND SIZE INFORMATION</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>EXPERIMENTAL STREAMING INTERVALS AND NUMBER OF TWEETS</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>STORAGE REQUIREMENTS IN THE CENTRAL REPOSITORY FOR THE WEST MIDLANDS TWEETS</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>CSV EXAMPLE FOR HIGHWAY COUNTER DATA</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>SENSOR METADATA SAMPLE</td>
<td>34</td>
</tr>
<tr>
<td>7</td>
<td>TRAFFIC EVENTS SQL DUMP SAMPLE</td>
<td>38</td>
</tr>
<tr>
<td>8</td>
<td>VIAVILRE TOLL SENSOR DATA SAMPLE</td>
<td>45</td>
</tr>
<tr>
<td>9</td>
<td>ASCENDI TOLL SENSOR DATA SAMPLE</td>
<td>45</td>
</tr>
<tr>
<td>10</td>
<td>TRUCK EVENTS AND GPS TRACKS SAMPLE DATA</td>
<td>50</td>
</tr>
<tr>
<td>11</td>
<td>TRUCK CONSUMPTION DATA SAMPLE</td>
<td>51</td>
</tr>
<tr>
<td>12</td>
<td>NUMBER OF TRUCK DATA READINGS BETWEEN 01/04/2015 AND 7/12/2015</td>
<td>53</td>
</tr>
<tr>
<td>13</td>
<td>T2.2 TIMELINE</td>
<td>83</td>
</tr>
<tr>
<td>14</td>
<td>T2.3 TIMELINE</td>
<td>84</td>
</tr>
<tr>
<td>15</td>
<td>T2.4 TIMELINE</td>
<td>85</td>
</tr>
<tr>
<td>16</td>
<td>T2.5 TIMELINE</td>
<td>86</td>
</tr>
</tbody>
</table>
## Definitions, Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITS</td>
<td>Intelligent Transportation Systems</td>
</tr>
<tr>
<td>CO2</td>
<td>Carbon dioxide</td>
</tr>
<tr>
<td>GNSS</td>
<td>Global Navigation Satellite System</td>
</tr>
<tr>
<td>OODA</td>
<td>Observe, Orient, Decide and Act</td>
</tr>
<tr>
<td>Dx</td>
<td>Deliverable (where x defines the deliverable identification number e.g. D1.1)</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communications Technology</td>
</tr>
<tr>
<td>TIS</td>
<td>Traveller Information Services</td>
</tr>
<tr>
<td>ETL</td>
<td>Extract-Transform-Load</td>
</tr>
<tr>
<td>BI</td>
<td>Business Intelligence</td>
</tr>
<tr>
<td>MDM</td>
<td>Multi-Dimensional Data Modelling</td>
</tr>
<tr>
<td>CRISP-DM</td>
<td>CRoss Industry Standard Process for Data Mining</td>
</tr>
<tr>
<td>KDD</td>
<td>Knowledge Discovery Databases</td>
</tr>
<tr>
<td>SEMMA</td>
<td>Sample, Explore, Modify, Model, Assess</td>
</tr>
<tr>
<td>PAM</td>
<td>Partitioning Around Medoids</td>
</tr>
<tr>
<td>FAMES</td>
<td>FAsst MEDoid Selection</td>
</tr>
<tr>
<td>ARIMA</td>
<td>Autoregressive Integrated Moving Average</td>
</tr>
<tr>
<td>SARIMA</td>
<td>Seasonal ARIMA</td>
</tr>
<tr>
<td>ANNs</td>
<td>Artificial Neural Networks</td>
</tr>
<tr>
<td>SVR</td>
<td>Support Vector Regression</td>
</tr>
<tr>
<td>CCZ</td>
<td>Congestion Charging Zone</td>
</tr>
<tr>
<td>ANPR</td>
<td>Automatic Number Plate Recognition</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>OBU</td>
<td>On-Board Unit</td>
</tr>
<tr>
<td>TrEPS</td>
<td>Traffic Estimation and Prediction tool</td>
</tr>
<tr>
<td>ERP</td>
<td>Electronic Road Pricing</td>
</tr>
<tr>
<td>IUs</td>
<td>In-vehicle Units</td>
</tr>
<tr>
<td>MOT</td>
<td>Mode Of Transportation</td>
</tr>
<tr>
<td>TERN</td>
<td>Trans-European Road Network</td>
</tr>
</tbody>
</table>
Executive summary

This deliverable is the basis for the development of the data infrastructure that will enable a continuous feeding of the Platform with the data from various sources. This process will encompass the gathering (availability) of data from data sources and its preparation (cleaning) for the further processing. Since the data will be originating from a very broad set of data sources, the infrastructure will include the big data technologies in order to ensure performances/scalability and reliability of the system.

Detailed description of data sources in pilots is provided, indicating the complexity of the formats, ways of the access and the potential usage. This information provides the requirements for the data infrastructure, which are critical for the operation of pilots as well as for the exploitation of the Platform as a whole.

The data infrastructure should support the need that data is analysed in near real time in order to ensure rapid execution and effect change. There are many situations, esp. on the crowded roads, that require immediate actions to prevent problems. Therefore, a special attention is given to the need for complex local processing that should happen on the edge (local/embedded devices, gateways) in order to increase the reactivity of the data processing. This creates a new set of requirements for supporting this type of local real-time actions, enabling the realization of the full potential of the underlying data sources, by ensuring the efficiency of processing and protecting the privacy of collected data.

The deliverable outlines the plan for the implementation of the envisioned data infrastructure.

The deliverable is related to D1.3 that provides detailed description of available data sources, whereas this deliverable focuses on the advanced concepts related to the data gathering infrastructure.
1 Introduction

1.1 Objectives of the Deliverable
The main goal of this deliverable is to support the implementation of the data infrastructure that should ensure continuous feeding of the Platform with the data from various sources. It is realized through three objectives: 1) the presentation of the characteristics of the available data sources in existing pilots, 2) the design of the data infrastructure architecture and 3) the draft of the implementation plan for the realization of the data infrastructure.

The main contribution is the definition of the process how the data can be collected in order to be processed by the Platform.

From the novelty point of view, the architecture creates the possibility for processing the data close to the data sources (so called edge processing), opening the opportunities to gain more insight and value from (IoT) data and providing more real-time awareness/reactivity in data processing. Such an architecture requires that analytics is embedded both directly within gateways such as mobile phones, but also endpoints as sensors, controllers, equipment. Collecting and analyzing data close to the endpoints means that action can take place locally in real or near-real time. In this way, only meaningful information needs to be backhauled to the datacenter or cloud for storage, benchmarking or advanced statistical analysis. Additionally, privacy of data will be better protected by ensuring that the private data will be processed locally.

1.2 Structure of the Deliverable
The deliverable is structured in the following way:

- Section 3 provides a comprehensive analysis of the data sources available in pilots
- Section 4 elaborates the conceptual design of the data infrastructure
- Section 5 provides a draft of the plan for the implementation and testing
2 Analysis of available data sources

2.1 Proactive improvement of transport systems quality and efficiency (West Midlands pilot)

2.1.1 Short description
The Birmingham pilot will utilise data from a heterogeneous set of sources. The data that will require intensive processing and analysis will be those related to traffic and social media. The characteristics of these streams are discussed in the sections below. Other data sources include, public transport data (network, timetables and real time information related to PT services), bicycle routes, air quality and weather data. For the latter descriptions have been presented in D1.3.

2.1.2 Data sources (with focus on speed and volume)

2.1.2.1 Birmingham City Council (BCC) Open Data Sources
Birmingham City Council (BCC) publishes various sets of open data for the public to download and reuse\(^1\). Recently, BCC has started to publish traffic-related data\(^2\) through Representational State Transfer (REST) endpoint calls. These include sensor readings of flows, average speeds, congestions, occupancies, travel times, incidents, parking, and road works.

**Raw Data Examples**

```json
- {
    Type: "Detector",
    Description: "12A -A449 I/B Nr Vicarage Rd",
    Northing: 295523,
    Value: {
        Status: "green",
        Percent: {
            Value: 24,
            content: 0,
            Threshold: 0
        },
        Trend: "falling",
        Level: 24
    },
    Easting: 388835,
    LastUpdated: "2016-03-09 09:45:00",
    SCN: "WMID-JTMS21"
}
```

Figure 1: Sample Flow Sensor Reading – JSON Format

---

\(^1\) [http://www.birmingham.gov.uk/open-data](http://www.birmingham.gov.uk/open-data)

\(^2\) [http://butc.opendata.onl/AL_OpenData](http://butc.opendata.onl/AL_OpenData)
Data Streaming Adaptor:

A data streaming and processing adaptor has been implemented in Python\(^3\) to perform the following functions:

- **PushBccDataToMongoFirstTime**: utility to establish the database connections, create the database collections, and insert the data sets for the first time into an empty data collection in MongoDB\(^4\) central repository.

- **PushBccTrafficToOptimumCollections**: utility to update the datastore periodically with newly updated records. This sub-component performs REST endpoint calls to the BCC Open Data Sources Endpoints every two minutes, considers the updated records, build the transformed JSON format for each updated record, and insert the updated records into the data store.

- **ENtoLL**: utility to convert geolocation data coordinates from the British National Grid (BNG)\(^5\), formerly known as the National Grid Reference (NGR), to latitude and longitude (lat/long WGS 84).

---

\(^3\) [https://www.python.org/](https://www.python.org/)

\(^4\) More information related to the data storage can be found in the deliverable 1.4

\(^5\) [http://sewhgpgc.co.uk/os.php](http://sewhgpgc.co.uk/os.php)
Figure 3: BCC Data Streaming and Processing Adaptor

Transformed Data Examples

```
{
    "id": ObjectId("56dfe6f58e5f0c60e2d7fa36d"),
    "status": "green",
    "measurement_datetime": ISODate("2016-03-09T10:05:00.000Z"),
    "lane": "all",
    "insertion_datetime": ISODate("2016-03-09T10:06:07.492Z"),
    "description": "A41 Warwick Rd; Hampton Lane",
    "loc": [  
        "type": "Point",
        "coordinates": [  
            -1.7681943644496707,
            52.4134881763253020
        ]
    ],
    "trend": "falling",
    "value": 10,
    "measurement_id": "WMTG-G0113_TCAM2-Zone3",
    "VehicleCategory": "all",
    "typeSensor": "Detector",
    "type": "Flow"
}
```

Figure 4: Sample Transformed Flow Reading – JSON Format
Data Metrics

An experiment based on a selected streaming snapshot of seven days from the streaming logs and the transformed data store was conducted to conduct a preliminary investigation of Traffic-Related data sets as live streamed from Birmingham City Council Open Data Sources. Initial analysis was performed to estimate the following data metrics:

- Incremental and Cumulative Data Streaming Rates
- Data Storage Requirements
- Distribution of Sensors by Measurement Types
- Frequency of Measurement Updates by Sensors
- Distribution of Sensor Updating Intervals

Experimental Data Information

Table 1: Measurement Types, Streaming Intervals, and No of Records considered

<table>
<thead>
<tr>
<th>Measurement Type</th>
<th>Start Date</th>
<th>Start Time</th>
<th>End Date</th>
<th>End Time</th>
<th>Total Number of Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flows</td>
<td>2016-02-26</td>
<td>00:00:00</td>
<td>2016-03-03</td>
<td>23:58:04</td>
<td>699639</td>
</tr>
<tr>
<td>Average Speeds</td>
<td>2016-02-26</td>
<td>00:00:04</td>
<td>2016-03-03</td>
<td>23:58:03</td>
<td>1522394</td>
</tr>
<tr>
<td>Travel Times</td>
<td>2016-02-26</td>
<td>00:00:02</td>
<td>2016-03-03</td>
<td>23:58:06</td>
<td>726684</td>
</tr>
<tr>
<td>Congestions</td>
<td>2016-02-26</td>
<td>00:00:00</td>
<td>2016-03-03</td>
<td>23:58:00</td>
<td>83165</td>
</tr>
<tr>
<td>Occupancies</td>
<td>2016-02-26</td>
<td>00:00:05</td>
<td>2016-03-03</td>
<td>23:58:07</td>
<td>2523</td>
</tr>
</tbody>
</table>
Incremental and Cumulative Data Streaming Rates

REST endpoint requests are scheduled to execute every two minutes. The number of updated measurements per request (incremental streaming) as well as the cumulative updated measurements processed and stored into the central repository have been analysed for flows (Figure 6 & Figure 7) and average speeds (Figure 8 & Figure 9).

Figure 6: Number of updated flow measurements per 2-minute request during the experimental period (7 days)

Figure 7: Total number of updated flow measurements during the experimental period (7 days)
Figure 8: Number of updated average speed measurements per 2-minute request during the experimental period (7 days)

Figure 9: Total number of updated average speed measurements during the experimental period (7 days)
Data Storage Requirements

Table 2: Measurements Volume and Size Information

<table>
<thead>
<tr>
<th>Measurement Type</th>
<th>Total Measurements</th>
<th>Average Daily Measurements</th>
<th>Average Hourly Measurements</th>
<th>Average Object Size (Bytes)</th>
<th>Daily Storage Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flows</td>
<td>699639</td>
<td>99948</td>
<td>4164</td>
<td>1414.261</td>
<td>134.8</td>
</tr>
<tr>
<td>Average Speeds</td>
<td>1522334</td>
<td>217476</td>
<td>9062</td>
<td>1414.261</td>
<td>293.32</td>
</tr>
<tr>
<td>Travel Times</td>
<td>726684</td>
<td>103812</td>
<td>4326</td>
<td>1414.261</td>
<td>140.02</td>
</tr>
<tr>
<td>Congestions</td>
<td>83165</td>
<td>11881</td>
<td>495</td>
<td>1414.261</td>
<td>16.02</td>
</tr>
<tr>
<td>Occupancies</td>
<td>35618</td>
<td>5088</td>
<td>212</td>
<td>1414.261</td>
<td>6.86</td>
</tr>
<tr>
<td>Total</td>
<td>3067440</td>
<td>438205</td>
<td>18259</td>
<td>1414.261</td>
<td>591.02</td>
</tr>
</tbody>
</table>

Notes:
- The Average Object Size is the average size of each document stored in the MongoDB collection (roadSensorValue) in bytes. This was computed as the `fileSize` divided by the number of documents.
- The fileSize is total size in bytes of the data files that hold the MongoDB database (BccTraffic). This value includes preallocated space and the padding factor.

Unique Sensors

![Figure 10: Number of unique sensors used to update each measurement type](image)

Figure 10: Number of unique sensors used to update each measurement type
Frequency of Sensor Updates

In order to be able to better assess the requirements of the data infrastructure we analysed some of available (at the beginning of the project) real-time data-sets regarding the frequencies of their data updates (data throughput of the sensors). We picked the inductive loop sensors from BCC, which have the biggest throughput among the anticipated sensor types. These results can be then used to have some initial idea of maximum data infrastructure throughput needs for sensor measurements. Independently of the updating frequency of the sensor, we can control and re-sample the values at the retrieve time, or later. This will be parameterized and will allow the platform to control and limit the data-flow, while still having enough information to be able to fulfil the requirements.  

The number of updated measurements for each measurement type has been aggregated by the sensor Ids across the period of the experiment. Contingency plots (Figure 11, Figure 12 and Figure 13) have then been produced to depict how many sensors (y-axis) have updated how many measurements (x-axis) for flow, average speed, and travel time readings, respectively.

Figure 11: Frequency of Sensor Updates for Flow Measurements

---

6 The section above was added as an additional explanation according to the EC Review comments. At the time of writing we already know all the throughputs and also have re-sampling controls in place. There was no need to throttle the data at retrieve time, but only at modelling/retrieve time for some uses.
Distribution of Sensor Updating Intervals

To obtain a preliminary idea about the most frequent updating intervals for BCC sensors, the most active 20 sensors for each sensor type (Detector, Scoot, and ANPR) during the 7-day experiment period have been fetched with their measurement timestamps. The time intervals
between the measurement timestamps have been calculated and depicted for the three sensor types using the histogram in Figure 14a and box plot in Figure 14b.

![Figure 14: Histogram (a) and Boxplot (b) of the updating time intervals of BCC sensor types](image)

### 2.1.2.2 National Traffic Information Service (NTIS)

The National Traffic Information Service (NTIS) can provide both real-time and historic data services ([http://www.trafficengland.com/services-info](http://www.trafficengland.com/services-info)). NTIS provides web services to push these data to each subscriber in real time. ([https://github.com/ntisservices/ntis-java-web-services-Release2.5](https://github.com/ntisservices/ntis-java-web-services-Release2.5)). Currently we are receiving real time data from sensors of type MIDAS, TMU, Fused Sensor, totally from 1000 sensors. ANPR sensors are moving to a new system, we are going to collect also from ANPR the next days. The data is available in DATEX II format which is a European XML standard for traffic information.

An example of data from MIDAS type of Sensor, sent by NTIS is presented in the following:

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header />
  <soapenv:Body>
    <d2lm:d2LogicalModel xmlns:d2lm="http://datex2.eu/schema/2/2_0" modelBaseVersion="2" extensionName="NTIS Published Services" extensionVersion="2.0">
      <d2lm:exchange>
        <d2lm:supplierIdentification>
          <d2lm:country>gb</d2lm:country>
          <d2lm:nationalIdentifier>NTIS</d2lm:nationalIdentifier>
        </d2lm:supplierIdentification>
      </d2lm:exchange>
      <d2lm:payloadPublication xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="d2lm:MeasuredDataPublication" lang="en">
        <d2lm:feedType>MIDAS Loop Traffic Data</d2lm:feedType>
        <d2lm:publicationTime>2013-07-10T14:59:10.421+01:00</d2lm:publicationTime>
        <d2lm:publicationCreator>
          <d2lm:country>gb</d2lm:country>
          <d2lm:nationalIdentifier>NTIS</d2lm:nationalIdentifier>
        </d2lm:publicationCreator>
        <d2lm:measurementSiteTableReference targetClass="MeasurementSiteTable" version="56.0" id="NTIS_MIDAS_Measurement_Sites" />
        <d2lm:headerInformation>
          <d2lm:confidentiality>restrictedToAuthoritiesTrafficOperatorsAndPublishers</d2lm:confidentiality>
          <d2lm:informationStatus>real</d2lm:informationStatus>
          <d2lm:urgency>normalUrgency</d2lm:urgency>
        </d2lm:headerInformation>
        <d2lm:siteMeasurements>
          <d2lm:measurementSiteReference targetClass="MeasurementSiteRecord" version="56.0" id="NTIS_MIDAS_Measurement_Sites" />
          <d2lm:measurementTimeDefault>2014-02-05T14:58:23.001Z</d2lm:measurementTimeDefault>
          <d2lm:measuredValue index="0">
            <d2lm:measuredValue>
              <d2lm:basicData xsi:type="d2lm:TrafficSpeed">
                <d2lm:averageVehicleSpeed>...
```
For each source we define:

- **Frequency of generating the data**
  
  Per 1 min for MIDAS and Fused Sensor and per 5 min for TMU and ANPR.

- **Size (amount) of data**
  
  8 Giga storage per day in MongoDB

- **Access to data** (free/price, protocol, access method: e.g. pool)
  
  Access to data is free and takes place through web services. The requirement is to accept the terms and conditions from NTIS.

- **Storage of the data** (no, locally, globally)
  
  Currently data are stored in MongoDB using Java driver.

- **Processing the data** (locally or globally).

The processing of the data involves two basic steps. In the first step through web services we take data and create a structure (document in MongoDB) that has some basic information and put it in a collection in the database (initialNTIS). At the second step the information from these documents is used along with the information of the static NTIS model to create two new structures (documents) and put them in two new collections (NTISroadSensorValue and NTISroadSensor) in the database. The connection between these sources happen through the measurementId of the initial document. This static model is provided by NTIS in xml files. The model is stored in two collections (MeasurementsSites, PredefinedLocations) in the database. If the creation of the new documents is successful, the document from the collection initialNTIS is deleted. If there was an issue in this transformation from the initial document to the new documents, an error or if the measurementId was not found in the NTIS static model, the initial document is stored in another collection named nonUpdated and the document is deleted from the initialNTIS collection. The second step is executed from a script in java that is running every 2 minutes via a cron job.

The static model of NTIS is updated approximately fortnightly. The model can be downloaded manually from the website of NTIS or received programmatically using an Http Get request. The file is a zip file with three xml files. A notification is also sent from NTIS through web services. When this notification is received an Http Get request takes place, getting the model in a zip file, store it locally, unzip it and for each xml create a new collection (drop the previous one and rename the new one).

```json
{  
  "measurementId": "ECB728B94BC80656E0438DC611AC34B2",
  "typeSensor": "MIDAS",
}
```
D2.1 - Report on Data Infrastructure Architecture and Implementation Plan
2.0, 10/04/2017

"type": "speed",
"index": 0,
"updated": "false",
"value": 0,
"datetime": ISODate("2016-03-07T14:09:00Z"),
"dataError": "false"
}

"3B08CC4F56504767BC7F9A4EB99A410D",
"type": "flow",
"typeSensor": "raw",
"datetime": ISODate(2015-04-29T09:38:00),
"value": 66,
"lane": "lane1",
"VehicleCategory": "A",
"dataError": "true",
"loc": {
  "type": "Point",
  "coordinates": [51.609920956754]
}

Following diagram illustrates the data flow of the process
2.1.2.3 Twitter

Twitter posts, or ‘tweets’, are the basic atomic building block of Twitter data. Tweets, also known more generically as “status updates.” Tweets can be embedded, replied to, liked, unliked and deleted. Data collectors of Tweets should tolerate the addition of new fields and variance in ordering of fields with ease. Not all fields appear in all contexts. It is generally safe to consider a nulled field, an empty set, and the absence of a field as the same thing. Please note that Tweets found in Search results vary somewhat in structure from this document.

Raw Data Example

Twitter Streaming API provides posts, known as “tweets”, in JSON data format. Figure 15 depicts a sample tweet as streamed based on a locations filter within the boundary of West Midlands, UK:

```json
{
  "id": "56d5c0ff320575c880d95f6d",
  "contributors": null,
  "truncated": false,
  "text": "IndieOver40 @nowcanapply It's Mrs K's favourite band will let her have the choice this week for the Welsh special. #!",
  "is_quote_status": false,
  "in_reply_to_status_id": 7044118054986974288,
  "id": 704442560743548768,
  "source": "@Twitter for Android/s",
  "retweeted": false,
  "coordinates": null,
  "timestamp_ms": "156849145168",
  "entities": {
    "user_mentions": [
      {
        "id": 2388739868,
        "indices": [23, 29]
      }
    ],
    "id_str": "2388739868",
    "screen_name": "IndieOver40",
    "name": "EverythingIndieOver40"
  }
}
```
Data Metrics

An experiment based on a selected streaming snapshot based on the Twitter public streaming API was conducted to obtain a preliminary estimation of streaming rates and storage requirements for three Twitter streaming pipelines:

- Tweets streamed from the United Kingdom
- Tweets streamed from the region of West Midlands
- Tweets streamed from the city of Birmingham
Experimental Data Information

Table 3: Experimental Streaming Intervals and Number of Tweets

<table>
<thead>
<tr>
<th>Source</th>
<th>Start Date</th>
<th>Start Time</th>
<th>End Date</th>
<th>End Time</th>
<th>Number of Tweets</th>
</tr>
</thead>
<tbody>
<tr>
<td>United Kingdom</td>
<td>2016-02-12</td>
<td>16:22:19</td>
<td>2016-02-12</td>
<td>22:34:16</td>
<td>132000</td>
</tr>
<tr>
<td>West Midlands</td>
<td>2016-02-12</td>
<td>15:17:51</td>
<td>2016-02-14</td>
<td>11:56:20</td>
<td>116800</td>
</tr>
<tr>
<td>Birmingham City</td>
<td>2016-02-12</td>
<td>15:18:44</td>
<td>2016-02-15</td>
<td>08:46:50</td>
<td>54600</td>
</tr>
</tbody>
</table>

Streaming Rates

Figure 16: depicts the streaming rates in seconds, minutes, and hours for tweets posted from within the United Kingdom (a), the region of West Midlands (b), and the city of Birmingham (c), respectively, as filtered by the API.

Streaming Intervals

Figure 17 depicts the time intervals in minutes to obtain 1000 tweets from the United Kingdom (a), as well as to obtain 100 tweets from West Midlands (b). On average, 1000 tweets from the UK are streamed every 3 minutes, whereas it takes 2 minutes to stream 100 tweets from the West Midlands region, with some longer intervals, as can be seen by the right skewed histogram in Figure 17b, due to lower volumes of tweets during the night hours.
Data Storage Requirements

For the research objectives of the Optimum project, an on-going West Midlands Twitter streaming pipeline has been set up to collect the tweets posted from the region of West Midlands. These will undergo further text analytics to extract and correlate Twitter features with traffic attributes to support the Optimum forecasting engine component. Storage requirements are estimated in Table 4.

Table 4: Storage Requirements in the central repository for the West Midlands tweets

<table>
<thead>
<tr>
<th>Metric</th>
<th>Estimated Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tweets per min</td>
<td>52</td>
</tr>
<tr>
<td>Tweets per hr</td>
<td>3120</td>
</tr>
<tr>
<td>Tweets per day</td>
<td>74880</td>
</tr>
<tr>
<td>Avg Tweet Size in Bytes (mongoDB)</td>
<td>4533</td>
</tr>
<tr>
<td>Avg daily Size in GB (mongoDB)</td>
<td>0.316119790077209</td>
</tr>
<tr>
<td>Avg daily Size in MB (mongoDB)</td>
<td>323.706665039062</td>
</tr>
</tbody>
</table>
2.1.3 Existing infrastructure, APIs and protocols (with focus on scale and reliability)

2.1.3.1 Birmingham City Council (BCC) Open Data Source API

Direct Data Access Mechanism:

In Direct Data Access, data sources / datasets are named and grouped by suitable class names (e.g. Flow, AverageSpeed, TravelTime) which are further grouped by authority giving a three tier model. The authority level caters for the data sources available from a particular government authority such as a local council (e.g. BCC) or traffic authority.

To access a data set where its name is known (e.g. Flow) and the name is unique on the server across all supported authorities and classes, then this is simply achieved through a REST request of the form \texttt{http(s)://<server>/<DataSetName>.<DataType>}

For example, if there is a data set for car occupancy data called 'Flow', where its updated measurements are provided periodically by the server 'butc.opendata.onl/AL_OpenData' and it is required to retrieve it in JSON format, then the URL to use in the REST endpoint call would be \texttt{http://butc.opendata.onl/AL_OpenData/Flow.json}. Data sets may be customised via parameters in which case the required parameters and values can be added to the endpoint call as additional URL arguments.

Data Security and User Context

Not all data sets may be for public consumption and legal constraints may limit access to both data sets and items within data sets. This includes traffic-related data streams, which are not accessible through REST endpoint calls from outside the UK.

Available Data Formats

The data is available for download from the Open Data Sources Website\(^7\) with any of the following data formats:

- XML – eXtendible Markup Language.
- JSON – a form of XML intended for App integration.
- Datex 2 – a form of XML adhering to the standards for data interchange of the EU. Note: At the moment, only UTMC Parking data is available in Datex 2 format.
- CSV – Comma Separated Values.
- XLS(X) – Microsoft Excel formats with XLSX being the most up to date format.

\(^7\) \url{http://butc.opendata.onl/AL_OpenData}
Resource Information

- **Resource URL**: http://butc.opendata.onl/AL_OpenData/<DataSetName>.json
- **Response formats**: JSON
- **Requires authentication?** No
- **Rate limited?** No
- **Requires Parameters?** No
- **Example Request**: http://butc.opendata.onl/AL_OpenData/Flow.json

### 2.1.3.2 National Traffic Information Service (NTIS) API

- Provide details about available programming interfaces
  - Web Services wsdl, ([https://github.com/ntisservices/ntis-java-web-services-Release2.5](https://github.com/ntisservices/ntis-java-web-services-Release2.5))
  - NTIS
- Provide details about used adapters and brokers, if any
- Watchdog

### 2.1.3.3 Twitter Streaming API

The Twitter Streaming API\(^8\) gives developers low latency access to Twitter\(^9\) global stream of Tweet data. A proper implementation of a streaming client will be pushed messages indicating Tweets and other events have occurred, without any of the overhead associated with polling a REST endpoint. For the purpose of collecting *filtered* live tweets, we consider the use of the POST statuses / filter\(^10\) streaming endpoint, which provides streams of the public data flowing through Twitter. Suitable for following specific users or topics, and data mining them. Several filtration criteria may be applied. These include filtering tweets that:

- Were posted within a geographically predefined region, or
- Were by predefined Twitter accounts, or
- Contained predefined set of key-words and/or hashtags.

Resource Information

- **Resource URL**: https://stream.twitter.com/1.1/statuses/filter.json
- **Response formats**: JSON
- **Requires authentication?** Yes (user context only)
- **Rate limited?** Yes

---

\(^8\) [https://dev.twitter.com/streaming/overview](https://dev.twitter.com/streaming/overview)

\(^9\) [https://twitter.com/](https://twitter.com/)

\(^10\) [https://dev.twitter.com/streaming/reference/post/statuses/filter](https://dev.twitter.com/streaming/reference/post/statuses/filter)
- **Requires Parameters?** Yes (At least one predicate parameter must be specified)
- **Example Request:** [https://stream.twitter.com/1.1/statuses/filter.json?track=twitter](https://stream.twitter.com/1.1/statuses/filter.json?track=twitter)

**Considered Request Parameters**

For our filtration purposes from live Twitter public streams, we aim to apply the following request parameters:

- **locations:** Specifies a set of bounding boxes to track. This includes a comma-separated list of longitude, latitude pairs specifying a set of bounding boxes to filter Tweets by. Only geo-located Twitter posts falling within the requested bounding boxes will be included. Each bounding box should be specified as a pair of longitude and latitude pairs, with the southwest corner of the bounding box coming first. For example:

<table>
<thead>
<tr>
<th>Parameter Value</th>
<th>Track Twitter posted from...</th>
</tr>
</thead>
<tbody>
<tr>
<td>[-2.033,52.381,-1.748,52.607]</td>
<td>Birmingham, UK</td>
</tr>
<tr>
<td>[-3.236,51.825,-1.172,53.226]</td>
<td>West Midlands, UK</td>
</tr>
</tbody>
</table>

- **follow:** A comma separated list of unprotected user IDs, indicating the users to return statuses for in the stream. The stream will contain Twitter posts that are
  - created by the user
  - retweeted by the user
  - replies to any posts created by the user
  - retweets of any posts created by the user

- **track:** A comma-separated list of phrases, key-words, or hashtags which will be used to determine what Tweets will be delivered on the stream. A phrase may be one or more terms separated by spaces, and a phrase will match if all of the terms in the phrase are present in the Tweet, regardless of order and ignoring case.

2.2 **Proactive improvement of transport systems quality and efficiency (Pilot Austria)**

The detailed description of the nature of the relevant data sources is given in the deliverable D1.3. Since all data sources are publically available and the services for accessing them are detailed in D1.3, we provide here only some indications about the performances relevant for the design of the data infrastructure:

- Computing a route through Vienna takes 38ms (for car only) in the single threaded implementation and 21ms for the dual threaded implementation.

- Computing an intermodal route through Berlin takes 1.7s (for public transport, walking, bike). (The public transport network in Berlin is quite tough...).
• Memory Usage for the network only (car, walking, bike) is for whole Austria approx. 300MB (2.3 million edges) – her information like historic travel times are already included. Current traffic situation is not included as this is loaded over a web interface.

• Things like bike sharing station locations etc. are almost neglectable with respect to size.

2.3 Proactive improvement of transport systems quality and efficiency (Pilot Ljubljana)

2.3.1 Short description
Essentially, Ljubljana pilot of Multi-modal routing scenario is the same as Austria and West Midlands pilots. However, there are some specifics in Ljubljana use case description. One of them is that there is a lot of open source data that can be used within the project. This data enables us to carry out two specialized scenarios: park & ride (providing suggestions on demand or proactively) and recommendation based on detected mobility patterns.

2.3.2 Data sources (with focus on speed and volume)
Data will be provided (we are implementing a version of Optimum server in their premises to be able to share the data) by LPP and includes data about bus lines, stations, schedules, time of arrivals etc. Available data includes:

- List of bus lines
- List of all stations
- List of station of particular bus lines
- List of all bus numbers that stop at particular station
- Timetable of particular station for particular day
- Timetable of particular bus line for particular day
- Timetable of particular station for current time
- Predicted time of arrival at certain station (real time)
- GPS coordinates of buses
- Anonymised data of ticket validation. Data comprises monthly aggregated data.

2.3.3 Existing infrastructure, APIs and protocols
There is a simple webapp with API available for Ljubljana bus traffic info available at www.trola.si, which is a part of http://www.opendata.si project. Two searches are enabled through a RESTful API:

- GET / (station): Search for station and return information about matched stations.
- GET / (station)/(bus): Search for station, filter by bus number and return information about matched stations.
Returned data contains information about bus station, number of the bus, direction and ETA of the bus to that station within next hour. This data is static and does not take into account real-time traffic conditions.

The rest of the data described in 2.3.2 is not publicly available, isn’t available in one place and is not easy to use. In addition to that, data is not automatically updated and services have no control over data retrieval. The setup itself is also inefficient due to recalculations at every query, which leads to occasional server crashes.

### 2.3.4 Proposed infrastructure

For the reasons described above we’re going to develop a server that will 1) simplify usage of the data and 2) grant control to administrators.

The plan is to implement a REST API that will enable access to dynamic and static information.

Service will include an admin console. Each user will receive a token from admin and this token will grant access to the data. Each user will only have access to data that administrator will preapprove.

<table>
<thead>
<tr>
<th>Type of data</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>On demand</td>
</tr>
<tr>
<td>Access to data</td>
<td>REST API</td>
</tr>
<tr>
<td>Relevant global context</td>
<td>This data is crucial for carrying MMR scenarios in Ljubljana</td>
</tr>
</tbody>
</table>

### 2.4 Proactive charging schemes for freight transport (Portuguese Pilot)

#### 2.4.1 Short description

The Proactive Charging Schemes pilot is based on dynamic charging for road use by freight vehicles based on real-time conditions of the transport network by the use of parameters, such as level of service provided in the roads; road safety; comfort in trips; environmental data (CO2 Emissions & Noise); and economic values (cost maintenance & toll revenues). The model, additionally to improving financial ratios in infrastructures management, allows increased quality of life for our users and population living near the urban/national Roads.

OPTIMUM’s dynamic charging model will combine historical and real-time data collected, in order to calculate highways tolls pricing with x hours advance time. This way, the logistic companies could plan their routes according to the variation of the tolls. It will also establish a table of tolls prices for periods of the day and the week, and measure the different impacts in traffic, accidents, revenues, maintenance costs, CO2 emissions and noise levels in the highways versus national roads.
2.4.2 Data sources (with focus on speed and volume)

2.4.2.1 IP Data Sources

2.4.2.1.1 Highway Counters

Infraestruturas de Portugal (IP) has many vehicle-counting sensors throughout their road network. For the purpose of the pilot, which covers only a portion of Portugal’s highways, focusing on the northern part of Portugal (Figure 18), IP has selected about 270 active sensors (135 in each road direction) (Figure 19) from which to extract vehicle passages, road occupancy and average speeds.

Figure 18: Pilot's highways
Raw Data Examples

Counter data is delivered in two main ways: by Secure FTP, in CSV or XLSX formats and via SQL dumps. An example of the CSV/ XLSX data is presented in Table 5. All sensor readings have a sample rate of five minutes. Furthermore, a list of sensor metadata was provided by IP.

Table 5: CSV example for highway counter data

<table>
<thead>
<tr>
<th>Row Labels</th>
<th>Completude</th>
<th>Class A</th>
<th>Class B</th>
<th>Class C</th>
<th>Class D</th>
<th>Ligeiros</th>
<th>Pesados</th>
</tr>
</thead>
<tbody>
<tr>
<td>Janeiro</td>
<td>100,00%</td>
<td>5745</td>
<td>1361649</td>
<td>90689</td>
<td>4580</td>
<td>1367394</td>
<td>95269</td>
</tr>
<tr>
<td>00:00</td>
<td>100,00%</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>00:05</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>00:10</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00:15</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00:20</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>65</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00:25</td>
<td>71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>71</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00:30</td>
<td>86</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>86</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00:35</td>
<td>97</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>97</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00:40</td>
<td>102</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>102</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>00:45</td>
<td>113</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>113</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00:50</td>
<td>106</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>106</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00:55</td>
<td>126</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>126</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01:00</td>
<td>100,00%</td>
<td>0</td>
<td>119</td>
<td>1</td>
<td>0</td>
<td>119</td>
<td>1</td>
</tr>
<tr>
<td>01:05</td>
<td>102</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>102</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>01:10</td>
<td>136</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>136</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01:15</td>
<td>140</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>140</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>01:20</td>
<td>132</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>132</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The columns in Table 5 represent, from left to right, the date and time (Row Labels), completeness of the data by month, day and hour (Completeness) and the vehicle count for each class of vehicles (motorcycles (Class A), light (Class B), heavy for passengers (Class C) and heavy for goods (Class D)), and for generic light (Ligeiros) and heavy (Pesados) vehicles. An example of the raw dump data is shown below:

<table>
<thead>
<tr>
<th>ID_SENSOR</th>
<th>CAT1</th>
<th>CAT2</th>
<th>CAT3</th>
<th>CAT4</th>
<th>ESTADO</th>
<th>FECHA</th>
<th>ID_REG</th>
<th>INTENSIDAD</th>
<th>LIGEIROS</th>
<th>OCUPACION</th>
<th>VELOCIDAD</th>
<th>VOLUMEN</th>
<th>PESADOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>N0_1</td>
<td>106</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>2017-01-01 00:00</td>
<td>10601350</td>
<td>200</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>N0_2</td>
<td>107</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>2017-01-01 00:30</td>
<td>10601351</td>
<td>200</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.5</td>
<td>0.6</td>
<td>0.7</td>
</tr>
<tr>
<td>N0_3</td>
<td>108</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>2017-01-01 01:00</td>
<td>10601352</td>
<td>200</td>
<td>0.3</td>
<td>0.4</td>
<td>0.5</td>
<td>0.6</td>
<td>0.7</td>
<td>0.8</td>
</tr>
<tr>
<td>N0_4</td>
<td>109</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>2017-01-01 01:30</td>
<td>10601353</td>
<td>200</td>
<td>0.4</td>
<td>0.5</td>
<td>0.6</td>
<td>0.7</td>
<td>0.8</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Figure 20: Database dump with counter data

The columns in Figure 20 represent, from left to right, the unique ID of the sensor, the vehicle count for each class of vehicles (motorcycles (CAT 1), light (CAT 2), heavy for passengers (CAT 3) and heavy for goods (CAT 4)), the state of the sensor (ESTADO), the date and time of the reading (FECHA), the unique ID for the reading (ID_REG), the traffic intensity (INTENSIDAD), the...
number of light vehicles (LIGEROS), the percentage of occupation of the highway (OCUPACION), the average speed (VELOCIDAD) the volume of vehicles in the highway (VOLUMEN) and the number of heavy vehicles (PESADOS).

The columns in Table 6 represent, from left to right, the highway concession’s name (Grupo), unique ID for the counter (Nome Equipamento), counter’s state (Estado), highway name (Silego Antigo), kilometre point (PK), section (Sublanço), latitude, longitude, bearing (Sentido) and concession holder (Holder).

Table 6: Sensor metadata sample

<table>
<thead>
<tr>
<th>Grupo</th>
<th>Nome Equipamento</th>
<th>Estado</th>
<th>Silego Antigo</th>
<th>PK</th>
<th>Sublanço</th>
<th>Latitude</th>
<th>Longitude</th>
<th>sentido</th>
<th>holder</th>
</tr>
</thead>
<tbody>
<tr>
<td>EP Grande Porto (ex AEDL)</td>
<td>A1_297+975_CT3687_C</td>
<td>Ativo</td>
<td>297,00</td>
<td>Santo Ovídeo - Coimbrões (A44)</td>
<td>41,110336</td>
<td>-8,607517</td>
<td>c</td>
<td>IP</td>
<td></td>
</tr>
<tr>
<td>EP Grande Porto (ex AEDL)</td>
<td>A1_297+975_CT3687_D</td>
<td>Ativo</td>
<td>297,00</td>
<td>Santo Ovídeo - Coimbrões (A44)</td>
<td>41,110336</td>
<td>-8,607517</td>
<td>d</td>
<td>IP</td>
<td></td>
</tr>
<tr>
<td>EP Grande Porto (ex AEDL)</td>
<td>A1_300+250_CT3688_C</td>
<td>Ativo</td>
<td>300,20</td>
<td>Coimbrões (A44) - Canidelo</td>
<td>41,125833</td>
<td>-8,635556</td>
<td>c</td>
<td>IP</td>
<td></td>
</tr>
<tr>
<td>EP Grande Porto (ex AEDL)</td>
<td>A1_300+920_CT3689_C</td>
<td>Ativo</td>
<td>300,70</td>
<td>Canidelo - Ponte da Arrábida Sul (Afurada)</td>
<td>41,132069</td>
<td>-8,635632</td>
<td>c</td>
<td>IP</td>
<td></td>
</tr>
<tr>
<td>EP Grande Porto (ex AEDL)</td>
<td>A1_300+920_CT3689_D</td>
<td>Ativo</td>
<td>300,70</td>
<td>Canidelo - Ponte da Arrábida Sul (Afurada)</td>
<td>41,132069</td>
<td>-8,635632</td>
<td>d</td>
<td>IP</td>
<td></td>
</tr>
</tbody>
</table>

Data Adaptor

A data collection and harmonization adaptor was implemented, in R language for testing purposes (Figure 21). Sensor data is stored in two different MongoDB BSON data structures: sensors and sensor_values. The first (Figure 22) stores sensors’ metadata, such as location, highway name, sensor holder, bearing or state. The latter (Figure 23), stores the sensor readings.
Figure 21: IP Counter Sensors Adaptor

- **StoreDumpSensorDataIntoMongo**: process that connects to a SQL database already containing the dump, cleans and harmonizes data and stores it into MongoDB `sensor_values` structure.
- **StoreCSVSensorDataIntoMongo**: this process reads a CSV file containing counter data readings, cleans and harmonizes the data and stores it into MongoDB `sensor_values` structure.
- **StoreSensorMetadataIntoMongo**: utility to store the sensor metadata list into MongoDB `sensors` structure.
Transformed Data Examples

```json
1 {
2     "id" : ObjectId("56ab5c569f6ed594781cc00f"),
3     "concession_name" : "EP Grande Porto (ex AEBL)",
4     "road_name" : "A1",
5     "road_type" : "highway",
6     "sensor_type" : "counter",
7     "km_point" : 297.0,
8     "sensor_id_ip" : "A1_297+975_CT3687_C",
9     "sensor_id_holder" : NumberInt(-1),
10    "section" : "Santo Oviedo - Coimbrões (A44)",
11    "state" : "Ativo",
12    "concession_holder" : "IP",
13    "bearing" : "c",
14    "location" : {
15        "type" : "Point",
16        "coordinates" : [
17            -8.607517,
18            41.11034
19        ]
20    }
21 }
```

Figure 22: Sample transformed counter metadata – BSON Format

```json
1 {
2     "id" : ObjectId("56ab5ce39f6ed594781cc171"),
3     "sensor_id" : ObjectId("56ab5c569f6ed594781cc153"),
4     "a_vehicles" : NumberInt(0),
5     "b_vehicles" : NumberInt(2088),
6     "c_vehicles" : NumberInt(48),
7     "d_vehicles" : NumberInt(0),
8     "date_time" : ISODate("2014-01-02T00:15:00.000+0000"),
9     "total_vehicles" : NumberInt(2136),
10    "light_vehicles" : NumberInt(2088),
11    "heavy_vehicles" : NumberInt(48),
12    "occupancy" : NumberInt(12),
13    "average_speed" : NumberInt(76),
14    "volume" : NumberInt(178)
15 }
```

Figure 23: Sample transformed counter reading – BSON Format
Data Metrics

Available counter data spans from January 1st, 2014 at 00:00 until the present day, and it has a sample rate of five minutes, which means 12 readings per hour. Per year, the number of readings for one counter is calculated below:

\[
\text{number of readings per hour} \times \text{hours per day} \times \text{days per year} = \\
12 \times 24 \times 365 = 105.120
\]

Multiplying by the total number of sensors we have the number of total readings per year:

\[
105.120 \times 270 = 28.382.400
\]

Of course some counters have time spans with no data, due to inactivity or malfunctioning periods. Data quality tests were made on sensor data, in order to check the overall quality of the readings. A sample of the data quality test is shown in Figure 24.
The sensors are organized in pairs, each for a specific direction of the highway. It is visible the time spans of inactivity for each sensor, represented by the white spaces. Figure 3.3.7 represents a sample of 50 counters for the year of 2014. From this test, it was created a quality factor, $\eta_{\text{sensor}}^{\text{year}}$, which represents the percentage of data completeness for a sensor per year:

$$\eta_{\text{sensor}}^{\text{year}} = \frac{\text{# of stored readings per year}}{\text{# of expected readings per year}} \times 100$$

For instance, regarding the sensor with ID $\text{A20}_0+650_\text{CT3683}_\text{C}$, corresponding to the 7th column of Figure 3.3.6, for the year 2014, the quality factor is:

$$\eta_{\text{A20}_0+650_\text{CT3683}_\text{C}}^{2014} = \frac{103.206}{105.120} \times 100 = 98,1792\%$$

### Data Storage Requirements

Per year, the 28,382,400 records correspond to 1,761,944,810 bytes, or 1,762 GB approximately. This value corresponds to MongoDB’s maximum data size necessities. This value does not take into account the sensors’ metadata, since it is small enough not to be representative (in the order of the hundreds of KB).

#### 2.4.2.1.2 Traffic Events

IP keeps records from traffic events occurring in their infrastructure since 2011. Of course not all events are kept, since some of them are not reported to authorities nor captured by the appropriate mechanisms (CCTV, traffic sensors, etc.).

### Raw Data Examples

As in the case of counter data, traffic events are delivered in two different ways: SQL dumps or Web Services. The SQL dumps (Table 7) are used to access historic events, going back to 2011 and the SOAP Web Service (Figure 25) is used to get daily updates for traffic events.

**Table 7: Traffic events SQL dump sample**

<table>
<thead>
<tr>
<th>TIPO</th>
<th>ESTRADA</th>
<th>KM</th>
<th>SENTIDO</th>
<th>DATA INICIO</th>
<th>DATA FIM</th>
<th>COORD_X</th>
<th>COORD_Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accident</td>
<td>N15</td>
<td>14</td>
<td>Crescente</td>
<td>11.02.02 07:05:00</td>
<td>11.02.02 09:35:55</td>
<td>41,18795653</td>
<td>-8,43626538</td>
</tr>
<tr>
<td>Accident</td>
<td>N1</td>
<td>132</td>
<td>Ambos</td>
<td>11.01.27 17:14:00</td>
<td>11.01.27 19:16:26</td>
<td>39,79876221</td>
<td>-8,74554542</td>
</tr>
<tr>
<td>Accident</td>
<td>A41</td>
<td>0</td>
<td>Crescente</td>
<td>11.02.21 19:52:28</td>
<td>11.03.02 01:00:57</td>
<td>41,23250836</td>
<td>-8,695554073</td>
</tr>
<tr>
<td>Accident</td>
<td>A4</td>
<td>11,4</td>
<td>Crescente</td>
<td>11.02.14 15:24:00</td>
<td>11.02.14 15:37:51</td>
<td>41,2012649</td>
<td>-8,549563942</td>
</tr>
<tr>
<td>AbnormalTraffic</td>
<td>A41</td>
<td>8,82</td>
<td>Ambos</td>
<td>11.02.12 00:54:00</td>
<td>11.02.12 10:15:00</td>
<td>41,24010777</td>
<td>-8,606287621</td>
</tr>
<tr>
<td>Event Type</td>
<td>N</td>
<td>Location</td>
<td>Start Date/Time</td>
<td>End Date/Time</td>
<td>Latitude</td>
<td>Longitude</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>----</td>
<td>----------</td>
<td>-------------------------</td>
<td>------------------------</td>
<td>----------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>MaintenanceWorks</td>
<td>N1</td>
<td>Crescente</td>
<td>11.02.08 09:00:00</td>
<td>11.02.08 15:43:20</td>
<td>38,976</td>
<td>8,973</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>N14</td>
<td>Ambos</td>
<td>11.02.16 01:07:48</td>
<td>11.03.02 01:00:57</td>
<td>41,403</td>
<td>8,500</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>N14</td>
<td>Ambos</td>
<td>11.02.08 16:27:00</td>
<td>11.02.08 16:37:29</td>
<td>41,352</td>
<td>8,553</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>A28</td>
<td>Crescente</td>
<td>11.03.11 19:17:00</td>
<td>11.03.11 20:32:31</td>
<td>41,164</td>
<td>8,479</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>A28</td>
<td>Crescente</td>
<td>11.02.09 20:47:00</td>
<td>11.02.09 21:52:00</td>
<td>41,149</td>
<td>8,408</td>
<td></td>
</tr>
<tr>
<td>AbnormalTraffic</td>
<td>A28</td>
<td>Decrescente</td>
<td>11.02.26 18:08:13</td>
<td>11.03.02 01:00:57</td>
<td>41,191</td>
<td>8,792</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>A4</td>
<td>Ambos</td>
<td>11.02.26 18:10:20</td>
<td>11.03.02 01:00:57</td>
<td>41,188</td>
<td>8,687</td>
<td></td>
</tr>
<tr>
<td>AbnormalTraffic</td>
<td>A4</td>
<td>Crescente</td>
<td>11.02.19 18:32:23</td>
<td>11.03.02 01:00:57</td>
<td>41,230</td>
<td>8,680</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>A4</td>
<td>Crescente</td>
<td>11.02.19 17:45:00</td>
<td>11.02.19 18:57:24</td>
<td>41,175</td>
<td>8,495</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>A4</td>
<td>Ambos</td>
<td>11.02.19 19:53:36</td>
<td>11.03.02 01:00:57</td>
<td>41,206</td>
<td>8,636</td>
<td></td>
</tr>
<tr>
<td>AbnormalTraffic</td>
<td>A41</td>
<td>Ambos</td>
<td>11.02.20 00:19:45</td>
<td>11.03.02 01:00:57</td>
<td>41,240</td>
<td>8,645</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>A28</td>
<td>Crescente</td>
<td>11.02.24 10:31:00</td>
<td>11.02.24 11:29:28</td>
<td>41,156</td>
<td>8,645</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>A4</td>
<td>Crescente</td>
<td>11.03.14 17:20:00</td>
<td>11.03.14 18:25:37</td>
<td>41,746</td>
<td>-6,566</td>
<td></td>
</tr>
<tr>
<td>VehicleObstruction</td>
<td>A4</td>
<td>Decrescente</td>
<td>11.03.14 18:46:00</td>
<td>11.03.14 18:49:19</td>
<td>41,191</td>
<td>8,481</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>A4</td>
<td>Decrescente</td>
<td>11.02.14 18:39:00</td>
<td>11.02.14 19:50:32</td>
<td>41,180</td>
<td>8,445</td>
<td></td>
</tr>
<tr>
<td>VehicleObstruction</td>
<td>A4</td>
<td>Crescente</td>
<td>11.02.14 21:53:00</td>
<td>11.02.14 22:00:13</td>
<td>41,193</td>
<td>8,484</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>A4</td>
<td>Crescente</td>
<td>11.03.06 20:08:00</td>
<td>11.03.06 21:12:58</td>
<td>41,174</td>
<td>8,436</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>A4</td>
<td>Decrescente</td>
<td>11.02.19 00:13:00</td>
<td>11.02.19 01:37:48</td>
<td>41,173</td>
<td>8,425</td>
<td></td>
</tr>
<tr>
<td>AbnormalTraffic</td>
<td>A4</td>
<td>Ambos</td>
<td>11.03.14 15:41:38</td>
<td>11.03.15 01:16:37</td>
<td>41,227</td>
<td>8,673</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>A4</td>
<td>Decrescente</td>
<td>11.02.15 17:56:00</td>
<td>11.02.15 19:32:48</td>
<td>41,214</td>
<td>8,273</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>A4</td>
<td>Decrescente</td>
<td>11.02.12 08:44:00</td>
<td>11.02.12 10:35:04</td>
<td>41,197</td>
<td>8,508</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>A4</td>
<td>Decrescente</td>
<td>11.02.13 11:25:00</td>
<td>11.02.13 12:00:00</td>
<td>41,203</td>
<td>8,542</td>
<td></td>
</tr>
</tbody>
</table>

The SQL dump’s fields represent, from left to right, the Datex II-standardized type of event (TIPO), road (ESTRADA), kilometre point (KM), bearing (SENTIDO), start date and time (DATA INICIO), end date and time (DATA FIM), latitude (COORD_X) and longitude (COORD_Y) of the event.
The Web Service provides a unique ID for the event or occurrence (Id), date and time of the record (Data), start (DataInicio) and end (DataFim) dates and times for the event, country code (CodigoPais), Datex II-standardized type of event (Tipo), state of the event (Estado), description (in Portuguese) (Descricao), initial and final municipalities (DistritoInicial, DistritoFinal) and counties (ConcelhoInicial, ConcelhoFinal), road name (Estrada), bearing (Direccao), geometric point (PontoGeometrico) and kilometre point (Km).

**Data Adaptor**

A data collection and harmonization adaptor was implemented, in R language for testing purposes (Figure 26). Traffic events, which are active, are stored in a MongoDB BSON data structure called events (Figure 27). The ones that are no longer active, or that have an end date prior to the present day are stored in a twin data structure called events_hist.
Figure 26: IP Traffic Events Adaptor

- **ReadTrafficEventsWebServiceIntoMongo**: process that connects to the Web Service via SOAP protocol, retrieves, cleans and harmonizes the data, and stores it into MongoDB `events` structure.
- **ReadDumpTrafficEventsIntoMongo**: the dump only contains past events, hence this process fetches data from the dump, cleans and harmonizes it and stores it into MongoDB `events_hist` structure.
- **StorePastEventsIntoHistorianDB**: this utility checks for past events in the `events` structure and stores them in the `events_hist` structure.
**Transformed Data Examples**

```json
1 {
   "id" : ObjectId("56fe7f873e0d7a3b58035857"),
   "type" : "Accident",
   "road" : "A28",
   "km_point" : 1.8,
   "bearing" : "northbound",
   "start_date_time" : ISODate("2011-03-11T19:17:00.000+0000"),
   "end_date_time" : ISODate("2011-03-11T20:32:31.000+0000"),
   "location" : {
      "type" : "Point",
      "coordinates" : [
         -8.647999,
         41.16412
      ]
   },
   "source" : "IP",
   "version" : NumberInt(1),
   "severity" : "unknown"
}
```

Figure 27: Sample transformed traffic events data – BSON Format

**Data Metrics**

Data quality tests were performed over the SQL dump traffic events data, in order to have a better idea of how the events data is scattered in terms of geography (Figure 28), time (Figure 29 and Figure 30) and type of event (Figure 31).

![Events other than accidents vs Accidents](image)

Figure 28: Scattering of accidents and other types of events throughout the pilot's area
Figure 29: Boxplot showing monthly variation of number of events per year

Figure 30: Boxplot showing variation of number of events per month between 2011-2015
Figure 31: Events distribution per event type

Figure 31 presents the number of events in the SQL dump per Datex-II type of event, namely from left to right, accidents, maintenance works, abnormal traffic, vehicle obstructions, animal obstructions, weather related traffic events, general obstructions, general incidents, activities, poor infrastructure, environmental obstructions, non-weather-related events, roadside assistance, equipment damage obstructions and construction works. Regarding the Web Service, it is updated daily, and does not present any regular update rate. It depends solely on traffic technicians' inputs.

Data Storage Requirements

The information concerning traffic events has low storage requirements. In fact, each event instance stored in the MongoDB database occupies an average of 270 bytes. In total, for the 3451 events stored, the maximum data size reserved by MongoDB is 932.815 bytes, or 932.82 kB.

2.4.2.1.3 Concession Toll passage counters

IP has access to toll counters' data, which are provided by the different concession holders. In the pilot’s area, there are two main concession holders: ViaLivre, which holds highway A28, and Ascendi, which is the concessionaire for A4, A25, A29 and A41. These highways have a total number of 66 toll sensors (33 for each bearing), from which 8 are from ViaLivre and the rest from Ascendi.
Raw Data Examples

Both concession holders deliver their data through SFTP, in CSV files (Table 8 and Table 9).

Table 8: ViaLivre toll sensor data sample

<table>
<thead>
<tr>
<th>CONCESSAO</th>
<th>PORTICO</th>
<th>SUBLANCO</th>
<th>DATA</th>
<th>CLASSE1</th>
<th>CLASSE2</th>
<th>CLASSE3</th>
<th>CLASSE4</th>
<th>CLASSE5</th>
</tr>
</thead>
<tbody>
<tr>
<td>93</td>
<td>2803</td>
<td>0</td>
<td>201010150000,00</td>
<td>36</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>2804</td>
<td>0</td>
<td>201010150000,00</td>
<td>21</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>2811</td>
<td>0</td>
<td>201010150000,00</td>
<td>16</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>2812</td>
<td>0</td>
<td>201010150000,00</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>2817</td>
<td>0</td>
<td>201010150000,00</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>2818</td>
<td>0</td>
<td>201010150000,00</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>2821</td>
<td>0</td>
<td>201010150000,00</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>2822</td>
<td>0</td>
<td>201010150000,00</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>2822</td>
<td>0</td>
<td>201010150005,00</td>
<td>32</td>
<td>8</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>2803</td>
<td>0</td>
<td>201010150005,00</td>
<td>17</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>2811</td>
<td>0</td>
<td>201010150005,00</td>
<td>13</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>2817</td>
<td>0</td>
<td>201010150005,00</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>2818</td>
<td>0</td>
<td>201010150005,00</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>2821</td>
<td>0</td>
<td>201010150005,00</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>2822</td>
<td>0</td>
<td>201010150005,00</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

The columns represent, from left to right, concession number (CONCESSAO), unique ID for the toll counter (PORTICO), section ID (SUBLANCO), date (DATA) and toll passages for classes 1 to 5 (CLASSE 1-5).

Table 9: Ascendi toll sensor data sample

<table>
<thead>
<tr>
<th>CONCESSAO</th>
<th>PORTICO</th>
<th>DATA</th>
<th>CLASS1</th>
<th>CLASS2</th>
<th>CLASS3</th>
<th>CLASS4</th>
<th>CLASS5</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>407</td>
<td>201501010000,00</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>408</td>
<td>201501010000,00</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>409</td>
<td>201501010000,00</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>410</td>
<td>201501010000,00</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>2510</td>
<td>201501010000,00</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>2514</td>
<td>201501010000,00</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>2518</td>
<td>201501010000,00</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>2522</td>
<td>201501010000,00</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>2525</td>
<td>201501010000,00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>2526</td>
<td>201501010000,00</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>2529</td>
<td>201501010000,00</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The columns in Table 3.3.5 represent, from left to right, concession number (CONCESSAO), unique ID for the toll counter (PORTICO), date (DATA) and toll passages for classes 1 to 5 (CLASS1-5).

Data Adaptor

A data collection and harmonization adaptor was implemented, in R language for testing purposes (Figure 32). Because both types of sensor metadata and data, toll and counter sensors, are stored in the same data structures sensors and sensor_values (Figure 33), a sensor_type (Figure 34) field was added to differentiate tolls and counters.

<table>
<thead>
<tr>
<th>CONCESSAO</th>
<th>DATA</th>
<th>CLASS1</th>
<th>CLASS2</th>
<th>CLASS3</th>
<th>CLASS4</th>
<th>CLASS5</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 2535</td>
<td>201501010000,00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17 2539</td>
<td>201501010000,00</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17 2543</td>
<td>201501010000,00</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17 2548</td>
<td>201501010000,00</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17 2554</td>
<td>201501010000,00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>17 2573</td>
<td>201501010000,00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11 2910</td>
<td>201501010000,00</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11 2923</td>
<td>201501010000,00</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11 2924</td>
<td>201501010000,00</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>12 4101</td>
<td>201501010000,00</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12 4102</td>
<td>201501010000,00</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12 4106</td>
<td>201501010000,00</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12 4107</td>
<td>201501010000,00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12 4108</td>
<td>201501010000,00</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12 4109</td>
<td>201501010000,00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12 4110</td>
<td>201501010000,00</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12 4113</td>
<td>201501010000,00</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12 4114</td>
<td>201501010000,00</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12 4115</td>
<td>201501010000,00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12 4116</td>
<td>201501010000,00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12 4119</td>
<td>201501010000,00</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12 4120</td>
<td>201501010000,00</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
- **StoreTollSensorMetadataIntoMongo**: utility to store the toll sensor metadata list into MongoDB sensors structure.
- **StoreViaLivreSensorDataIntoMongo**: this process reads a CSV file containing ViaLivre toll sensor data readings, cleans and harmonizes the data and stores it into MongoDB sensor_values structure.
- **StoreAscendiSensorDataIntoMongo**: this process reads a CSV file containing Ascendi toll sensor data readings, cleans and harmonizes the data and stores it into MongoDB sensor_values structure.
Transformed Data Examples

```json
1 {
    "_id" : ObjectId("56ab5c569f6ed594781cc081"),
    "Concession_name" : "Costa Prata",
    "road_name" : "A25",
    "road_type" : "highway",
    "sensor_type" : "toll",
    "km_point" : 12.93,
    "sensor_id_ip" : "-1",
    "sensor_id_holder" : NumberInt(2509),
    "section" : "Aveiro",
    "state" : "Ativo",
    "concession_holder" : "Ascendi",
    "bearing" : "c",
    "location" : {
        "type" : "Point",
        "coordinates" : [
            -8.611389,
            40.64592
        ]
    }
}
```

Figure 33: Sample transformed toll sensor metadata – BSON Format

```json
1 {
    "_id" : ObjectId("57015e7c60a5dee6439d5122"),
    "sensor_id" : ObjectId("56ab5c569f6ed594781cc081"),
    "date_time" : ISODate("2015-01-01T22:45:00.000+0000"),
    "class_1_vehicles" : NumberInt(5),
    "class_2_vehicles" : NumberInt(0),
    "class_3_vehicles" : NumberInt(0),
    "class_4_vehicles" : NumberInt(0),
    "class_5_vehicles" : NumberInt(0),
    "light_vehicles" : NumberInt(5),
    "heavy_vehicles" : NumberInt(0),
    "total_vehicles" : NumberInt(5)
}
```

Figure 34: Sample transformed toll sensor reading – BSON Format

Data Metrics

Available toll sensor data from ViaLivre spans from October 15th, 2010 at 00:00 until the present day, and it has a sample rate of five minutes, which means 12 readings per hour. Available toll sensor data from Ascendi spans from January 1st, 2014 at 00:00 until the present day, and it has a sample rate of five minutes, which means 12 readings per hour. Per year, the number of readings is the same as in the case of counters, 6,937,920.
Of course some toll sensors have time spans with no data, due to inactivity or malfunctioning periods. Data quality tests were made on sensor data, in order to check the overall quality of the readings. A sample of the data quality tests is shown in Figure 35.

![Data Quality Test](image)

**Figure 35:** Data quality test for a A28 toll sensor readings per hour, from 15th of October to 31st December, 2010

Figure 35 represents the number of readings per hour for a sensor in A28, during the data span covered for 2010. In a generic case, there are 12 readings per hour (one reading per 5 minutes), but there are flaws in the data, due to toll sensor malfunctioning.

**Data Storage Requirements**

The data storage requirements are the same as for IP’s counter data.

2.4.2.2 **LS Data Sources**

Luis Simões (LS) provides data from their truck fleet. The data received comprises data from four trucks spanning from the 1st of April, 2015 to the 7th of December 2015 and daily consumption rates.
2.4.2.2.1 Truck Events and Consumption

Truck events are captured from the ECU (ignition on and off, speed excess, etc.) and from drivers’ inputs (beginning and ending of deliveries), whereas consumption data is aggregated daily. Data is delivered in CSV files through SFTP for both concessions.

**Raw Data Examples**

<table>
<thead>
<tr>
<th>Id</th>
<th>Data</th>
<th>Código</th>
<th>Descrição</th>
<th>Local</th>
<th>Código do Conduitor</th>
<th>Total Km</th>
<th>Velocidade</th>
<th>Latitude</th>
<th>Longitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23-07-2015 00:01</td>
<td>1958</td>
<td>Ev.Temporizador-1</td>
<td></td>
<td>1130</td>
<td>894725</td>
<td>79</td>
<td>40,5877</td>
<td>-6,75893</td>
</tr>
<tr>
<td>2</td>
<td>23-07-2015 00:03</td>
<td>1958</td>
<td>Alt. Estado da Viatura</td>
<td></td>
<td>1130</td>
<td>894728</td>
<td>0</td>
<td>40,59212</td>
<td>-6,79258</td>
</tr>
<tr>
<td>3</td>
<td>23-07-2015 00:03</td>
<td>1958</td>
<td>Alt. Estado Condução</td>
<td></td>
<td>1130</td>
<td>894728</td>
<td>0</td>
<td>40,59212</td>
<td>-6,7926</td>
</tr>
<tr>
<td>4</td>
<td>23-07-2015 00:03</td>
<td>1958</td>
<td>Evento de Ignição Off</td>
<td>Alt. de Estado da Viatura</td>
<td>1130</td>
<td>894728</td>
<td>0</td>
<td>40,59212</td>
<td>-6,7926</td>
</tr>
<tr>
<td>5</td>
<td>23-07-2015 00:04</td>
<td>1958</td>
<td>Exc. de Rot. em Ralenti Tempo Máximo</td>
<td></td>
<td>1130</td>
<td>894728</td>
<td>0</td>
<td>40,59212</td>
<td>-6,7926</td>
</tr>
<tr>
<td>6</td>
<td>23-07-2015 00:04</td>
<td>1958</td>
<td>Evento de Paragem</td>
<td></td>
<td>1130</td>
<td>894728</td>
<td>0</td>
<td>40,59212</td>
<td>-6,7926</td>
</tr>
<tr>
<td>7</td>
<td>23-07-2015 00:09</td>
<td>1958</td>
<td>Ped. Actual. Listas</td>
<td></td>
<td>1130</td>
<td>894728</td>
<td>0</td>
<td>40,59212</td>
<td>-6,7926</td>
</tr>
<tr>
<td>8</td>
<td>23-07-2015 00:09</td>
<td>1958</td>
<td>Alt. Estado Condução</td>
<td></td>
<td>1130</td>
<td>894728</td>
<td>5</td>
<td>40,59212</td>
<td>-6,7926</td>
</tr>
<tr>
<td>9</td>
<td>23-07-2015 00:10</td>
<td>1958</td>
<td>Alt. Estado da Viatura</td>
<td></td>
<td>1130</td>
<td>894728</td>
<td>6</td>
<td>40,59212</td>
<td>-6,79263</td>
</tr>
<tr>
<td>10</td>
<td>23-07-2015 00:11</td>
<td>1958</td>
<td>Exc. de Rot. em Ralenti</td>
<td>Alt. de Estado da Viatura</td>
<td>1130</td>
<td>894728</td>
<td>52</td>
<td>40,59225</td>
<td>-6,79558</td>
</tr>
<tr>
<td>11</td>
<td>23-07-2015 00:11</td>
<td>1958</td>
<td>Excesso de Velocidade</td>
<td></td>
<td>1130</td>
<td>894729</td>
<td>87</td>
<td>40,5924</td>
<td>-6,79965</td>
</tr>
<tr>
<td>12</td>
<td>23-07-2015 00:12</td>
<td>1958</td>
<td>Excesso de Velocidade</td>
<td></td>
<td>1130</td>
<td>894730</td>
<td>91</td>
<td>40,59352</td>
<td>-6,80935</td>
</tr>
<tr>
<td>13</td>
<td>23-07-2015 00:13</td>
<td>1958</td>
<td>Excesso de Velocidade</td>
<td></td>
<td>1130</td>
<td>894731</td>
<td>73</td>
<td>40,60468</td>
<td>-6,81748</td>
</tr>
</tbody>
</table>
Columns of Table 10 represent, from left to right, unique ID of the reading (ID), date and time (Data), truck ID (Código), event description (Descrição), address of the event (Local), driver ID (Código do Condutor), truck’s total kilometres (Total Km), instant speed (Velocidade), latitude (Latitude) and longitude (Longitude). If the time between events is greater than 5 minutes, a special event is triggered so that there isn’t five minutes between events, so the data contains GPS location data with a sample rate of at least five minutes.

Table 11: Truck consumption data sample

<table>
<thead>
<tr>
<th>Id</th>
<th>Descrição</th>
<th>Código</th>
<th>Dist.Per.(Km)</th>
<th>Vel.Média (Km/H)</th>
<th>Vel.Cruzeiro (Km/H)</th>
<th>Cons.Efect.(Lit)</th>
<th>Cons.Esp. (Lit./100 Km)</th>
<th>Cons.Espec.P (Def.Pond. (Lit./100 Km))</th>
<th>Desv.Cons.P (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23-07-2015</td>
<td>1958</td>
<td>756</td>
<td>54,74</td>
<td>62,99</td>
<td>282</td>
<td>37,3</td>
<td>30</td>
<td>24,34</td>
</tr>
<tr>
<td>2</td>
<td>24-07-2015</td>
<td>1958</td>
<td>915</td>
<td>56,78</td>
<td>69,1</td>
<td>361</td>
<td>39,45</td>
<td>30</td>
<td>31,51</td>
</tr>
<tr>
<td>3</td>
<td>25-07-2015</td>
<td>1958</td>
<td>342</td>
<td>73,48</td>
<td>78,84</td>
<td>101</td>
<td>29,53</td>
<td>30</td>
<td>-1,56</td>
</tr>
<tr>
<td>4</td>
<td>26-07-2015</td>
<td>1958</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>27-07-2015</td>
<td>1958</td>
<td>374</td>
<td>41,18</td>
<td>63,7</td>
<td>182</td>
<td>48,66</td>
<td>30</td>
<td>62,21</td>
</tr>
<tr>
<td>7</td>
<td>29-07-2015</td>
<td>1958</td>
<td>261</td>
<td>58,17</td>
<td>73,43</td>
<td>77</td>
<td>29,5</td>
<td>30</td>
<td>-1,66</td>
</tr>
<tr>
<td>8</td>
<td>30-07-2015</td>
<td>1958</td>
<td>3</td>
<td>13,11</td>
<td>22,69</td>
<td>1</td>
<td>33,33</td>
<td>30</td>
<td>11,11</td>
</tr>
<tr>
<td>9</td>
<td>31-07-2015</td>
<td>1958</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>01-08-2015</td>
<td>1958</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>02-08-2015</td>
<td>1958</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>03-08-2015</td>
<td>1958</td>
<td>317</td>
<td>61,18</td>
<td>76,65</td>
<td>90</td>
<td>28,39</td>
<td>30</td>
<td>-5,36</td>
</tr>
<tr>
<td>13</td>
<td>04-08-2015</td>
<td>1958</td>
<td>834</td>
<td>67,87</td>
<td>79,44</td>
<td>265</td>
<td>31,77</td>
<td>30</td>
<td>5,92</td>
</tr>
<tr>
<td>14</td>
<td>05-08-2015</td>
<td>1958</td>
<td>699</td>
<td>55,47</td>
<td>67,09</td>
<td>280</td>
<td>40,06</td>
<td>30</td>
<td>33,52</td>
</tr>
<tr>
<td>15</td>
<td>06-08-2015</td>
<td>1958</td>
<td>771</td>
<td>68,12</td>
<td>77,94</td>
<td>243</td>
<td>31,52</td>
<td>30</td>
<td>5,06</td>
</tr>
<tr>
<td>16</td>
<td>07-08-2015</td>
<td>1958</td>
<td>841</td>
<td>63,15</td>
<td>74,09</td>
<td>281</td>
<td>33,41</td>
<td>30</td>
<td>11,38</td>
</tr>
<tr>
<td>17</td>
<td>08-08-2015</td>
<td>1958</td>
<td>238</td>
<td>68,59</td>
<td>75,9</td>
<td>72</td>
<td>30,25</td>
<td>30</td>
<td>0,84</td>
</tr>
<tr>
<td>18</td>
<td>09-08-2015</td>
<td>1958</td>
<td>2</td>
<td>5,84</td>
<td>15</td>
<td>1</td>
<td>50</td>
<td>30</td>
<td>66,67</td>
</tr>
<tr>
<td>19</td>
<td>10-08-2015</td>
<td>1958</td>
<td>529</td>
<td>59,53</td>
<td>70,09</td>
<td>217</td>
<td>41,02</td>
<td>30</td>
<td>36,74</td>
</tr>
<tr>
<td>20</td>
<td>11-08-2015</td>
<td>1958</td>
<td>836</td>
<td>66,62</td>
<td>72,5</td>
<td>268</td>
<td>32,06</td>
<td>30</td>
<td>6,86</td>
</tr>
<tr>
<td>22</td>
<td>13-08-2015</td>
<td>1958</td>
<td>821</td>
<td>68,48</td>
<td>74,5</td>
<td>247</td>
<td>30,09</td>
<td>30</td>
<td>0,28</td>
</tr>
</tbody>
</table>
Columns in Table 11 represent, from left to right, the unique ID for the consumption daily reading (id), date (Descrição), truck ID (Código), covered distance (Dist.Per.(Km)), average speed (Vel.Média (Km/H)), cruise speed (Vel.Cruzeiro (Km/H)), effective consumption (Cons.Efect.(Lit)), consumption per 100 Km (Cons.Espec.(Lit./100 Km)), expected consumption (Cons.Espec.P.Def.Pond.(Lit./100 Km)) and percent deviation from the expected consumption (Desv.Cons.P (%)).

Data Adaptor

A data collection and harmonization adaptor was implemented, in R language for testing purposes (Figure 36). Localized truck events are stored into a truck_events structure in MongoDB (Figure 37), whereas consumption data is stored in a MongoDB structure called truck_consumption_rates (Figure 38).

- **StoreTruckEventsIntoMongo**: process that reads an Excel spreadsheet with trucks’ event data, cleans and harmonizes the data and stores it in MongoDB truck_events structure.
- **StoreTruckConsumptionIntoMongo**: process that reads an Excel spreadsheet with trucks’ daily consumption data, cleans and harmonizes such data and stores it into MongoDB truck_consumption_rates structure.
Transformed Data Examples

```json
1 {
2   "id" : ObjectId("56952957585c5b8cc894c822"),
3   "date" : ISODate("2015-04-01T07:25:00.000+0000"),
4   "vehicle_id" : NumberInt(1953),
5   "description" : "Início Servico",
6   "information" : "Viagem - Iniciado: 000004475764",
7   "address" : "Rua Vieira Pinto - Oliveira do Douro - Vila Nova de Gaia - PRT - [4110]",
8   "driver_id" : NumberInt(202),
9   "total_km" : NumberInt(756934),
10  "speed" : NumberInt(0),
11  "location" : {
12    "type" : "Point",
13    "coordinates" : [
14      -8.595266,
15      41.13205
16    ]
17  }
18 }
```

Figure 37: Truck localized events’ data - BSON format

```json
1 {
2   "id" : ObjectId("5671689dfe9246202cf2b362"),
3   "date" : ISODate("2015-03-31T23:00:00.000+0000"),
4   "vehicle_id" : NumberInt(1953),
5   "distance" : 132.0,
6   "average_speed" : 29.61,
7   "cruise_speed" : 52.13,
8   "effective_consumption" : 55.0,
9   "spec_consumption" : 42.42,
10  "expected_consumption" : 38.0,
11  "deviation" : 41.41
12 }
```

Figure 38: Truck daily consumption data - BSON format

Data Metrics

LS provided data from four trucks, spanning from the 1st of April, 2015 to the 7th of December 2015, which represents about 13,000 trips containing origin-destination pairs and GPS tracks, with each interval between GPS readings being equal or lower than five minutes, when the truck is on trip. This means that, for the 251 days of the data span, we have a total of 330.994 readings for truck events and 251 readings for consumption rates (Table 12). In terms of data quality, there are no significant gaps on the consumption rates nor on the GPS tracks.

<table>
<thead>
<tr>
<th>Truck ID</th>
<th>Event Readings</th>
<th>Consumption Readings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1953</td>
<td>69.693</td>
<td>251</td>
</tr>
</tbody>
</table>

Table 12: Number of truck data readings between 01/04/2015 and 07/12/2015
Data Storage Requirements

In terms of data storage requirements, the total data size occupied for the 251 day data span is of 110,517,674 bytes for the events and 221,884 bytes for the consumption rates. Generalizing for yearly requirements, the total data storage size requirements per year are 160,712,953 bytes for the events and 322,660 bytes for the consumption rates.

2.4.3 Existing infrastructure, APIs and protocols (with focus on scale and reliability)
Most of the data gathered for the Portuguese pilot is already aggregated within the companies that provide it. Hence, there is no need for APIs or special infrastructures to access the data; rather, data is provided directly via a File Sharing app, such as Google Drive or Dropbox, as in the case of LS, or by Secure File Transfer Protocol folders, provided by IP. The only exception is the Traffic Events Web Service from IP.

2.4.3.1 IP Traffic Events Web Service
Data AccessMechanism
The mechanism to access data is based on the SOAP (Simple Object Access Protocol) Web service. The service provides daily information about traffic events that happen on the Portuguese road infrastructure.

Data Security and User Context
The Web service data is not public, requiring credentials handled by IP to access it.

Available Data Formats
The Web service data format is XML – (eXtendible Markup Language).

Resource Information

- **Response formats:** XML
- **Requires authentication?** Yes
- **Rate limited?** No
- **Requires Parameters?** Credentials
2.4.3.2 IP Secure File Transfer Protocol

Data Access Mechanism

For the case of Excel spreadsheets, CSV files and database dumps, IP opened a SFTP folder to provide the data. This is also valid for the concession holders’ data, namely Ascendi and ViaLivre toll sensors data.

Data Security and User Context

The access to the SFTP is restricted to project partners. Credentials were provided by IP.

Available Data Formats

The available data formats in the SFTP are:

- Excel spreadsheets;
- CSV files;
- SQL database dumps;

2.4.3.3 LS Data

Data Access Mechanism

LS has no defined protocol for sharing their data with the project. Data is coming in batches and shared through file sharing applications such as Dropbox and Google Drive. There are private folders shared among project partners to the effect.

Data Security and User Context

LS data is proprietary, and for exclusive use of the project.

Available Data Formats

The common data format is Excel spreadsheet format.

2.5 Integrated Car2X communication platform

2.5.1 Short description

Car2X pilot is focused on Adria Mobil’s intelligent motorhomes. The goal of the pilot is to optimize user experience – regarding length, comfort and overall convenience of the travel. To achieve our goals that are defined in use case scenarios, motorhomes were equipped with several sensors that provide data about internal state (living space, car) and location.
2.5.2 Data sources (with focus on speed and volume)

Two data sources were already made available by Adria Mobil since the start of this project. One of them refers to motorhome location data. Location data is available for each of 10 Adria’s rental motorhomes and can be reached in two ways. As:

- Real-time stream of GPS coordinates
- History of GPS coordinates

The same group of motorhomes, with additional 8 motorhomes is also in the process of being equipped with sensors that collect available OBD data.

The other (already available) data source is extended sensor data (60 sensors + actuators), that is measured in one motorhome.

2.5.2.1 Real time stream of GPS coordinates

Real time stream of GPS coordinates is exposed as a RESTful service. Data can be collected with a POST call to https://vps.blackblox.si/mobileapi/last_locations/, with username and password sent in body of the call. Data is returned in JSON format. It contains data of each motorhome’s last state.

<table>
<thead>
<tr>
<th>Type of data</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>Updates every 60 sec (when vehicle is moving)</td>
</tr>
<tr>
<td>Size (amount of data)</td>
<td>Last state of 10 motorhomes</td>
</tr>
<tr>
<td>Access to data</td>
<td>REST API</td>
</tr>
<tr>
<td>Processing the data</td>
<td>Data is retrieved and sent to mobility patterns service (nextPin). There, GPS locations are clustered into “paths” and “staypoints”.</td>
</tr>
<tr>
<td>Storage of the data</td>
<td>nextPin stores proccessed data in a Postgres database.</td>
</tr>
</tbody>
</table>

Relevant data sources

Relevant global context

This data is crucial for carrying out T1 in SmartAutohome use case scenario (described in D1.3)

Broker/middleware to be used

Example:

```json
{
    "result": [
        {
            "device_id": 6524,
            "label": "Twin 600",
            "lon": "15.18465166",
```
2.5.2.2 History of GPS data

Historical GPS data is also exposed as a RESTful service. It returns data for a specific motorhome at a specific date. Data can be collected with a POST call to https://vps.blackblox.si/mobileapi/device/day_history/, with username, password, vehicle ID and date attached in body of the call. Data is returned in JSON format.

<table>
<thead>
<tr>
<th>Type of data</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>Updates every 60 sec (when vehicle is moving)</td>
</tr>
<tr>
<td>Size (amount of data)</td>
<td>REST API</td>
</tr>
</tbody>
</table>

Access to data: REST API

Processing the data: Data was retrieved and sent to mobility patterns service (nextPin). There, GPS locations were clustered into “paths” and “staypoints”. Once the historical data was imported, we started to collect real time data.

Storage of the data: nextPin stored processed data in a Postgres database.

Relevant data sources

Relevant global context: This data is crucial for carrying out T1 in SmartAutohome use case scenario (described in D1.3)

Broker/middleware to be used
Example:

```json
{
    "result": [
        {
            "id": 852540762,
            "lon": "14.57411166",
            "lat": "45.98617",
            "alt": 256,
            "time": 1429605277,
            "speed": 0,
            "device_id": 7048,
            "day": "2015-04-21",
            "type": 0,
            "digital": 0,
            "distance": "5308.6321"
        },
        {
            "id": 852540764,
            "lon": "14.57413",
            "lat": "45.98621666",
            "alt": 256,
            "time": 1429605287,
            "speed": 2,
            "device_id": 7048,
            "day": "2015-04-21",
            "type": 0,
            "digital": 0,
            "distance": "5308.6375"
        }
    ]
}
```

**2.5.2.3 OBD data**

The same group of 10 motorhomes will also be equipped with OBD sensors. These sensors collect data for available OBD 2 fields. Currently the sensors are being installed into the motorhomes and will allow us to get the measurement every 30 seconds:

- 0x04 PID - Calculated engine load value
- 0x05 PID - Engine coolant temperature
- 0x0B PID - Intake manifold absolute pressure
• 0x0C PID - Engine RPM
• 0x0D PID - Vehicle speed
• 0x0F PID - Intake air temperature
• 0x10 PID - MAF air flow rate
• 0x21 PID - Distance traveled with malfunction indicator lamp (MIL) on
• 0x2C PID - Commanded EGR
• 0x2F PID - Fuel Level Input

2.5.2.4 Sensor feed
There is one motorhome that is equipped with approximately 60 sensors (in addition to GPS data also temperature, battery-, toilet-, door-, lights status, humidity levels etc.). Data visualization is available at:


A screenshot from the link above is visible in Figure 39.

![Graph](image)

Figure 39: Sensor data visualization. Here, only 7 variables were chosen to display.

Raw data can is available as a SQL dump.

<table>
<thead>
<tr>
<th>Type of data</th>
<th>SQL dump</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>Every second</td>
</tr>
<tr>
<td>Size (amount of data)</td>
<td>~35GB and growing</td>
</tr>
<tr>
<td>Access to data</td>
<td>Either SQL dump or direct queries to mySql DB.</td>
</tr>
<tr>
<td>Processing the data</td>
<td>Data is pushed from motorhome to <a href="http://mustang.ijs.si:8080/AdriaServer/">http://mustang.ijs.si:8080/AdriaServer/</a> in JSON format. From there it is saved to a mySql database.</td>
</tr>
<tr>
<td>Storage of the data</td>
<td>mySql database</td>
</tr>
<tr>
<td>Relevant data sources</td>
<td></td>
</tr>
</tbody>
</table>
Relevant global context
This data is crucial for carrying out T1 in SmartAutohome use case scenario (described in D1.3)

2.6 General data sources

2.6.1 Short description
There are several additional data sources that can contribute to multiple pilots indirectly. Some of them can be used in all pilots (like Foursquare), while others are still specific for a particular region (e.g. Ljubljana).

2.6.2 Data sources

2.6.2.1 Traffic events (Slovenia)
Slovenian traffic events are collected from open source URL http://opendata.si/promet/events/. Data is retrieved every few minutes. Using adapters, data is then stored in a Postgres DB. Cleaning is performed on this DB – obsolete traffic events are removed.

<table>
<thead>
<tr>
<th>Type of data</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>Updates when new information is available</td>
</tr>
<tr>
<td>Size (amount of data)</td>
<td>Varying, few KB</td>
</tr>
<tr>
<td>Access to data</td>
<td>REST API</td>
</tr>
<tr>
<td>Processing the data</td>
<td>Removing due events</td>
</tr>
<tr>
<td>Storage of the data</td>
<td>Postgres database.</td>
</tr>
</tbody>
</table>

Relevant data sources

Relevant global context
This data can be used in all use cases, that are related to routing (but data is limited to Slovenia)

Broker/middleware to be used

Example:

```json
{
   "dogodki": {
      "dogodek": [
         {
            "dovoljenjeDatKon": null,
            "y_wgs": 46.577010807845454,
            "kategorija": "A5",
            "zbrisano": null,
            "operaterSprememba": null,
            "isMejniPrehod": false,
         }
      ]
   }
}```
### 2.6.2.2 Parking spaces (Ljubljana)

There is data for Ljubljana’s parking spaces – number of regular parking spaces, number of parking spaces for disabled, prices, locations, availability etc. Data is available at [http://opendata.si/promet/parkirisca/lpt/](http://opendata.si/promet/parkirisca/lpt/).

<table>
<thead>
<tr>
<th>Type of data</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>Updates when new information is available</td>
</tr>
<tr>
<td>Size (amount of data)</td>
<td>Varying, few KB</td>
</tr>
<tr>
<td>Access to data</td>
<td>REST API</td>
</tr>
<tr>
<td>Processing the data</td>
<td></td>
</tr>
<tr>
<td>Storage of the data</td>
<td></td>
</tr>
</tbody>
</table>
### Relevant data sources

| Relevant global context | This data can be used in all use cases, that are related to routing (but data is limited to Ljubljana) |

### Broker/middleware to be used

#### Example:

```json
{
    "U_delovnik": "od 6.00 do 20.00 (ponedeljek – petek)",
    "Cena_splosno": null,
    "Ime": "Tivoli I",
    "A_St_Mest": null,
    "Invalidi_St_mest": 2,
    "Cena_ura_Eur": null,
    "KoordinataX": 461615,
    "Opis": "dnevna tarifa od 6,00 ure do 20,00 ure; prvi dve uri 0,60 €, vsaka naslednja ura 0,60 €\n\nparkirnina avtobus; dnevna tarifa od 6,00 ure do 20,00 ure; 4,80 € /uro",
    "ID_Parkirisca": 3,
    "U_splosno": null,
    "zasedenost": {
        "Cas": "2016-03-22 14:03:00",
        "ID_ParkiriscaNC": 3,
        "Cas_timestamp": 1458651780,
        "P_kratkotrajniki": 65
    },
    "U_sobota": null,
    "St_mest": 360,
    "KoordinataX_wgs": 14.498993762396804,
    "Cena_dan_Eur": null,
    "ID_ParkiriscaNC": 3,
    "KoordinataY": 101628,
    "Cena_mesecna_Eur": null,
    "KoordinataY_wgs": 46.057723461467276,
    "Upravljalec": "JP LPT d.o.o.",
    "Tip_parkirisca": "Avtomatizirano"
}
```

#### 2.6.2.3 Bicycle sharing (Ljubljana)

Another data source available for Ljubljana is the data from bicycle sharing (Bicikelj) stations. The idea of bicycle sharing was very well received by locals. Number of stations is still increasing
since the beginning. Now, even some of the parking lots in the periphery of the city have Bicikelj stations.

The data is available at [http://opendra.si/promet/bicikelj/list/](http://opendra.si/promet/bicikelj/list/) and includes the address of the station, location, number of available bikes, number of free (“parking”) spaces, time of update etc.

<table>
<thead>
<tr>
<th>Type of data</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>Updates when new information is available</td>
</tr>
<tr>
<td>Size (amount of data)</td>
<td>few KB</td>
</tr>
<tr>
<td>Access to data</td>
<td>REST API</td>
</tr>
</tbody>
</table>

### Processing the data

### Storage of the data

### Relevant data sources

### Relevant global context

This data can be used in all use cases, that are related to routing (but data is limited to Ljubljana)

### Broker/middleware to be used

**Example:**

```
"updated": 1458649417,
"bonus": "0",
"timestamp": "2016-03-22T13:23:37.577Z",
"fullAddress": "PREŠERNOV TRG-PETKOVŠKOVO NABREŽJE",
"number": 1,
"address": "PREŠERNOV TRG-PETKOVŠKOVO NABREŽJE",
"lat": "46.051367",
"lng": "14.506542",
"open": "1",
"name": "PREŠERNOV TRG-PETKOVŠKOVO NABREŽJE",
"station_valid": true,
"station": {
  "available": "15",
  "ticket": "0",
  "total": "20",
  "free": "5"
}
```
2.6.2.4 Loop sensors (Slovenia)

There is approximately 250 loop sensors on Slovenian roads. Their data is available at [http://opendata.si/promet/counters/](http://opendata.si/promet/counters/). The data includes location, region, current traffic status, occupancy, speed, flow and gap between consecutive vehicles.

This is the same data that we use for traffic prediction – 3, 6, 9, 12 and 15 hours in advance. Predictions are made online (take into account current road status) and cover traffic status, speed, flow and occupancy (visualization available at [http://traffic.ijs.si/](http://traffic.ijs.si/)).

<table>
<thead>
<tr>
<th>Type of data</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>Updates once per hour</td>
</tr>
<tr>
<td>Size (amount of data)</td>
<td>Last status</td>
</tr>
<tr>
<td>Access to data</td>
<td>REST API</td>
</tr>
<tr>
<td>Processing the data</td>
<td>This data is used for online traffic predictions</td>
</tr>
<tr>
<td>Storage of the data</td>
<td>Historical data also available as JSON dumps (~3 years of data)</td>
</tr>
</tbody>
</table>

**Relevant data sources**

**Relevant global context**

This data can be used in all use cases, that are related to routing (but data is limited to Slovenia).

**Broker/middleware to be used**

**Example:**

```json
{
"stevci_vmax": 100,
"stevci_smerOpis": "Barjanska - Peruzzijeva",
"id": "0178-22",
"stevci_ura": "14:25:00",
"stevci_gap": "999,9",
"georss_point": "463074 097385",
"stevci_datum": "22/03/2016",
"stevci_cestaOpis": "AC-A1",
"stevci_geoY_wgs": 46.01962915242322,
"stevci_regija": "Ljubljana",
"updated": 1458649500,
"stevci_stacionaza": 2.29,
"stevci_lokaciijaOpis": "LJ (južna obvoznica)",
"stevci_odsek": 18,
"stevci_stev": 0,
"stevci_smer": 22,
```


2.6.2.5 Location data providers

We currently use Foursquare API (reference, documentation) to obtain location suggestions based on coordinates when detecting frequent locations with mobility patterns detection service. Later on, we can use this API when providing recommendations to Optimum users. In addition to Foursquare, if there is need for it, we may start using also other location data providers such as Factual, Google places or Facebook places.

However, for now we only use Foursquare. Developer’s API enables several “intent” modes: browsing for venue, trying to check-in, matching venues and finding most globally relevant venues.

<table>
<thead>
<tr>
<th>Type of data</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>On demand</td>
</tr>
<tr>
<td>Size (amount of data)</td>
<td>Last status</td>
</tr>
<tr>
<td>Access to data</td>
<td>REST API. Client id and Client secret are needed (provided by Foursquare)</td>
</tr>
</tbody>
</table>

Processing the data

Storage of the data

Relevant data sources

Relevant global context

This data can be used in all use cases that either need venue suggestion or mapping from coordinates to actual venues

Broker/middleware to be used

```json
"stevci_stat": 6,
"stevci_geoX_wgs": 14.51818035950113,
"stevci_occ": 0,
"stevci_geoY": 97385,
"stevci_geoX": 463074,
"stevci_statOpis": "Ni prometa",
"stevci_hit": 0,
"summary": "AC-A1, LJ (južna obvoznica) : Barjanska - Peruzzijeva (p) - Ni prometa (0 vozil/h, povp. hitrost=0km/h, povp. razmik=999,9s, zasedenost=0,0%)",
"stevci_pasOpis": "(p)",
"stevci_lokacija": 178,
"title": "AC-A1, LJ (južna obvoznica) : Barjanska - Peruzzijeva (p)"
},
```
Example:

```
"venues": [
    {
      "id": "4b852c92f964a520534f31e3",
      "name": "IJS (Institut Jožef Stefan)",
      "contact": {},
      "location": {
        "address": "Jamova 39",
        "lat": 46.042399683906304,
        "lng": 14.487873745203604,
        "distance": 72,
        "postalCode": "1000",
        "cc": "SI",
        "city": "Ljubljana",
        "state": "Ljubljana",
        "country": "Slovenia",
        "formattedAddress": [
          "Jamova 39",
          "1000 Ljubljana",
          "Slovenia"
        ]
      },
      "categories": [
        {
          "id": "4bf58dd8d48988d19f941735",
          "name": "College Technology Building",
          "pluralName": "College Technology Buildings",
          "shortName": "Technology",
          "icon": {
            "prefix": "https://ss3.4sqi.net/img/categories_v2/education/collegeacademicbuildings_technology_",
            "suffix": ".png"
          },
          "primary": true
        }
      ],
      "verified": false,
      "stats": {
        "checkinsCount": 2248,
        "usersCount": 162,
        "tipCount": 2
      }
    }
  ]
```

2.6.2.6 Collection app library

For purposes of mobility patterns detection service we developed mobile app libraries for Android and iOS that collect location data from mobile phone and send it to our personal mobility patterns server. The data that is collected is latitude, longitude, accuracy, activity (walking, cycling, and driving) and wi-fi signals around the location. We developed an app (for both OS) around this library, which allows users to see a very simple visualization of returned results. However, libraries can be used independently in any mobile app, particularly we have in mind Optimum app.

Each user (owner of mobile phone with app installed) can also see his/her data in web app, but they need their token to access the data there.

<table>
<thead>
<tr>
<th>Type of data</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>Every 30 seconds</td>
</tr>
<tr>
<td>Size (amount of data)</td>
<td>Latitude, longitude, accuracy and time</td>
</tr>
<tr>
<td>Access to data</td>
<td>Processed data as Postgres dump (for own data)</td>
</tr>
</tbody>
</table>
Processing the data

Mobility patterns detection service clusters raw data into staypoints and path

Storage of the data

Processed data is stored in Postgres DB.

Relevant data sources

Relevant global context

This data can be used in all use cases that either need venue suggestion or mapping from coordinates to actual venues

Broker/middleware to be used

Example:

```json
{"locations": [
  {
    "latitude":53.00847,"longitude":1.71399,"time":1458743699000,
  },
  "user":"TT5576"
]
```

2.6.2.7 Weather (Slovenia)

Weather data can be used traffic predictions or taken into account in providing recommendations to users. This historical data is obtained from ARSO (Slovenian agency for environment) and is exposed through a RESTful API. There is also a node.js module called ‘arso-weather’ which makes it even easier to obtain the data. Data includes information about rainfall, wind, pressure and other features, collected every half an hour.

<table>
<thead>
<tr>
<th>Type of data</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>On demand</td>
</tr>
<tr>
<td>Size (amount of data)</td>
<td>Data for more than 6 years</td>
</tr>
<tr>
<td>Access to data</td>
<td>REST API</td>
</tr>
<tr>
<td>Processing the data</td>
<td></td>
</tr>
<tr>
<td>Storage of the data</td>
<td></td>
</tr>
<tr>
<td>Relevant data sources</td>
<td></td>
</tr>
<tr>
<td>Relevant global context</td>
<td>This data can be used in Slovenian use-cases</td>
</tr>
<tr>
<td>Broker/middleware to be used</td>
<td></td>
</tr>
</tbody>
</table>
Example:

```json
[
  { ts: 1271023200,
    dateTime: '2010-04-12 00:00',
    p: '980',
    pmin: '980',
    pmax: '980',
    t2m: '4.3',
    t2mmin: '4.2',
    t2mmax: '4.4',
    rh: '82',
    rhmin: '79',
    rhmax: '88',
    t2m_termin: '4.2',
    rh_termin: '82',
    padavine: '0',
    veter_hitrost: '0.2',
    veter_vek_smer: '85',
    veter_max_hitrost: '1.3',
    energija_gl: '0',
    energija_di: '0' },
  { ts: 1271025000,
    dateTime: '2010-04-12 00:30',
    p: '980',
    pmin: '980',
    pmax: '980',
    t2m: '4.2',
    t2mmin: '4.2',
    t2mmax: '4.3',
    rh: '82',
    rhmin: '80',
    rhmax: '88',
    t2m_termin: '4.3',
    rh_termin: '82',
    padavine: '0',
    veter_hitrost: '0.2',
    veter_vek_smer: '351',
    veter_max_hitrost: '0.9',
    energija_gl: '0',
    energija_di: '0' }
]
2.6.2.8 Weather (global)
Forecast.io provides historical weather data and weather predictions (minute-by-minute, hour-by-hour, day-by-day, current conditions) for specific location. Both are easily accessible through a REST API. However, 1000 requests per day are free of charge, every additional request is billed $0.0001.

<table>
<thead>
<tr>
<th>Type of data</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>On demand</td>
</tr>
<tr>
<td>Size (amount of data)</td>
<td>Some locations up to 60 years</td>
</tr>
<tr>
<td>Access to data</td>
<td>REST API</td>
</tr>
</tbody>
</table>

Processing the data
Storage of the data
Relevant data sources
Relevant global context This data can be used in all use-cases
Broker/middleware to be used

Example (current location and predictions):

```json
{
  "latitude": 46,
  "longitude": 14,
  "timezone": "Europe/Ljubljana",
  "offset": 2,
  "currently": {
    "time": 1460468329,
    "summary": "Clear",
    "icon": "clear-day",
    "precipIntensity": 0,
    "precipProbability": 0,
    "temperature": 59.63,
    "apparentTemperature": 59.63,
    "dewPoint": 45.09,
    "humidity": 0.59,
    "windSpeed": 9.71,
    "windBearing": 239,
    "visibility": 6.21,
    "cloudCover": 0.01,
    "pressure": 1011.74,
    "ozone": 333.76
  },
  "hourly": {
```
| "summary": "Partly cloudy starting tonight, continuing until tomorrow afternoon.", |
| "icon": "partly-cloudy-day", |
| "data": [ |
|   { |
|     "time": 1460466000, |
|     "summary": "Clear", |
|     "icon": "clear-day", |
|     "precipIntensity": 0, |
|     "precipProbability": 0, |
|     "temperature": 60.12, |
|     "apparentTemperature": 60.12, |
|     "dewPoint": 45.56, |
|     "humidity": 0.59, |
|     "windSpeed": 10.04, |
|     "windBearing": 239, |
|     "visibility": 6.21, |
|     "cloudCover": 0.02, |
|     "pressure": 1011.86, |
|     "ozone": 335.85 |
|   }, |
| ... |
| ]}
2.6.2.9 Video cameras (Slovenia)

Video data from DARS (Slovenian highway agency) is available for Slovenian highways and some regional roads. OpenCV (Open source computer vision library) or some other tool can be used to perform video and image analysis.

<table>
<thead>
<tr>
<th>Type of data</th>
<th>JPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>2 frames per minute</td>
</tr>
<tr>
<td>Size (amount of data)</td>
<td>Real time stream</td>
</tr>
<tr>
<td>Access to data</td>
<td>REST API</td>
</tr>
</tbody>
</table>

Processing the data

Storage of the data

Relevant data sources

Relevant global context

This data can be used in Slovenian use-cases

Broker/middleware to be used

Example:

http://kamere.dars.si/kamere/Sentvid_Jug/cam11.jpg
3 Proposed data infrastructure – conceptual design

3.1 Introduction – the need for Edge processing

There are many complexities involved in gaining the type of insights that businesses increasingly require from the transportation-related data, especially those which belong to IoT, since there are several challenges of executing analytics on IoT data from a centralized environment. Some of the challenges are caused by the nature of the data itself and the physical environment where the data resides. Other challenges are related to how to protect highly sensitive data. In addition, important issues are related to latency, and the overall complexity of the environment. In the following we elaborate on the most important ones, which should be reflected in the data infrastructure architecture:

Data infrastructure is highly fragmented
A transportation environment is typically comprised of a myriad of sensors and devices communicating over non-standard protocols that are difficult to integrate and manage. Additionally, wireless mesh sensors are often capable of running for years on a single battery by requiring small amounts of power and not connecting directly to the Internet.

Latency and inconsistent connectivity can be inhibitors
Solutions often require rapid data insights and control responses. Typically, the required speed of reaction can’t be achieved if latency is introduced from sending data and application calls between remote devices and centralized systems.

Data movement and storage can be costly
Many devices, sensors and networks are connected to the Internet and create and receive data around the clock. This generates tremendous amounts of data that must be transferred to a location for storage and analysis. As the number of devices expands and the volume of data increases, the costs of data transport and data storage can quickly become prohibitive.

Connecting infrastructure and devices can introduce security risks
Security is paramount to any data-driven solution. While isolation is certainly a way to avoid risk, it also prevents the system from taking advantage of the value of external data feeds and tapping into even more powerful remote processing to supplement local analytics capabilities. In addition, many connected devices that collect and transfer data to a centralized repository lack the capability to deploy sophisticated security controls and safeguards.
3.2 The benefits of Edge processing

Edge processing is critical in transportation because of three fundamental requirements: low latency, maintaining user privacy, and pooling of resources at different layers.

The example of low latency: a mesh of edge nodes (mobile gateways) in an intelligent traffic control system can share the collected traffic information to streamline traffic during peak hours, localize accidents, and re-route traffic away from congested traffic areas.

Similarly, the pooling of resources can be used in infotainment systems where fog-based applications on each user’s phone and in public transport allow the users to share and stream downloaded content from nearby users without a persistent network connection.

Finally, safety systems for automated vehicles, surveillance systems on the roads, and ticketing systems in public transport can collect a lot of information in terms of sensor and video data. These systems ideally should only communicate aggregated data to the cloud to maintain user privacy and conserve user bandwidth smartly. The cloud can extract useful business insights such as where to plan the best routes in longer time scales, but is unlikely to provide the latency guarantees in short time scale. The analytics are distributed across edge and cloud to enable real-time decision making at the edge, and policy control and data insights driven from the cloud.

3.3 Edge processing - elements

As already elaborated, in order to enable more reactivity close to the data sources, we argue that the data processing infrastructure should be compatible with the modern distributed Cloud architectures, esp. with those that support local processing, so called Fog computing architectures. In particular, data processing infrastructure should be aligned with Edge computing/processing paradigm.

Following figure shows three distinct layers in the architecture in which the part of the processing is pushed closer to the clients’ infrastructure or data sources (to the “edge”). It shows three distinct layers in the architecture in which the part of the processing is pushed closer to the clients’ infrastructure or data sources (to the “edge”).
3.3.1 Layers

3.3.1.1 Data sources
As the name says, this is the layer where the data sources are concentrated, likely sensors, measurements, machine outputs and so on. In general, we can divide them into two types, noticeably different based on the way they transmit data to the edge cloud, but in essence the data gets consolidated and processed in a similar or same fashion later on. This is also important to show the diversity of possible inputs and the variety of protocols.

3.3.1.1.1 (W)LAN
These sources can be easiest described as those utilizing the TCP/IP stack, and they are connected using WiFi or Ethernet on the data OSI layer. Protocols used here can be CoAP (constrained application protocol) or MQTT, which are well known in the IoT world, or something like OPC (Open Platform Communications) typical for industrial and manufacturing systems. The examples do not have to stop here, any kind of TCP/IP application protocol can be used, like HTTP.
We’ll give just a short summary of the most popular protocols from these categories and the typical usage:

- **CoAP (Constrained Application Protocol)** can be easiest described as HTTP over UDP. TCP, which is the native transport layer protocol that HTTP uses, generally has the connection and reliability features which may introduce drawbacks in constrained networks. That is why CoAP is designed, to work over the more lightweight (although less reliable) UDP transport layer, which makes it suitable for low powered devices operating within a relatively reliable network. Other than that, it is quite similar to HTTP, works in a client-server, request-response mode and has the same most important verbs like GET, POST, PUT and DELETE. It is suitable for moderately frequent messaging.

- **MQTT** is a publish/subscribe protocol, meaning it needs a broker and works on topics and in general supports many-to-many communication. Topics are arbitrary literal strings, but also hierarchical subscription can be used. Wildcards are allowed when registering a subscription (but not when publishing) allowing whole hierarchies to be observed by clients. For example, in a home automation setup, topic `/sensors/kitchen/*` would subscribe to all the sensors from the kitchen or `/sensors/*/temperature` would subscribe to only temperature sensors from all rooms.

- **OPC (Open Platform Communications)** is a set of standards for industrial M2M uses. Previous generations were characterized by multiple standards like OPC-DA for data access (the most used one), OPC-HAD for historical data access and so on. They were based and limited on Microsoft OLE, COM and DCOM technologies, however some connectors can be found for Java for OPC-DA and the new OPC Unified Architecture (successor to the OPC-DA) is planned as multiplatform. OPC-DA (or OPC-UA) works in a request-response model, however there are plans on implementing a pub/sub feature with an AMQP under-layer and a central hub. With OPC-UA there is a subscription option, however without a broker so the subscription is between two points only, making the connection push-based instead of request-response. The OPC stack is usually implemented by commercial vendors so there is not much documentation, training, tutorials, software or tools available freely, which makes sense as it is designed for industrial applications where a degree of support is needed. However, that makes it the unobvious choice when starting a new project if not previously experienced with it.

**3.3.1.1.2 LR-WPAN**

LR-WPAN (Low-Rate Wireless Personal Networks) are wireless networks with very low power demands often used for communication between simple, also low-powered devices, usually powered with batteries. In the IoT world they are used to implement sensor meshes or arrays,
between microcontroller boards such as Arduino and so on. The mesh topology is very important with sensors on the move, such as kettle collars, agricultural sensors etc. Concrete implementations usually are built upon the 802.15.4 standard and those include ZigBee, ZWave and etc. Higher level application protocols do not apply here, with exceptions for some like MQTT-SN (MQTT for Sensor Networks) but they are not as popular, so usually a custom protocol is used, tailored for the use case.

### 3.3.2 Edge cloud

In the figure, and in our vision, the edge cloud is implemented as a kind of IoT hub, with hardware being similar to Raspberry Pi, Beaglebone or some similar embedded device. Devices like that are powerful enough to execute simpler algorithms, serve as a hub for all of the wired or wireless sensors around the plant and transmit data to the online layer. The edge cloud directly should only display simple console output to the client, or eventually a simple web UI for initial setup or debugging. All of the processing results are reported to the online cloud where they are presented in a rich UI. The most important features covered are:

- **Local sensor/IoT Hub**
  
  Sensors from around the main hub send their readouts to this device. Their operational status and health is checked from this point.

- **Adaptation**
  
  Adaptation from different protocols the sensors are using to a narrower set of data formats and fewer internal protocols is done on this layer.

- **Embedded Complex Event Processing (embedded CEP)**
  
  The edge cloud devices should have an embedded CEP, locally tracking different variables and events for a situation of interest.

- **Filtering**
  
  Values heavily contaminated with noise or out of range, which may be caused by a faulty sensor, should be filtered out at this point.

- **Anomaly Detection**
  
  Anomaly detection algorithms on the incoming measurements could be executed on this level, anomalies are reported to the layer up, to the online cloud for further analysis.

- **Batching**
  
  Measurements from the sensors don’t have to be immediately pushed to the online cloud for further analysis. They can be locally stored, batched, and then compressed and uploaded on a
regular interval or when a condition is satisfied. This may offload the central cloud and promote low power consumption.

Since these are low powered devices they can be stored in places without special cooling or power requirements, and with low maintenance cost for the client.

- **Triggers**

The edge cloud devices may execute some action based on simple triggers both from the upper online cloud layer or lower data sources. This should not be mixed with the CEP which has patterns deployed for complex processing. Triggers may be some simple events that can schedule maintenance, initialize upload or emit a sound warning.

On-line cloud is out of scope of this deliverable.

### 3.4 Data infrastructure: Conceptual architecture

Based on the analysis of the Data sources described in the Section 3, in this subsection we provide the conceptual architecture for the Edge Computing.
From the Figure 41 above, we can see that the basic idea is that all the relevant data-feeds are being pushed and/or pulled into our publish/subscribe middleware. According to the edge paradigm, as much as possible should be calculated/prepared already on the devices. While this benefits the load of the infrastructure, the architecture itself doesn’t require it. It works the same, whether the data coming in is in a raw format, or already has more complex structures such as detected complex events, aggregates, etc. Depending on the data-source, it is being pulled or pushed into the middleware, where only the subscribers will get the relevant information, while the rest will be filtered out. Optimum backend infrastructure and our watchdog services are subscribed to all the events. One for functionality and the watchdog for being able to monitor all the data-sources and pipelines that are required for data to be coming in. Similarly also the users/other Optimum components, or external partners can subscribe to the feeds in a similar way as our backend system. For the non-real time information, the data subscribers, can query directly backend.

While all the data-sources are being pushed through middleware, it is possible also to directly pull them, or subscribe directly to the feed if necessary and the feed adapter allows this. This brings us the flexibility to construct a distributed infrastructure, where publish/subscribe middleware doesn’t need to be only one machine/ one point of entry. Depending on the server load, we will increase the number of pub-sub nodes according to the need of the project.

### 3.5 Data retrievers, watchdog

Good quality and stability of the data feeds is actually the most crucial component if we want to accomplish the goals of the project. This was also our experience from other machine learning and traffic related projects. For this reasons we decided to put a lot of focus on the stable collection and provision of various data feeds. Our experience speaks that there should be at least one full time person responsible for data collection and maintenance, which can easily be the most expensive operation through the project. For this reason, to bring the costs of the maintenance down and increase the stability, we are proposing a data watchdog. The watchdog should at least automatize monitoring/reporting and re-starting of the data-feeds in cases when something goes wrong. We already have quite many data retrievers currently running and the number will only increase and so will increase the maintenance load and complexity.

The watchdog should be able to monitor data retrievers directly (retrievers can be written in any programming language) or through the publish/subscribe mechanism, where it monitors retrieved events.
At startup watchdog runs all retrievers that are included in its repository. After each successful retrieval, retrievers must send a signal to watchdog. If the watchdog doesn’t receive a signal from a particular retriever too long it will automatically restart it. Implementation of the method that pings watchdog is obligatory to achieve desirable results. Optionally, retrievers can in similar fashion send signals about the number of objects received and error that might have occurred. It is also very easy to connect new retriever to Pingdom\textsuperscript{11} once it was included in watchdog repository. Visualization of the current watchdog’s structure can be seen in Figure 42.

Watchdog (currently in development version) has a simple admin console (Figure 43:). There, all retrievers are listed, with information where (if at all) the data is sent, number of objects received and last data updates. If any error occurred it is reported in column “connection”. It is also possible to change settings of retrievers that are written in Java and extend a class Retriever that was created for that purpose. That enables to change/add hosts and different parameters, depending on configurations.

![Figure 42: Watchdog](image)

<table>
<thead>
<tr>
<th>Retriever</th>
<th>Host</th>
<th>Port</th>
<th>Parameters</th>
<th>Connection</th>
<th>Objects received</th>
<th>Last data update</th>
<th>Change settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoordsRetriever</td>
<td>localhost</td>
<td>8001</td>
<td>1456847044000</td>
<td>2009</td>
<td>03/01/2016 16:44:04</td>
<td>Change settings</td>
<td></td>
</tr>
<tr>
<td>TrafficEventsRetriever</td>
<td>localhost</td>
<td>120</td>
<td>145684710966</td>
<td>0</td>
<td>03/01/2016 16:46:39</td>
<td>Change settings</td>
<td></td>
</tr>
<tr>
<td>MotorhomeStatusRetriever</td>
<td>localhost</td>
<td>8001</td>
<td>1456823066000</td>
<td>10</td>
<td>03/01/2016 11:42:46</td>
<td>Change settings</td>
<td></td>
</tr>
</tbody>
</table>

![Figure 43: Information about data retrievers currently running](image)

This watchdog is still in early stages of development. The goal is to develop a web service where users will be able to simply add new retrievers, change settings etc. solely with interaction through GUI.

### 3.6 Support for standard big data paradigm

Currently all the data infrastructure and architecture plans are built with the idea to be used in the big-data scenarios and be scalable according to the big-data paradigm. All the services/tools and data/communication routings in between are easily customizable and easy to replicate

\textsuperscript{11} https://www.pingdom.com/
through servers. The publish/subscribe mechanism and edge computing paradigm both contribute to that as well, additionally to standard big data tools such as Hadoop, Spark and Storm that we are ready to employ when necessary.

While big-data paradigm such us map-reduce (Hadoop, etc.), allow speed ups and scalability gains for some problems, it’s not a “solve it all” solution and it brings its own overheads. For these reasons we decided to take care that our infrastructure will support such systems and allow us to easily employ it, but will not pursue these techniques before we run into the scalability problems. At this point we don’t know yet all the bottlenecks and the full scale of our real-time data feeds, so the approach is to be able to distribute any part of the infrastructure over the cluster of machines when the bottleneck will be identified.
4 Implementation plan

Most of the data infrastructure is running as a prototype, or near to be running and deployed already at the time of writing this deliverable. We have in place some of the data-retrievers, together with a decent window of historical data, which will be crucial for machine learning and analytics tasks later. Also, additionally, we had to develop services and converters for some of the consortium partners, who didn’t have mechanisms to share their data before the start of the project (Adria, LPP). The existing infrastructure (developed since the beginning of the project), is distributed and running at:

- **Project Data providers**
  - LPP Servers. Their infrastructure didn’t support safe and reliable data-sharing, so we installed Optimum data server software on the replicated LPP databases in their premises. This allows us to gather the real-time information about the busses, stations, routes and even their existing predictions of arrivals.
  - Adria embedded devices. Aligned with the Edge computing paradigm, some of the data-collection and simple aggregation is happening on the embedded devices in their motorhomes. The devices are sending the collected data to JSI Optimum data servers.

- **Project Data Collection.** Distributed across countries to reduce load on the servers and consortium partners.
  - JSI Servers. We have a data forwarding server that collect traffic related information for Slovenia (LPP bus data, Adria motorhomes, UK traffic events, tweets, Slovenian traffic events, Loop sensors covering all Slovenia, etc). These are being forwarded or requested by Optimum publish/subscribe infrastructure.
  - Uninova server. Similarly as JSI, also Uninova has an instance of data forwarding server, covering mostly Portugal data-feeds.

- **Intrasoft hardware Cluster.** Which is running virtual machines with data-bases for Optimum backend. Its machines are also employed to do parts of the distributed data collection, currently having historical and real-time social data feeds.

Aligned with the current developments and state of the platform, we will continue according to the plan. WP2 development will continue to follow the stages as shown in Figure 44. Each step can be mapped to one (or two) Tasks defined in WP2.
As part of the first stage, we will continue to collect all the data that is/or might be relevant for the project, clean it and store it for later retrieval and access through publish/subscribe middleware. This stage corresponds to tasks T2.2 and T2.3. In some cases, data will be stored in raw format (quick to implement), in other cases (when the raw format is not enough) the data will already be cleaned and preprocessed before storage. But at this stage, the focus should not be yet on the data format, but more on the coverage and actually having the access to the data. Once the data sources that we’re actually going to use will be selected, cleaning and simple enrichment will be performed on their feeds in real time. Indexing will be implemented and querying will be enabled so that data will be conveniently available for follow-up tasks and the project consortium in general. This covers all possible sources of the data (multi-modal), social and other textual feeds included. For textual feeds, which will deem relevant for the project, indexing, querying and online feature extraction will be in place, which will allow machine learning tasks and further analysis with as low overhead as possible. The timeline and additional details for the development plan are listed in tables Table 13, Table 14, Table 15 and
Table 16 respectively.

Table 13: T2.2 timeline

<table>
<thead>
<tr>
<th>T2.2</th>
<th>Description</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.1</td>
<td>Finish data collectors, adapters that are missing and start collecting raw sensor streams. Consortium partners can start testing available data-feeds.</td>
<td>28 May 2016 (M13)</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Working first version of the sensor feed infrastructure - Data will be fully accessible to other consortium partners.</td>
<td>31 July 2016 (M15 – D2.2)</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Select data sources to be used. Here we will identify the sources according to usability. Some of the data feeds are more useful than others and will get more attention, since we simply don’t have enough resources to cover all of them completely.</td>
<td></td>
</tr>
<tr>
<td>2.2.4</td>
<td>Cleaning, storing and simple enrichment of the relevant data</td>
<td></td>
</tr>
<tr>
<td>2.2.5</td>
<td>Implementing indexing and enabling querying</td>
<td></td>
</tr>
<tr>
<td>2.2.6</td>
<td>Working final version of the sensor feed infrastructure</td>
<td>31 August 2017 (M28 – D2.6)</td>
</tr>
</tbody>
</table>

Table 14: T2.3 timeline

<table>
<thead>
<tr>
<th>T2.3</th>
<th>Description</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.1</td>
<td>Initial wrappers and data collectors should be in the first version of the infrastructure</td>
<td>28 May 2016 (M13)</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Identify data modalities (data types) that are relevant for Optimum. Also what kind of social feeds should we collect – mostly will be traffic related.</td>
<td></td>
</tr>
<tr>
<td>2.3.3</td>
<td>Working first version of the social data feed infrastructure.</td>
<td>31 July 2016 (M15 – D2.3)</td>
</tr>
<tr>
<td>2.3.4</td>
<td>Select data sources to be used, and processed through keyword filters and</td>
<td></td>
</tr>
</tbody>
</table>
In second stage, all the relevant feeds and storage will begin to converge towards the extended standard Datex II compliant form. When this will not be possible we will at least use the same vocabulary and build upon the existing Datex II nomenclature. In the third stage, additional data harmonization methods and services will be implemented, which will allow us to align all the data feeds into similar time frequencies and normalized sizes, so the additional analysis and usage will be as easy as possible. Based on selected data sources and requirements a list of needed methods and tools will be formed. For the first version, the first set of enrichment and alignment methods will be introduced. In the second version, more sophisticated data enrichment and alignment of data streams will be added. These two stages match T2.4.

Table 15: T2.4 timeline

<table>
<thead>
<tr>
<th>T2.4</th>
<th>Description</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4.1</td>
<td>Review the T2.1 and T2.3 and prepare the initial plan and requirements for fusion and enrichment</td>
<td>31 January 2016 (Done)</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Datex II conversion</td>
<td></td>
</tr>
<tr>
<td>2.4.3</td>
<td>Initial methods for enrichment and alignment should be added to the Optimum data infrastructure</td>
<td>31 June 2016 (M14)</td>
</tr>
<tr>
<td>2.4.4</td>
<td>Define list of needed methods and tools based on 2.2.3 and 2.3.4</td>
<td></td>
</tr>
<tr>
<td>2.4.5</td>
<td>First version of the methods in the infrastructure from T2.3 and T2.2</td>
<td>31 October 2016 (M18)</td>
</tr>
<tr>
<td>2.4.6</td>
<td>Non-trivial data enrichment and alignment of data streams</td>
<td></td>
</tr>
<tr>
<td>2.4.7</td>
<td>Final version of the alignment and enrichment methods in the infrastructure from T2.2 and T2.3</td>
<td>31 October 2017 (M30)</td>
</tr>
</tbody>
</table>

Fourth stage corresponds to T2.5 – here, data operators will be provided and database will be optimized for further use in high level analytics and predictions in stage 5, which is already out of WP2 scope.
D2.1 - Report on Data Infrastructure Architecture and Implementation Plan

Table 16: T2.5 Timeline

<table>
<thead>
<tr>
<th>T2.5</th>
<th>Description</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5.1</td>
<td>Closely follow the developments on T1,T2,T3,T4 to get an idea about the needed optimizations</td>
<td>Throughout all the project</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Review the workings of the data infrastructure and decide on needed operators</td>
<td>31 December 2015 (Done)</td>
</tr>
<tr>
<td>2.5.3</td>
<td>Optimization plan for the data infrastructure, based on the current state of the results from T2,3,4.</td>
<td>31 June 2016 (M14)</td>
</tr>
<tr>
<td>2.5.4</td>
<td>First (simple) data operators accessible through a library and an API</td>
<td>31 June 2016 (M14)</td>
</tr>
<tr>
<td>2.5.5</td>
<td>Initial tests of the data operators on the working infrastructure</td>
<td>31 October 2016 (M18 D2.5)</td>
</tr>
<tr>
<td></td>
<td>Improve and influence the performance of the data infrastructure and initial version of the working operators.</td>
<td>31 October 2016 (M18 D2.5)</td>
</tr>
<tr>
<td></td>
<td>Final version of database operators and optimized database</td>
<td>31 October 2017 (M30 D2.9)</td>
</tr>
</tbody>
</table>

More detailed architecture considering development by tasks and with the data operators (implemented as QMiner library) is presented in the Figure 45.

Each technical partner will support pilot partners in their country in terms of development of sensor and protocol adapters. Meaning, JSI will support Slovenian pilots, Uninova Portuguese pilots and so on. Both sensor and protocol adapters will be integrated inside data retrievers. These can be running in a distributed manner, or on central server on both. The data watchdog will be monitoring each data-source, either directly or through publish/subscribe infrastructure.

The data-harmonization, cleaning and alignment methods will be mostly implemented as part of open-source QMiner platform. It is being developed across various machine learning projects (commercial and research). It is already getting initial versions of data harmonization methods for various activities, e.g. replacing missing values, duplicated values, resampling etc. Additionally we have some of the data operators already in place (e.g. stream aggregates, moving averages). Others are actively being added and over the course of the project (e.g. staypoint detection algorithm). The resulting library will be used for online stream analytics (T3.5) but it can be also used for purposes of offline analysis.
Figure 45: Architectural diagram of WP2

Since the infrastructure allows it and because of specifics of particular components, deployment of the data operators library can/will be distributed on its own, or combined with big-data tools such as Hadoop, Spark, Storm, etc.
The central cluster is stationed at Intrasoft premises. Several components will be deployed and run there. For the other components, the central server will just serve as a central access point, but the actual components will run elsewhere (other partners, external components, etc.). The reason for that is that some components might have proprietary issues or the deployment itself is so specific that it would be too much overhead to try and deploy it in central cluster.

Each component that will be exposed through external API will have additional data-access layer on top of it. This layer will allow for security and control over who has access to data. This way we can control and react to any unexpected behavior by users. We can also enable different access levels for particular user or, analogously, different access level for particular set of data. This is planned to be done by re-using the custom admin and data access tool being built for LPP data access due to privacy issues and server capacity. This tool is currently being developed and is planned to be ready on M15. The data access and administration tool is described in more detail in D1.4 (Conceptional Architecture) under 5.5.1 External data Access Web Services.
5 Conclusion

This deliverable is the basis for the development of the data infrastructure that will enable a continuous feeding of the Platform with the data from various sources. This process will encompass the gathering (availability) of data from data sources and its preparation (cleaning) for the further processing.

There are two main advantages from the technology point of view:

1) Since the data will be originating from a very broad set of data sources, the infrastructure will include the big data technologies in order to ensure performances/scalability and reliability of the system.

2) The data infrastructure should support the need that data is analysed in near real time in order to ensure rapid execution and effect change. Therefore, a special attention is given to the need for complex local processing that should happen on the edge (local/embedded devices, gateways) in order to increase the reactivity of the data processing

The both advantages enable the realization of the full potential of the underlying data sources, by ensuring the efficiency of processing and protecting the privacy of collected data.

These requirements will be taken into account in the implementation of the envisioned data infrastructure.