Grant Agreement Number: 731993

Project acronym: AUTOPILOT

Project full title: AUTOmated driving Progressed by Internet Of Things

D-3.7

Test data management platform architecture

Due delivery date: 31/05/2018

Actual delivery date: 31/05/2018

Organisation name of lead participant for this deliverable: AKKA

Project co-funded by the European Commission within Horizon 2020 and managed by the European GNSS Agency (GSA)		
Dissemination level		
PU	Public	X
PP	Restricted to other programme participants (including the GSA)	
RE	Restricted to a group specified by the consortium (including the GSA)	
CO	Confidential , only for members of the consortium (including the GSA)	

Project funded by the European Union's Horizon 2020 Research and Innovation Programme (2014 – 2020)

Document Control Sheet

Deliverable number:	D3.7
Deliverable responsible:	AKKA
Work package:	3
Editor:	Sadeq Zougari

Author(s) – in alphabetical order		
Name	**Organisation**	**E-mail**
AITAAZIZI Amine	AKKA	amine.ait-aazizi@akka.eu
BANOUAR Yassine	CONTI	yassine.banouar@continental-corporation.com
DALET Benoit	AKKA	benoit.dalet@akka.eu
FALCITELLI Mariano	CNIT	mariano.falcitelli@cnit.it
KAHALE Elie	AKKA	elie.kahale@akka.eu
KALOGIROU Kostas	CERTH	kalogir@certh.gr
MATTA Joe	VEDECOM	joe.matta@vedecom.fr
NETTEN Bart	TNO	bart.netten@tno.nl
RIAL Moisés	CTAG	moises.rial@ctag.com
SCHREINER Floriane	VEDECOM	floriane.schreiner@vedecom.fr
SCHOLLIERS Johan	VTT	johan.Scholliers@vtt.fi
VELIZHEV Alexander	IBM RE	ave@zurich.ibm.com
ZOUGARI Sadeq	AKKA	sadeq.zougari@akka.eu

Document Revision History			
Version	**Date**	**Modifications Introduced**	
		Modification Reason	**Modified by**
V0.1	31/11/2017	ToC	ZOUGARI, Sadeq
	30/01/2018	First Draft	ZOUGARI, Sadeq
		Draft with responsibilities	ZOUGARI, Sadeq
V0.9	09/05/2018	Draft ready for peer review	All
V0.10	29/05.2018	All partners contributions completed	All
V1.0	29/05/2018	Version including peer reviewers comments	DALET Benoit, D'ORAZIO Leandro, NETTEN Bart, ZOUGARI Sadeq

Abstract

This document presents the test data platform architecture. The platform includes the Pilot Sites' Test Servers which collect test data at pilot site level, and the Centralised Test Server that stores all the test data for evaluation, and also the evaluation results.

This document explains the implemented architectures in technical detail. For each pilot site, the document identifies the design requirements and presents the chosen solution to be implemented for the development of data uploading, data processing, data storage and data accessing components.

This document also describes what will be implemented to support the provisioning of test data for the evaluation tasks that require log data from vehicles, IoT platforms and cloud services, as well as situational data from the pilot sites to detect situations and events, and to calculate the indicators, and subjective data such as survey results and questionnaires from user and stakeholder activities.

The Centralised Test Server has to implement features that enable the storage and sharing of collected test data in a harmonised way, from any pilot site, and provide interfaces to access and use these data for evaluation purposes.

Legal Disclaimer

Abbreviations and Acronyms

Acronym	Definition
ACP	Access Control Policy
ADA	Automated Data Analysis
ADAS	Advanced Driver-Assistance Systems
AE	Application Entity
API	Application Programming Interface
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
AVP	Automated Valet Parking
CAM	Cooperative Awareness Message
CAN	Controller Area Network
CIN	Content INstance
CON attribute	CONtent attribute
CSE	Common Services Entity
CTS	Centralised Test Server
DDS	Data Distribution Service

DMP	Data Management Plan
EC	European Commission
ETSI	European Telecommunications Standards Institute
FAIR	Findable – Accessible – Interoperable-Reusable
FESTA	Field opErational teSt supporT Action
GA	Grant Agreement
GPS	Global Positioning System
HMI	Human-Machine Interface
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol
ITS	Intelligent Transportation Systems
JSON	JavaScript Object Notation
LIDAR	Light Detection And Ranging
NAS	Network Attached Storage
OBU	On-Board Unit
ORDP	Open Research Data Pilot
PO	Project Officer
PSTS	Project site test server
RADAR	RAdio Detection And Ranging
RAID	Redundant Array of Independent Disks
REST	REpresentational State Transfer
RPM	Revolutions Per Minute
SFTP	Secured File Transfer Protocol
UPER	Unaligned Packed Encoding Rules
URI	Universal Resource Identifier
VPN	Virtual Private Network
WP	Work Package
XER	XML Encoding Rules

Table of Contents

List of Figures

List of Tables

Executive Summary

This deliverable D3.7 *“Test Data Management Platform Architecture”* presents the AUTOPILOT test data management architectures of the pilot sites, in particular the architecture of the Pilot Sites Test Servers, and the architecture of the AUTOPILOT Centralised Test Server, all of which will be implementing the *“Data Collection and Integration methodology”* defined in deliverable D3.6.

Each pilot site clearly identifies the software constraints (non- functional requirements) that will be used to define the final architecture and to finally select the solution to be implemented.

D3.7 also gives an overview of the data flows, the data collection processes inside the pilot sites, and the data uploading and processing into the Pilot Site Test Servers. This document also describes the software architecture of the Centralised Test Server and the services and interfaces provided. D3.7 describes how the evaluator will upload and share its evaluation result.

In addition, this deliverable serves as the starting point for the software implementation of each Pilot Site Test Server and of the Centralised Test Server.

1 Introduction

1.1 Purpose of the document

This document presents the architecture of the **Test Data Management Platform**.

Chapter 2 *"**Test data collection and evaluation system overview**"* is an introductory paragraph summarising the principles of the AUTOPILOT project. It describes the global context, gives an overview of the Pilot Site Test Server principles and of the Centralised Test Server role.

Chapters 3 to 7 *"**Pilot sites specificities**"* describe each pilot site's architecture in detail. The first paragraph describes the main requirements guiding the architecture and software development. The subsequent paragraphs detail the architecture and the components of the PSTS.

Chapter 8 *"**Centralised Test Server Architecture**"* details the Centralised Test Server architecture. As for the pilot sites; the requirements, services and components are described. A paragraph is also dedicated to the interfaces, API and HMI used to interact with the CTS.

1.2 Intended audience

This deliverable (D3.7) is a public document and therefore, the intended audience for this document is considered to be anyone that is interested in data collection and data sharing architecture solutions that could be implemented in large-scale collection and management of test data and evaluation results.

Within the project, the main intended audience for this deliverable is considered to be all AUTOPILOT participants involved in the implementation of the tools, the software and data management system at the pilot site and centralised level and those involved in the technical evaluation (WP2, WP3 and WP4).

1.3 Definitions

Table 1: Definitions

Terms	Definition
Centralised Test Server	A centralised server storing and providing access to test data and evaluation results
Pilot Site Test Servers	A collection of servers and tools in charge of data collection at pilot site level
Test data	Data collected during piloting activities according to requirements defined by WP4
Test data management platform	A collection of tools and servers including centralised and distributed test servers for collection of test data

2 Test data collection and evaluation system overview

This section presents an overview of the test data management system as an introduction to the description of the architecture of its components.

2.1 Context overview

The test data management platform is used to manage and collect the data produced by operational tests so that it is available to evaluators.

The platform is composed of several distributed Pilot Sites Test Servers and a Centralised Test Server.

The **Pilot Sites Test Servers** (PSTS) are the base elements in responsible for collecting and processing test data from vehicles and IoT.

The role of the **Centralised Test Server** (CTS) is to manage and store test data uploaded from the Pilot Sites' Test Servers, and allow evaluators to access and use this data. The evaluator can also upload and share its evaluation results with AUTOPILOT partners.

The architecture of the PSTS and of the CTS are guided by the requirements (about data, tools), the specifications (about tests, evaluation) and the guidelines (about data management) defined in the document "D3.6 Data collection and integration".

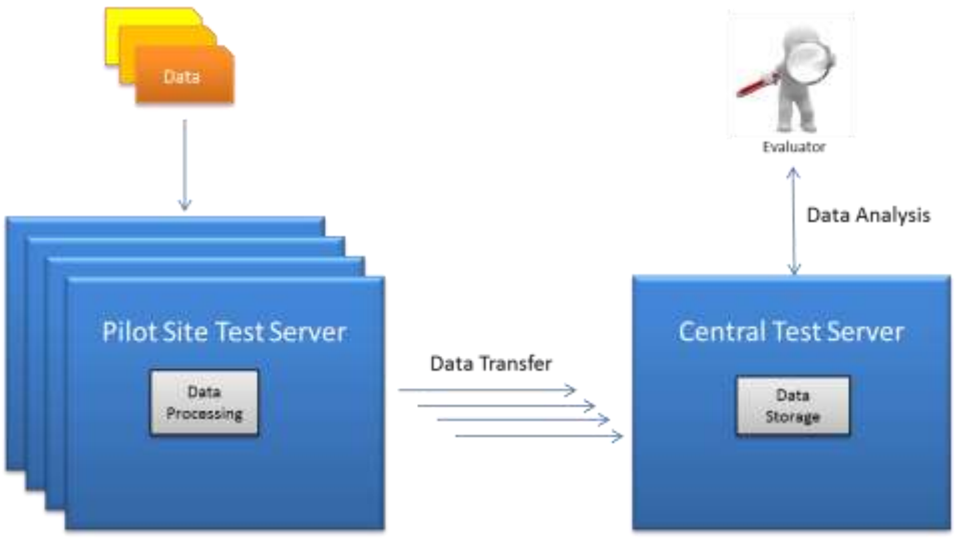


Figure 1: AUTOPILOT global system context view

Pilot Sites Test Servers interact with the pilot site test environment:

- Each pilot site is in charge of collecting data from vehicles, sensors, IoT, data platforms, environment, etc., while running test scenarios
- Each pilot site parses, filters, cleans and prepares collected data
- Each pilot site transfers log data in common format for evaluation to the Central Test Server

Central Test Server interacts with pilot sites and evaluators

- Central server collects, sorts and stores received test data and metadata

Evaluators interact with the Central Server

- Metadata are used to browse and retrieve collected and stored test data
- Evaluators can access and search data in order to analyse and evaluate the project
- Evaluators can store their evaluation results in a dedicated database

2.2 Pilot Site Test Server overview

A PSTS can be described as a workflow composed of the following components:
a data collection system that hands data to the processing components (parsing, filtering, quality check, enrichment) and a storage component collecting processed data which are made ready for uploading to the CTS.

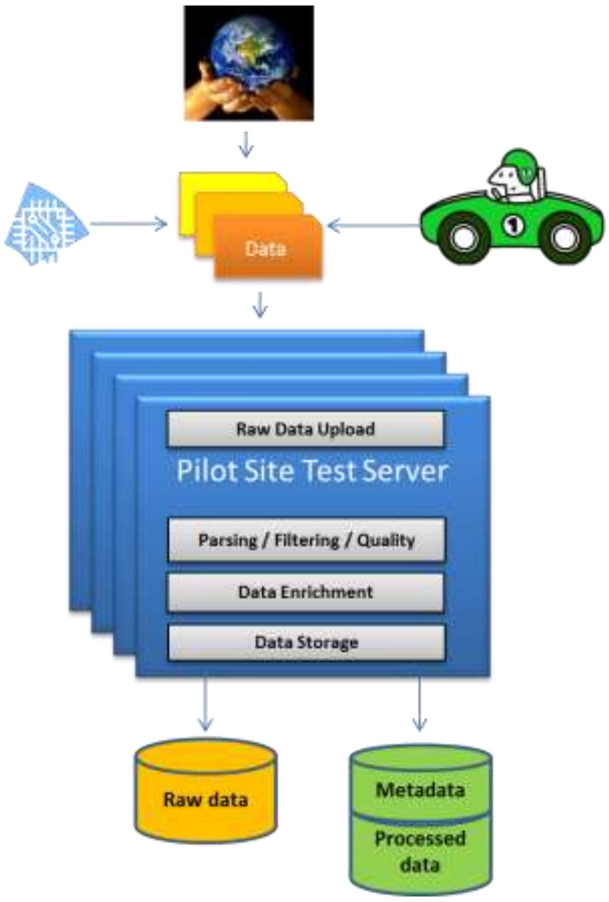


Figure 2: Pilot Site Test Server generic system view

A specific Pilot Site Test Server must collect and store all the data created by a pilot site: contextual data, acquired data, derived data, aggregated data (optional), and metadata.

General requirements applicable to all pilot sites:

- Each pilot site must ensure the data collection according to the data quality requirements.
- Each pilot site must provide the requested data in compliance with evaluation requirements

(time tag, formats, measurements, etc.).
- Data must be provided with its metadata (complete description of test and test data).
- Each pilot site must ensure that IoT data remains on the device until it is stored in the platform.
- Concerning data format requirements, data to be sent to the central IoT platform must follow a predefined plan.
- Each pilot site must provide a secure environment for data storage.
- Each pilot site needs to have the necessary tools to check data quality. Automated scripts may be provided to process large datasets in order to ease and enable post-processing of aggregated data.

2.3 Centralised Test Server overview

The Centralised Test Server collects data produced by pilot sites and makes it available to evaluators. Evaluators will be able to download data from the centralised server, and access specific types of data directly inside the Server (API REST). The Server will also provide a way to store the results of the evaluation, and enriched data produced by the evaluators.

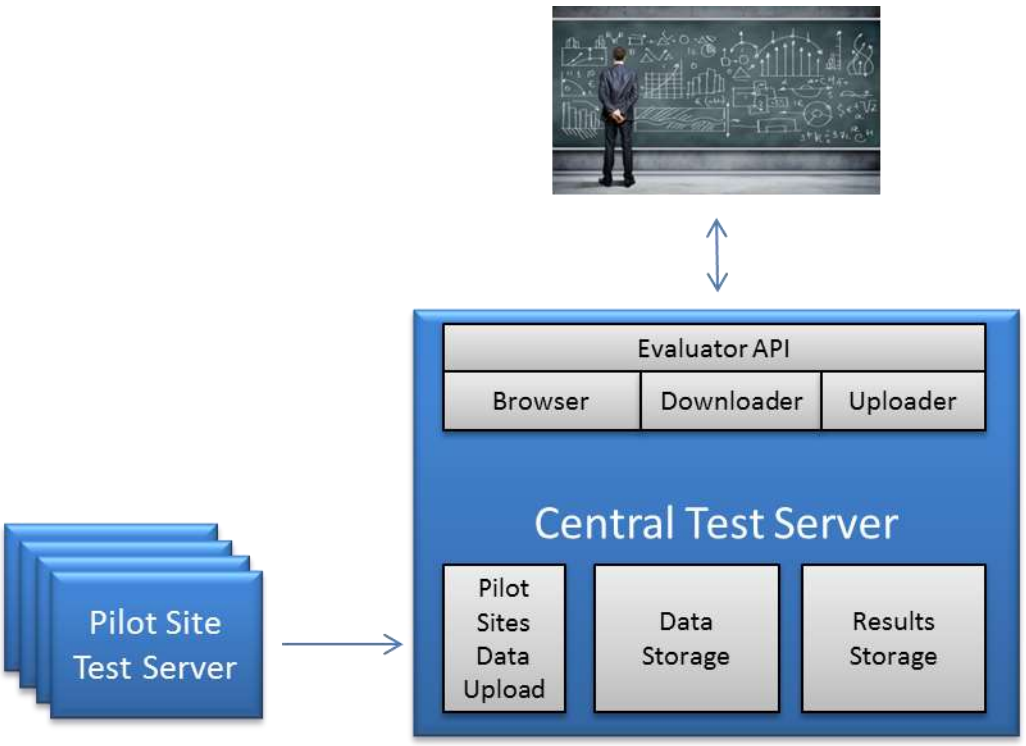


Figure 3: Centralised Test Server generic system view

General requirements of the Centralised Test Server:

- Data must be described using additional information called metadata. The latter must provide information about the data source, the data transformation and the conditions in which the data has been produced.
- As the project will collect several data categories and several data types, several metadata descriptions must be provided to describe the characteristics of each measure or component and also how the data was produced and collected.

- The Centralised Test Server will collect all of the Pilot Sites' test data and metadata required for the evaluation.
- Before uploading to the CTS, data must be anonymised.

3 Versailles pilot site specificities

This section presents the architecture of the Versailles pilot site.

3.1 Architecture design rationale

This subsection presents the rationale behind the global design, i.e. it describes and justifies the main design decisions. The requirements and constraints define what the project expects from the architecture.

3.1.1 Non-functional requirements

The system requirements prescribe the architecture at the system level; the components requirements refine the system requirements for a specific subsystem.

The requirements listed below are the most important non-functional requirements that the Pilot Sites will comply with (or adhere to) in order to ensure optimal integration in the global AUTOPILOT project.

3.1.1.1 Design constraints

From an architecture standpoint, a constraint is an architectural design or implementation decision that has been selected to be treated as if it were a formal requirement.

The following table displays Versailles' design constraints.

Table 2: Versailles pilot site design constraints

ID	Title	Description	Source
NFRQ-DC-01	IoT	PS should be able to connect to get data from an IoT platform.	GA
NFRQ-DC-02	Data collection	PS must be able to collect data stored or published in the Sensinov IoT platform.	D3.6
NFRQ-DC-03	Data collection	PS must enable manual collection of test data using a hard drive, flash drive, USB key	D3.6
NFRQ-DC-04	Data Storage	PS Test Server should use a database to store test data descriptions.	D3.6
NFRQ-DC-05	Data Storage	PS Test Server should use a FOT database to store test data.	D3.6
NFRQ-DC-06	Storage	PS Test Server must have a storage file system.	GA – DMP, D3.6
NFRQ-DC-07	Data Transfer	PS Test Server must allow stored test data to be sent by ftp to the Centralised Test Server.	D3.6
NFRQ-DC-08	Test Data Interface	PS must send data that are compliant to the provided description	D3.6 decision reused at PS level
NFRQ-DC-09	Test Server Interface	PS Test Server should provide an API for Versailles Test Data uploading	D3.6
NFRQ-DC-10	Test Data Monitoring	Pilot site should display the status of the test data upload.	D3.6 reused at PS level
NFRQ-DC-11	Data Format & Data Model	Vehicle data must be stored in a human readable format and must follow the	D4.1 & D3.6

		WP4.1 parameter naming rule, parameter type and parameter quality constraints	
NFRQ-DC-12	Data Format & Data Model	GPS data must be stored in a human readable format or standard format and must follow the WP4.1 parameter naming rule, parameter type and parameter quality constraints	D4.1 & D3.6
NFRQ-DC-13	Data Format & Data Model	IoT data must be stored according to an agreed data model and data format	D4.1 & D3.6
NFRQ-DC-14	Data model	CAM, DENM and SPAT must be stored in agreed data format and data model: ITS G5	D4.1 & D3.6
NFRQ-DC-15	Networking	PSTS must be connected to the Internet	D3.6

3.1.1.2 Architecture quality goals

This section presents two kinds of quality requirements:

- Performance and Scalability requirements
- Availability and Reliability requirements

Table 3: Versailles Pilot site non-functional quality requirements

ID	Title	Description	Source
NFRQ-QR-01	Data Interface	PS Test Server must send the Test Data according to interface defined by CTS (tar file and description file)	D3.6
NFRQ-QR-02	Data Storage / Backup	PSTS must provide a secured storage and backup to avoid data loss.	D6.9, D3.6
NFRQ-QR-03	Data Storage	PSTS must provide enough storage space.	D3.6
NFRQ-QR-04	Data interface	Pilot site must provide all mandatory parameters for Urban driving, Car sharing and Platooning	D4.1 and D3.6
NFRQ-QR-05	Data Storage	PS Test Server must store IoT, vehicle and survey data as they are provided	D4.1
NFRQ-QR-06	Data identification	All provided data must be easily identified by station ID and application ID	D4.1 & D3.6
NFRQ-QR-07	Data Identification	All the provided data must be clearly identified with metadata and timestamped	D6.9, D4.1 & D3.6
NFRQ-QR-08	Data synchronisation	All PS Versailles data sources must be synchronised	D4.1 & D3.6
NFRQ-QR-09	IoT Data Model & Data Format	PS Test must provide IoT data in the right data model and the right data format	D4.1 & D3.6
NFRQ-QR-10	Vehicle Data Model & Data Format	PS Test must provide Vehicle data in the right data model and the right data format	D4.1 & D3.6
NFRQ-QR-11	Surveys Data Model & Data Format	PS Test must provide Survey data in the right data model and the right data format	D4.1 & D3.6

3.1.1.3 IoT platform requirements

This section presents the requirements related to the IoT platform.

[Note that these requirements are mostly concerning what is expected from the IoT platform.]

Table 4: Versailles Pilot site IoT platform requirements

ID	Title	Description	Source
NFRQ-IOT-01	Platform Availability	The IoT platform must be available when running test sessions. (M19)	D2.3 & D3.2
NFRQ-IOT-02	Platform Availability	The IoT platform must remain available after test sessions for data collecting.(M36)	D3.2
NFRQ-IOT-03	Platform feature	The IoT platform must provide pub/sub and discovery mechanism	D3.2 & D3.6
NFRQ-IOT-04	IoT Platform standard	The IoT platform must provide standard oneM2M implementation	D2.3
NFRQ-IOT-05	Platform Storage Capabilities	The IoT platform must provide storage means until data are collected by the pilot site.	D3.6
NFRQ-IOT-06	Data logging	The IoT platform must provide logging of events.	D4.1
NFRQ-IOT-07	Data synchronisation	The IoT platform must be synchronised with any device or vehicle of the Versailles PS.	D4.1
NFRQ-IOT-08	Data synchronisation	The IoT platform must add a time stamp upon data/measure reception.	D4.1

3.1.1.4 Other non-functional requirements

This section includes:

- Security requirements

Table 5: Versailles pilot site non-functional requirements

ID	Title	Description	Source
NFRQ-NF-01	Data privacy	All collected data including (surveys) must be anonymised before storage.	D6.9

3.1.2 Design rationale

The design decisions are guided by the following use cases of the PSTS:

- Uploading data from Versailles pilot sites
- Browsing, searching data
- Storage of various type of data
- Upload tasks monitoring
- Users and profiles management
- Downloading data

- Selecting and sending data to the CTS

3.2 Global architecture

This section presents the essentials of Versailles pilot site system architecture, including main API components, services and functionalities.

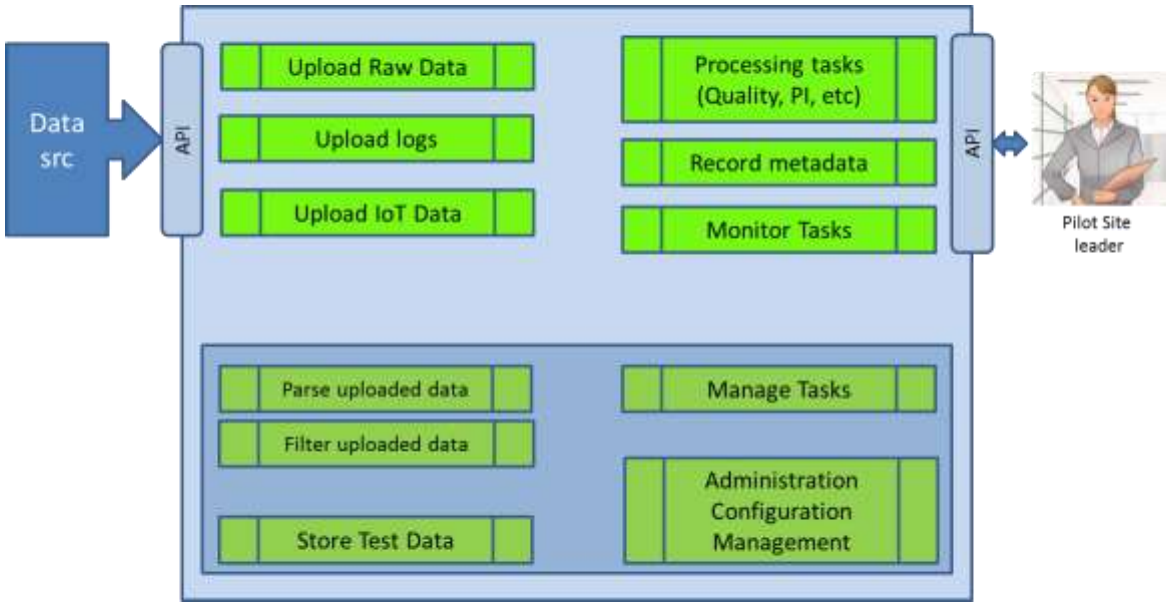


Figure 4: Versailles Pilot Site Test Server architecture

3.3 Components architecture

This section presents the detailed components architecture.

3.3.1 Data upload components

This section describes the main data collection components that will be deployed to acquire PS test data. It is not about the data collection by the devices, but how devices, platforms and other sources' data are uploaded to the Test Server Platform.

The following list summarises the collected and uploaded data, and describes the types of data format and the container file type:

- Vehicle data (CAN): JSON file(s) containing the parameter list agreed with WP4
- CAM data: ASCII file containing ETSI CAM encoded as XER encoding rules, and as UPER encoding rules
- IP and ARP packets: Pcap file containing all CAM and other ETSI messages on 802.11-OCB and all IP and ARP messages
- IoT data: JSON files compliant to oneM2M data containing the data related to each use case

3.3.1.1 IoT platforms

Following the oneM2M standard, platforms implementing it act as a CSE (Common Services Entity) and expose a set of functions (data storage, discovery, etc.). Data generated by devices and stored in the platform can then be consumed by applications depending on the provided service. In order to make data available to other applications, each application can store the generated outputs in the oneM2M platform. Therefore, for the evaluation, PSTS can be considered as an Application Entity by the oneM2M platform that has the rights to access to all stored data.

Interactions with the platform are based on the REST architectural style in order to facilitate the development of services and increase interoperability with devices. This architectural style considers each physical or logical entity as a resource having a remotely accessible representation. Each resource is uniquely addressable via a Uniform Resource Identifier (URI). For the interaction with the oneM2M platform, HTTP protocol can be used to manage resources.

For data collection, oneM2M standard defines different procedures:

- **Request/Response:** the requester sends a request to the platform in order to retrieve a specific resource. Using HTTP protocol, the request is a GET request with the URI of the requested resource, and the returned response contains the representation of the requested resource;
- **Subscription/Notification:** the requester subscribes to the resources collection in order to track changes (e.g. creation of a resource) in the oneM2M platform. The requester creates a specific resource called subscription in the tracked collection. This resource contains the address of the subscriber. In the case of a resource creation, the oneM2m platform sends a notification to the subscriber containing the representation of the created resource;
- **Discovery:** the requester sends a discovery request that allows discovering resources residing on the oneM2M platform. The discovery procedure is based on a retrieve (GET through HTTP protocol) request where the URI includes the root resource from where the discovery begins. The result of the request is a list of all child resources under the root resource. When the *filterCriteria* parameter is specified in the URI, the result is reduced to all specific child resources under the root matching the *filterCriteria* condition.

For the evaluation procedure, the subscription/notification procedure can be used when the PSTS requires data in real-time. In this case, in order to get data stored from all devices and services in oneM2M platform, the PSTS shall create the subscription resource and specifies the endpoint where to receive notifications in all collections. Otherwise, if there is no requirement regarding receiving resources in real-time, the PSTS can use the discovery procedure to retrieve a set of data matching specific conditions. Before data ingestion and processing, collected resources are stored on a local server.

For structuring resources, oneM2M standard defines a resource tree. For Versailles PS, the global resource tree regarding the car-sharing service is depicted by Figure 5. The structure is based on the separation of resources by data and service types (parking slots, charging spots, vehicles, car sharing, etc.).

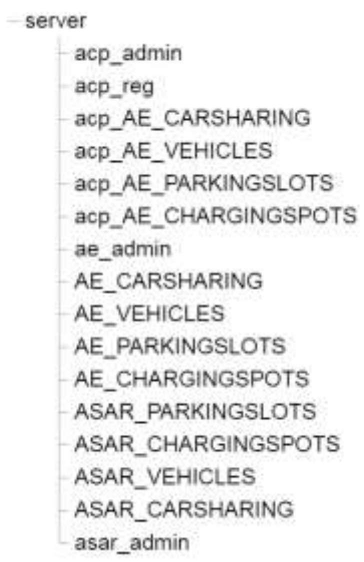


Figure 5: Global overview of oneM2M resource tree for Versailles PS

Considering the global structure of the resources tree for the Versailles PS, the parent resource for parking slots data (Figure 6) is AE_PARKINGSLOTS. This resource contains a first level of containers related to each parking (e.g. CNT_PARKING-1). Then, a second level related to Parking slots (e.g. CNT_SLOT-101). The third level of containers is DATA containing slots data (CIN resources).

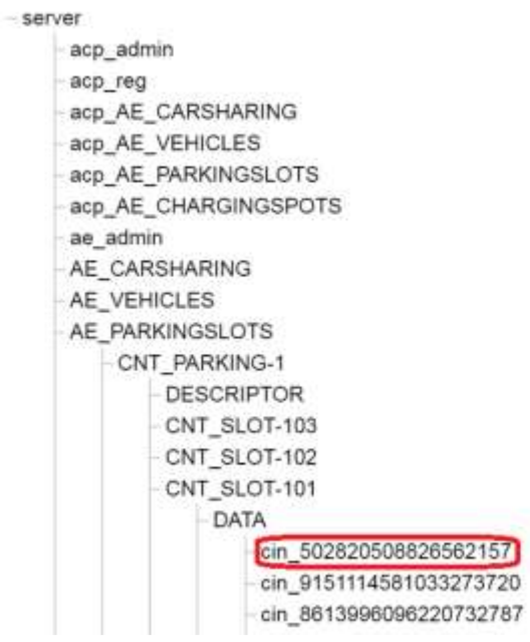


Figure 6: Structure of Parking Slots Data for Versailles PS

For instance, in the case of an occupied slot, data can be as follows:

```
{
 "SlotStatus": "OCCUPIED",
 "LicensePlate": "AP-001-VE"
}
```

Under the DATA container, each new data is stored as a CIN (Content Instance) resource in the oneM2M platform. For the previous example, the CIN resource can be as depicted bellow where

CON attribute contains the raw data:

```
"m2m:cin": {
  "rn": "cin_5634606628112954",
  "ty": 4,
  "ri": "/server/cin-5634606628112954",
  "pi": "/server/cnt-7621087546889705752",
  "ct": "20180328T145038",
  "lt": "20180328T145038",
  "et": "20180328T145038",
  "st": 0,
  "cnf": "application/json",
  "cs": 39,
  "con": "{\"SlotStatus\": \"OCCUPIED\",\"LicensePlate\": \"AP-001-VE\"}"
}
```

When the Subscription/Notification procedure is used, the subscriber receives a notification for each new data. For instance, in the case of new parking slot data (e.g. parking 1, slot 101), the notification is as follows:

```
{
 "m2m:sgn": {
  "nev": {
   "rep": {
    "m2m:cin": {
     "rn": "cin_5634606628112954",
     "ty": 4,
     "ri": "/server/cin-5634606628112954",
     "pi": "/server/cnt-7621087546889705752",
     "ct": "20180328T145038",
     "lt": "20180328T145038",
     "et": "20180328T145038",
     "st": 0,
     "cnf": "application/json",
     "cs": 39,
     "con": "{\"SlotStatus\": \"OCCUPIED\",\"LicensePlate\": \"AP-001-VE\"}"
    }
   },
   "rss": 1
  },
  "sud": false,
  "sur": "/server/server/AE_PARKINGSLOTS/CNT_PARKING-1/CNT_SLOT-101/DATA/SUBSCRIBER "
 }
}
```

The data hierarchies, relevant for applications consuming the data from the OneM2M platform, are described below:

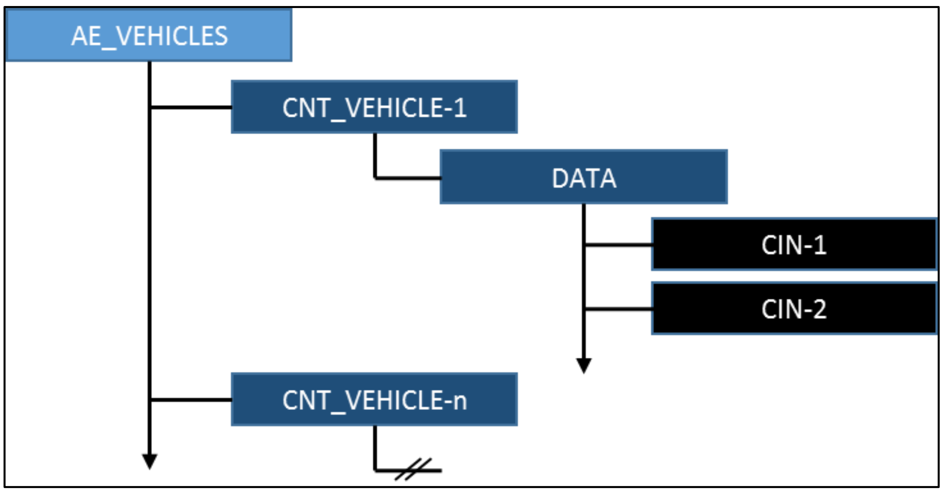


Figure 7: Versailles AE_VEHICLES hierarchy

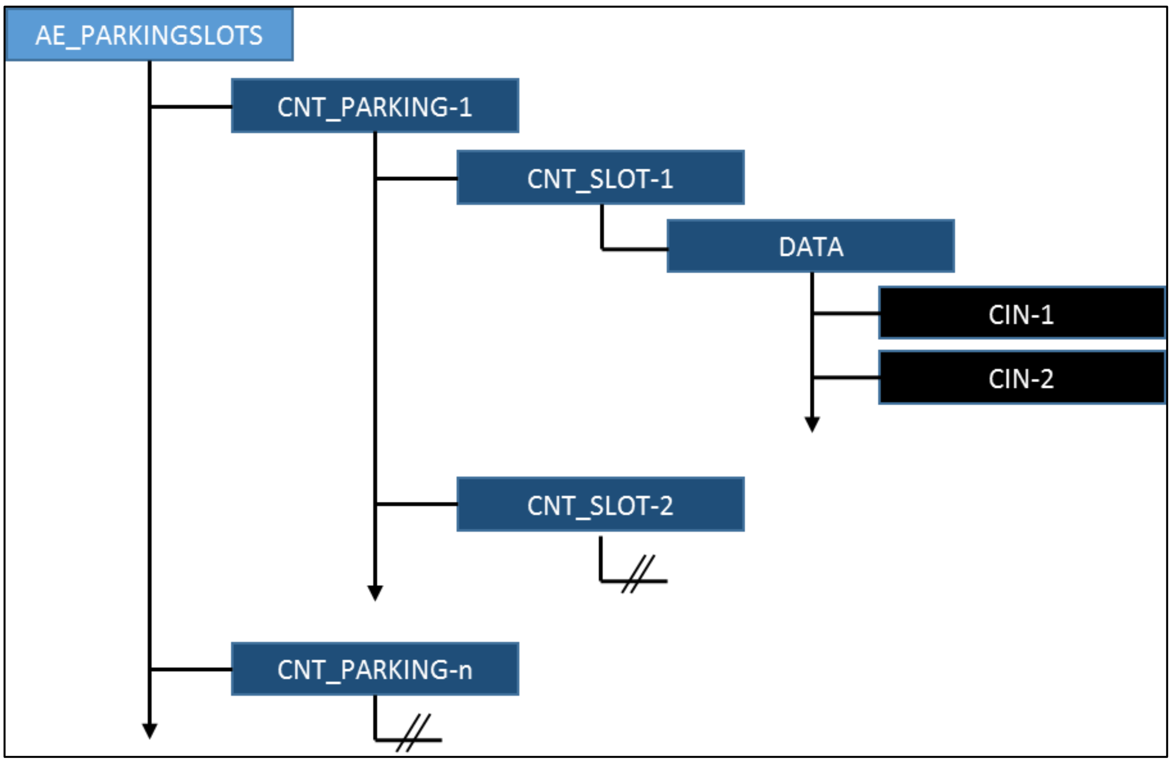


Figure 8: Versailles AE_PARKINGSLOTS hierarchy

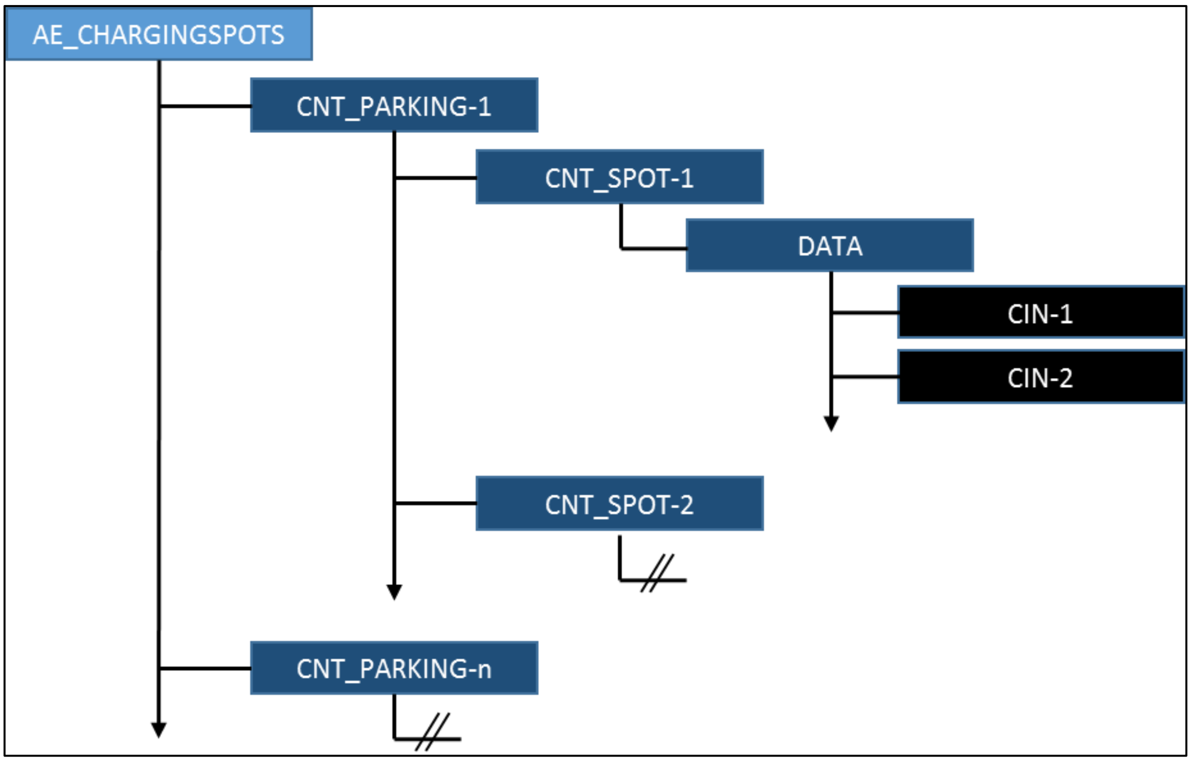


Figure 9: Versailles AE_CHARGINGSPOTS hierarchy

Urban driving

The parent resource for the urban driving slot is AE_URBAN_DRIVING. The URBAN_DRIVING Application Entity consists of AE_URBAN_PEDESTRIAN and AE_URBAN_BICYCLE containers as it is shown below.

ID	Name	State Tag	Max Nbr Of Instance	Max Byte Size	Max Instance Age	Current Nbr Of Instance	Current Byte Size	Actions
[illegible]	[illegible]	0	[illegible]	0	0	0	0	
[illegible]	[illegible]	0	[illegible]	0	0	0	0	

Figure 10: Versailles URBAN_DRIVING Application Entity

The AE_URBAN_PEDESTRIAN will contain the subscriptions of pedestrians and the AE_URBAN_BICYCLE container will contain the subscriptions of bicycles respectively. An example of a subscription is shown below.

ID	Name	Creation Time	Actions
[illegible]	[illegible]	[illegible]	

Figure 11: Versailles AE_URBAN_DRIVING containers

After a successful subscription, the data will be sent in `JSON` format according to oneM2M API standard rules. The communication will take place between a pedestrian's smart-phone and the oneM2M server for the pedestrian urban driving use case and between the bicycle ECU and the oneM2M server for the bicycle case. The hierarchy of the urban driving application entity is depicted in Figure 12.

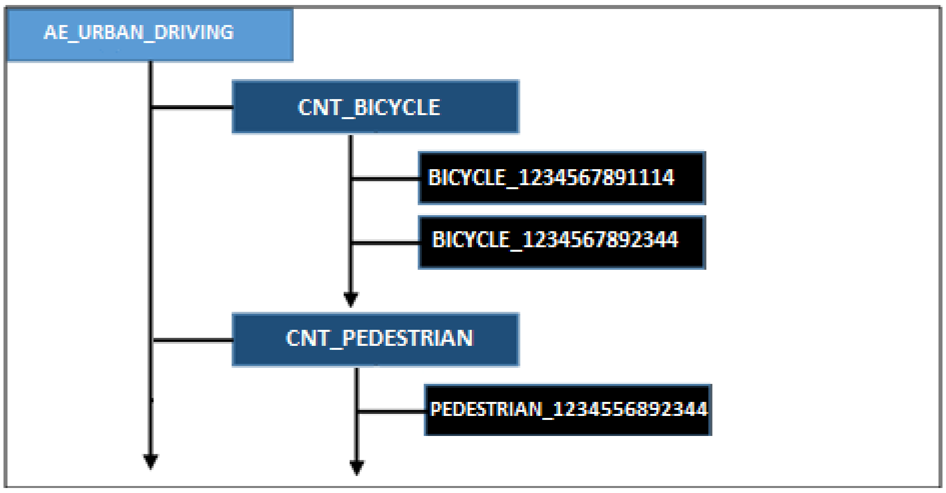


Figure 12: Versailles AE_URBAN_DRIVING hierarchy

3.3.1.2 Vehicle data

All the necessary vehicle data will be published at runtime and available in the OneM2M platform (vehicles application entity).

3.3.1.3 Surveys data

The tool to be used on the French pilot site has to be decided by WP4. SoSci could be used to create the Versailles' user acceptance tests. The results will be collected and uploaded to the SoSci Cloud Platform as a consolidated .csv file. This file can be downloaded later from the SoSci Cloud Platform through a sign in process. Then, this .csv file can be uploaded to Versailles Pilot Site Test Server manually and be available for further data analysis. All data collection and storing will comply with the General Data Protection Regulation (GDPR), EU016/679[1].

3.3.1.4 V2X messages data

In the VFLEX vehicle prototype there are three distinct means of collecting and uploading data to the cloud (PSTS). Firstly, the CAM messages generated by the PC-AD entity in VFLEX are sent to the cloud. These CAM messages are formed by the PC-AD by using, among others, data that circulates on the CAN network(s) in the car. Some of the CAN parameters may be logged as part of the CAM message logging in the OneM2M platform.
The CAM messages sent to the cloud are formatted according to CAM specification, and using XER encoding rules. Secondly, the IP-OBU entity in the VFLEX collects the data packets circulated on Ethernet and on 802.11-OCB links (CAM messages, and all IP and ARP messages); this data is stored in a local file formatted according to the *pcap* format; the file is captured using the commands *tshark*

[1] https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32016R0679

and *tcpdump*; this file may be sent to the cloud as well, depending on the requirements. Thirdly, a special entity in the vehicle collects more data in the car.

3.3.2 Data processing components

The purpose of this paragraph is to detail the raw data processing: data preparation, quality check, filtering, analysis, enrichment, metadata generation and storage.

Note that at PSTS level, only mandatory quality check and filtering are performed. More advanced filtering, processing, calculation and enrichment treatments are performed by the technical evaluation task on data gathered from the Centralised Server.

The Figure 13 outlines the process that will be followed by the Versailles PS:

The data collection process will collect and group all test data related to a specific test scenario or test session. Then data are described and annotated. Finally, according to the test data interface defined in the data collection methodology, a Test Data archive is created and stored.

Depending on the type of data collected, several processing tasks will be performed in order to provide clean and ready to use data to the evaluator.

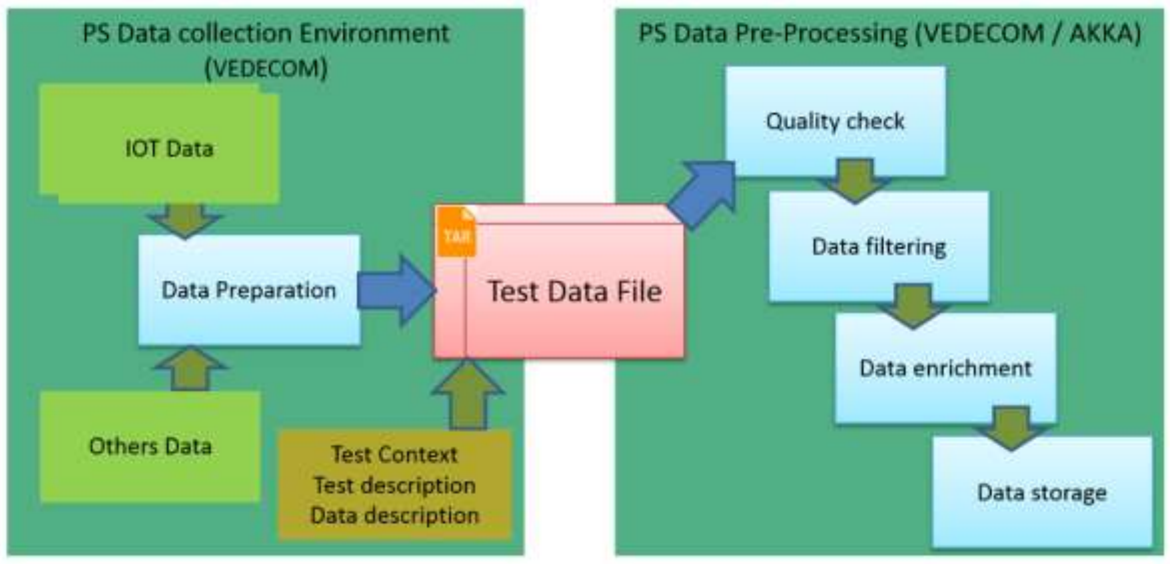


Figure 13: Test data collection and pre-processing on PS Versailles

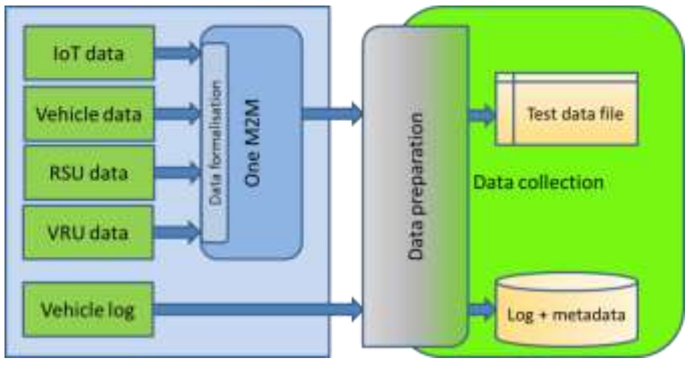


Figure 14: Test data sources and flows in PS Versailles

3.3.2.1 Analysis
The data stored in the pilot site will be compliant with the OneM2M model for the IoT data and the ETSI ITS-G5 standard for V2X data.

3.3.2.2 Quality check
The aim of data quality is to check that the data can be useful and ready for processing. This is done by checking the following items:

- Assessing and quantifying missing data
- Controlling data values and units of measure
- Checking that all the data are synchronised
- Checking that the data are timestamped
- Checking that the data are compliant to the predefined data model and data format
- Checking that the data are clearly identified by station id and application id

3.3.2.3 Processing and enrichment
At the level of PS site, processing and enrichment is limited to providing mandatory data. There is no calculation of one of the following: Events detection, Situation, indicator calculation.

3.3.2.4 Metadata definition
At the level of PS Versailles, Metadata definition will be limited to providing a complete description of the Test Data API described in D3.6.

3.3.3 Data storage

3.3.3.1 Raw data storage
There is no raw data storage. Refer to section 3.3.1.1 IoT platforms.

3.3.3.2 Processed data storage
Processed data will be stored and indexed in a PS test server.

The Figure 15 represents the data storage means on the French pilot site:

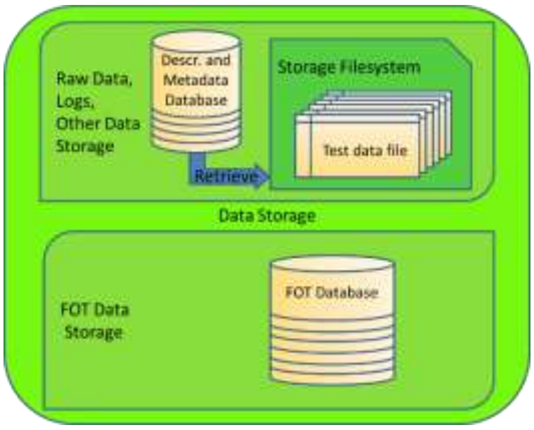


Figure 15: Data storage means on the French PS

The PS test server will contain two distinct types of storage:
- A database and a file system for raw data, logs and derived data

- A database (FOT optional) for test data

The “Descr. and Metadata Database” shown in Figure 15 contains all the information extracted from the “description.xml” (test data metadata) described in D3.6.

4 Brainport pilot site specificities

This section presents the architecture of the Brainport Pilot Site Test Server (PSTS). The PSTS is the Automated Data Analysis (ADA) server environment from on https://ada1.tno.nl/autopilot. All data from the pilots of Use Cases for Platooning, Car Rebalancing, Highway Pilot and Automated Valet Parking will be uploaded to the ADA repository. The pre-processed data will also be accessible from ADA and made available to the Central Test Server described in section 8.

The architecture, rational and requirements are based on the common approach to data logging, analysis and evaluation for the use cases of the PSTS, as specified in the "AUTOPILOT_CommonLogFormatDescription_extension", which is an extension of the "InterCor_CommonLogFormatDescription". Documents and related specification documents are provided on ProjectPlace [2].

4.1 Architecture design rationale

This subsection presents the rationale of the global design, i.e. it describes and justifies the main design decisions. The requirements and constraints define what the project expects from the architecture.

4.1.1 Non-functional requirements

Requirements below are the most important non-functional requirements for the Brainport PSTS in order to ensure an optimal integration into the global AUTOPILOT project.

4.1.1.1 Design constraints

Table 6: Brainport pilot site design constraints

ID	Title	Description	Source
NFRQ-DC-01	Data uploading	PSTS should provide a facility to manually and automatically upload log data in common formats	D3.6
NFRQ-DC-02	Common Log Formats	PSTS should be able to manage log data in predefined common formats, as defined in section 3	D4.1
NFRQ-DC-03	Structured data	PSTS only allows uploading of structured log data, as defined in section 3.	D3.6
NFRQ-DC-04	Data Storage	PSTS should use a database to store test data descriptions	D3.6
NFRQ-DC-05	Data access	PSTS should provide access to the pre-processed data	D3.6
NFRQ-DC-06	Data access	PSTS should provide access to the pre-processed data for uploading to the Central Test Server	D3.6
NFRQ-DC-07	Data quality	All applications providing logging, and all log data provided, shall be time synchronised	D4.1

[2] "AUTOPILOT_CommonLogFormatDescription_extension" and related descriptions and specifications, ProjectPlace: [Pilot sites | Brainport | Evaluation], https://service.projectplace.com/pp/pp.cgi/r1663184828

NFRQ-DC-08	Data quality	All data shall be logged with timestamps in UTC in milliseconds or higher precision.	3
NFRQ-DC-09	Data quality	Only log data of the same message type can be provided in a single log file	3
NFRQ-DC-10	Data quality	All stations, IoT devices, IoT services, IoT platforms and applications have a project-unique identifier	3

4.1.1.2 Architecture quality goals

This section presents two kinds of quality requirements:

- Performance and Scalability requirements
- Availability and Reliability requirements

Table 7: Brainport pilot site non-functional quality requirements

ID	Title	Description	Source
NFRQ-QR-01	Data Storage	PSTS must provide enough storage space.	D3.6
NFRQ-QR-02	Data Storage / Backup	PSTS must provide a secured storage and backup to avoid data loss (RAID1).	D6.9, D3.6
NFRQ-QR-03	Data identification	All provided data must be easily identified as specified in the common formats, as defined in [3]	D4.1 & D3.6

4.1.1.3 IoT platforms requirements

This section presents the requirements related to the IoT platforms.

Table 8: Brainport pilot site IoT platform requirements

ID	Title	Description	Source
NFRQ-IOT-01	Platform feature	The IoT platform must provide pub/sub and discovery mechanism	D3.2 & D3.6
NFRQ-IOT-02	Platform Storage Capabilities	The IoT platform must provide storage means until data are collected by the Pilot Site.	D3.6
NFRQ-IOT-03	IoT Platform standard	The IoT platform(s) must provide data logging of the standardised common IoT messages as defined in Task 2.3	D2.3
NFRQ-IOT-04	Data Logging	The IoT platform(s) must provide unique identifiers with every IoT message as defined by the IoT device that generated the IoT message	D4.1
NFRQ-IOT-05	Data management	The IoT platform must implement the "Store and Share" paradigm with data historisation	

4.1.1.4 Other non-functional requirements

This section includes:

- Security requirements

- Legal requirements

Table 9: Brainport pilot site non-functional requirements

ID	Title	Description	Source
NFRQ-NF-01	Security	The PSTS must implement some levels of assurance for authentication	
NFRQ-NF-02	Legal	The PSTS should meet GDPR requirements asset in AUTOPILOT	D6.7
NFRQ-NF-01	Data privacy	All collected data including (surveys) must be anonymised before storage.	D6.7

4.1.2 Design rationale

The design decisions are guided by the rational for the approach in section 3:

- Logging is primarily collected and stored for validation and evaluation.
- Only structured logging is accepted and stored.
- Raw sensor data is not stored, only detections or interpreted data.
- Logging, storage and pre-processing is supported for common logging as defined in section 3 and D4.1.
- Logging is provided in files containing log items, such as IoT messages, of the same message type.
- Logging distinguishes logging of communication, vehicle data, application and control logic.
- For communication logging either the complete messages are logged as specified by the communication standard, or the message identification information is logged.
- Logging is uploaded and pre-processed per use case and per session for validation or pilot test run.
- The Automated Data Analysis (ADA) toolset is applied for processing the log data. ADA contains predefined scripts, as described in the following subsections.
- The results of ADA processing are provided together with the processed log data in a data base per experiment. This database can be uploaded by the CTS and partners such as the evaluators, for further data analyses and evaluations

4.2 Global architecture

The Brainport pilot site implements a distributed approach to data management. The use cases will use different IoT platforms, IoT devices, cloud services and automated vehicles. Figure 16 shows some of the types of platforms, devices and services as an example. Data will be collected and stored in multiple data management systems before uploading the data to the PSTS.

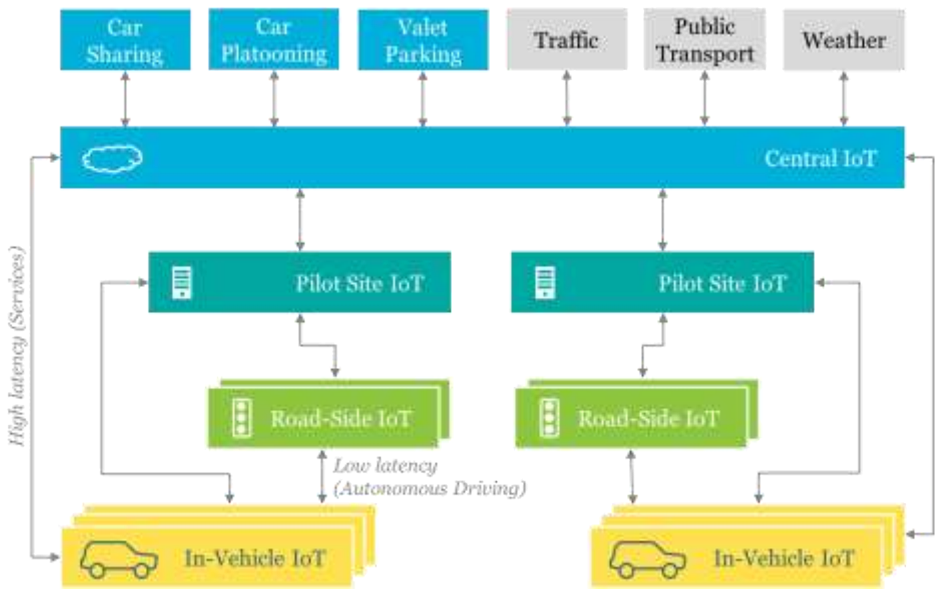


Figure 16: IoT platforms and services in the Brainport pilot site

Different IoT platforms are used in various combinations for the different use cases. The Sensinov oneM2M platform as the central IoT platform, and IoT Watson, FiWare and Ocean are used as federated IoT platforms. Cloud services and platforms are connected for example for Traffic Management and Control, Platooning service, Highway Pilot anomaly detections, HD Map and ADAS, crowd monitoring, parking spot detection, car and ride sharing, and route planning. Different automated and cooperative vehicle platforms are used, each with specific implementations of in-vehicle IoT platforms.

Figure 17 shows a high level architecture for collecting, uploading, pre-processing and analysing log data from vehicles, vehicle IoT platforms, (federated) IoT platforms, IoT devices and cloud services. Log data relevant to evaluations are converted into common log formats and uploaded to the PSTS. The Brainport PSTS consists of a PostgreSQL data base management system, that is structured in data bases per use case and pilot or test session.

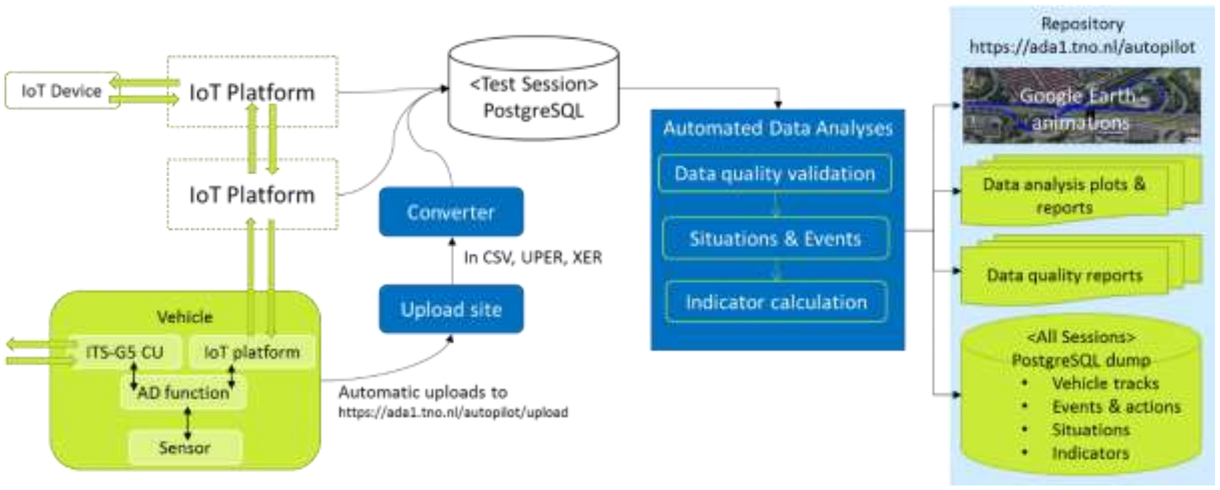


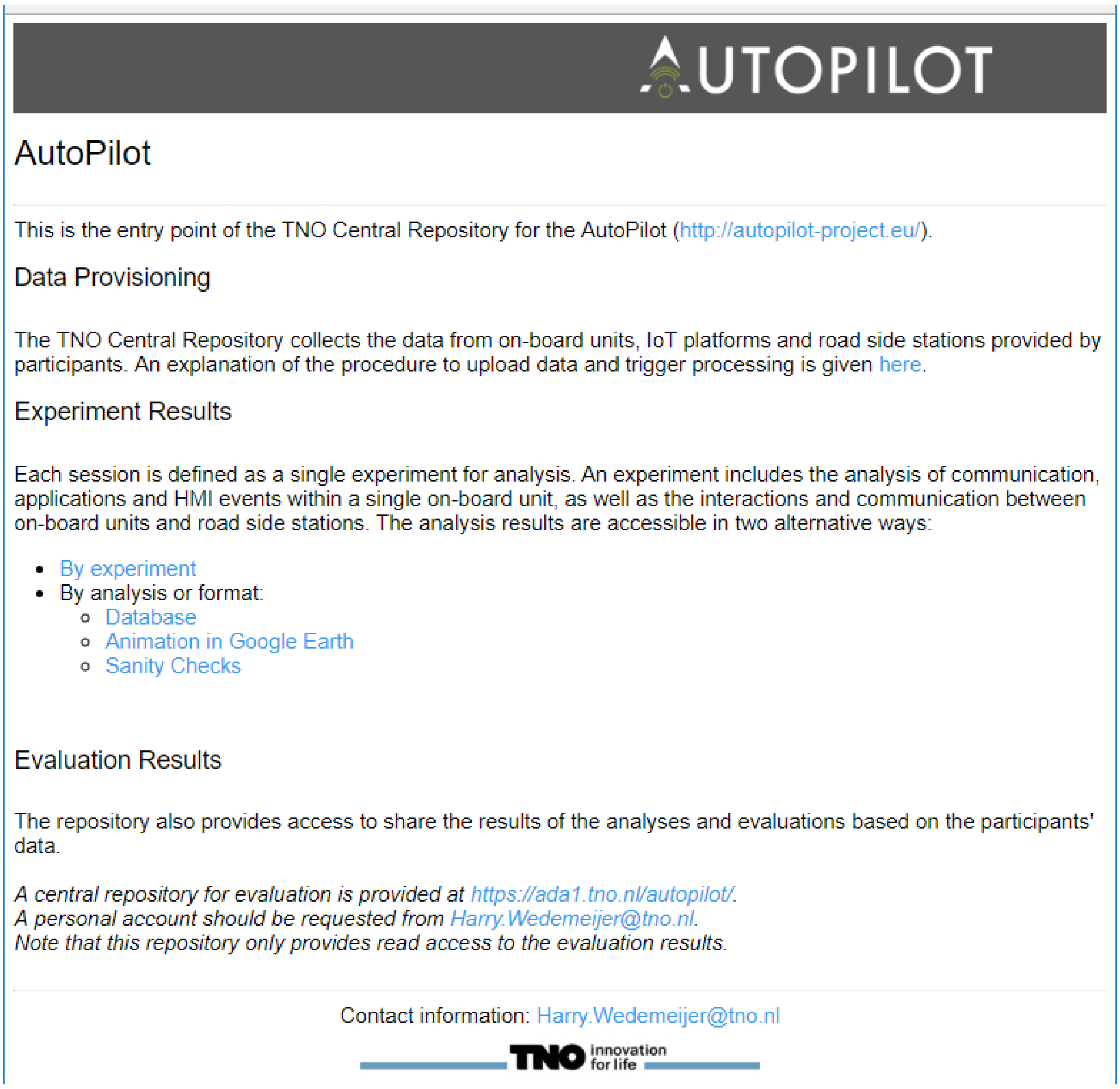
Figure 17: High-level architecture of the Brainport Pilot Site Test Server

Once all data from a single test session is uploaded, including the data from all vehicles, IoT devices, IoT platforms and cloud services involved in this test session, pre-processing is executed. The pre-processing consists of one or more data analysis steps such as data quality validation or sanity checks, detection of situations and events, and the calculation of indicators. The scope and output of the pre-processing may differ per use case, and subject alignment with (technical) evaluations in work package 4. All outputs from the PSTS pre-processing will be provided to the CTS in the form of PostgreSQL database dumps, including pre-processing log data, data analyses results, data quality reports, vehicle tracks, application events and actions, situations and generic indicators.

4.3 Components architecture

4.3.1 Data repository

The PSTS is accessible to partners via a web interface on https://ada1.tno.nl/autopilot/. Figure 18 shows the home page.



AUTOPILOT

AutoPilot

This is the entry point of the TNO Central Repository for the AutoPilot (http://autopilot-project.eu/).

Data Provisioning

The TNO Central Repository collects the data from on-board units, IoT platforms and road side stations provided by participants. An explanation of the procedure to upload data and trigger processing is given here.

Experiment Results

Each session is defined as a single experiment for analysis. An experiment includes the analysis of communication, applications and HMI events within a single on-board unit, as well as the interactions and communication between on-board units and road side stations. The analysis results are accessible in two alternative ways:

- By experiment
- By analysis or format:
  - Database
  - Animation in Google Earth
  - Sanity Checks

Evaluation Results

The repository also provides access to share the results of the analyses and evaluations based on the participants' data.

A central repository for evaluation is provided at https://ada1.tno.nl/autopilot/.
A personal account should be requested from Harry.Wedemeijer@tno.nl.
Note that this repository only provides read access to the evaluation results.

Contact information: Harry.Wedemeijer@tno.nl

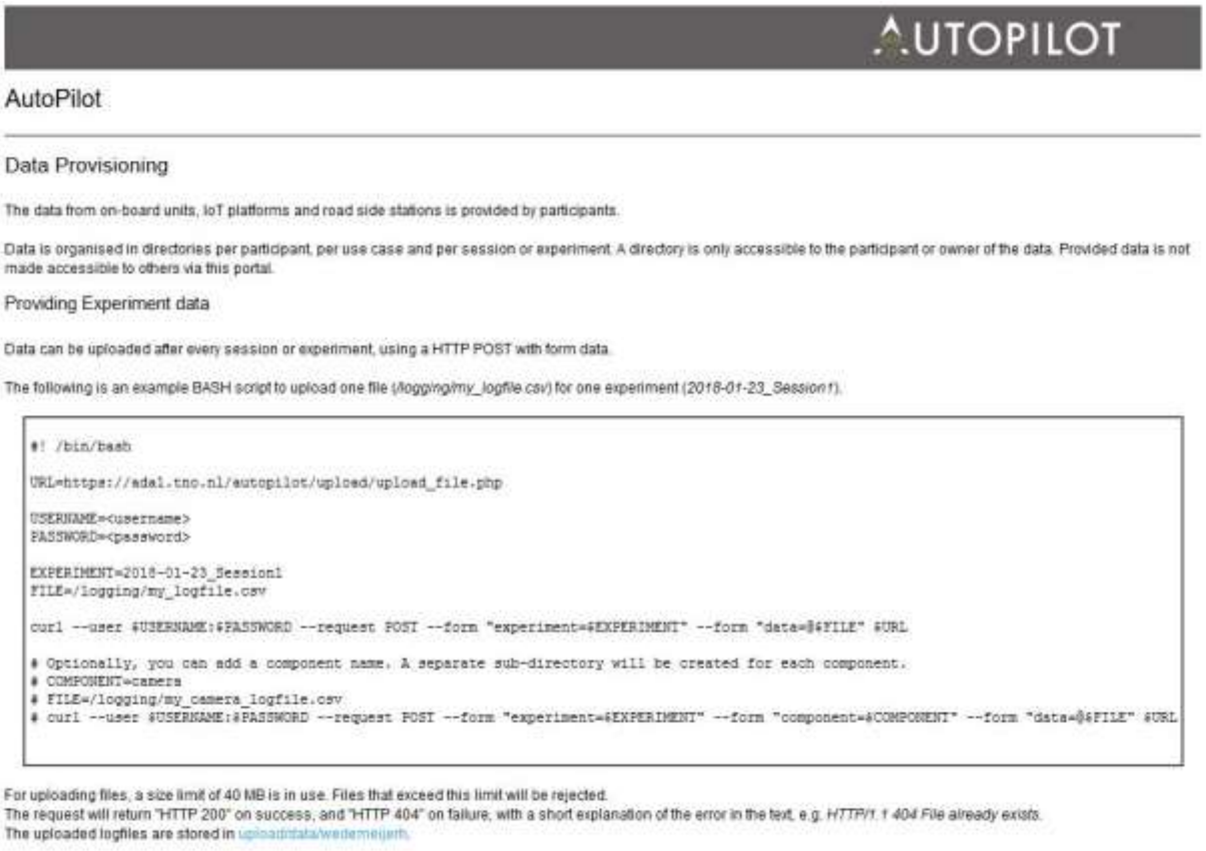
TNO innovation for life

Figure 18: Brainport PSTS web interface on https://ada1.tno.nl/autopilot

4.3.2 Data upload components

The web interface of the repository provides access for uploading log data automatically. The data provisioning section in Figure 18 links to Figure 19 on the following page which explains:

- The upload process with an example script to upload data automatically from any data source on the internet.
- Explanation how to organise log data into experiments.
- A section to define the experiment and, once all data of an experiment are uploaded, trigger the automated processing of the log data for the experiment analysis.


AutoPilot

Data Provisioning

The data from on-board units, IoT platforms and road side stations is provided by participants.

Data is organised in directories per participant, per use case and per session or experiment. A directory is only accessible to the participant or owner of the data. Provided data is not made accessible to others via this portal.

Providing Experiment data

Data can be uploaded after every session or experiment, using a HTTP POST with form data.

The following is an example BASH script to upload one file (*/logging/my_logfile.csv*) for one experiment (*2018-01-23_Session1*).

```
#! /bin/bash

URL=https://adal.tno.nl/autopilot/upload/upload_file.php

USERNAME=<username>
PASSWORD=<password>

EXPERIMENT=2018-01-23_Session1
FILE=/logging/my_logfile.csv

curl --user $USERNAME:$PASSWORD --request POST --form "experiment=$EXPERIMENT" --form "data=@$FILE" $URL

# Optionally, you can add a component name. A separate sub-directory will be created for each component.
# COMPONENT=camera
# FILE=/logging/my_camera_logfile.csv
# curl --user $USERNAME:$PASSWORD --request POST --form "experiment=$EXPERIMENT" --form "component=$COMPONENT" --form "data=@$FILE" $URL
```

For uploading files, a size limit of 40 MB is in use. Files that exceed this limit will be rejected.
The request will return "HTTP 200" on success, and "HTTP 404" on failure, with a short explanation of the error in the text, e.g. *HTTP/1.1 404 File already exists.*
The uploaded logfiles are stored in uploaddata/wedemeijerh.

Figure 19: Brainport PSTS web interface for data uploading

Every pilot site and use case has its own folder structure to organise log data. This folder structure is only accessible by the use case partners. A partner may also have a separate folder structure. The log folders are not accessible to other partners.

Log data is organised in experiments. An experiment is a single pilot test sessions or test run. All log data should be uploaded into a single experiment folder. The web interface allows the uploaded data to be viewed. The repository does not support sftp to upload or access log data.

4.3.3 Data processing components

Figure 20 shows the data processing flow for Automated Data Analysis (ADA) of the log data for an experiment. The process consists of six steps that will be explained in the following subsections.

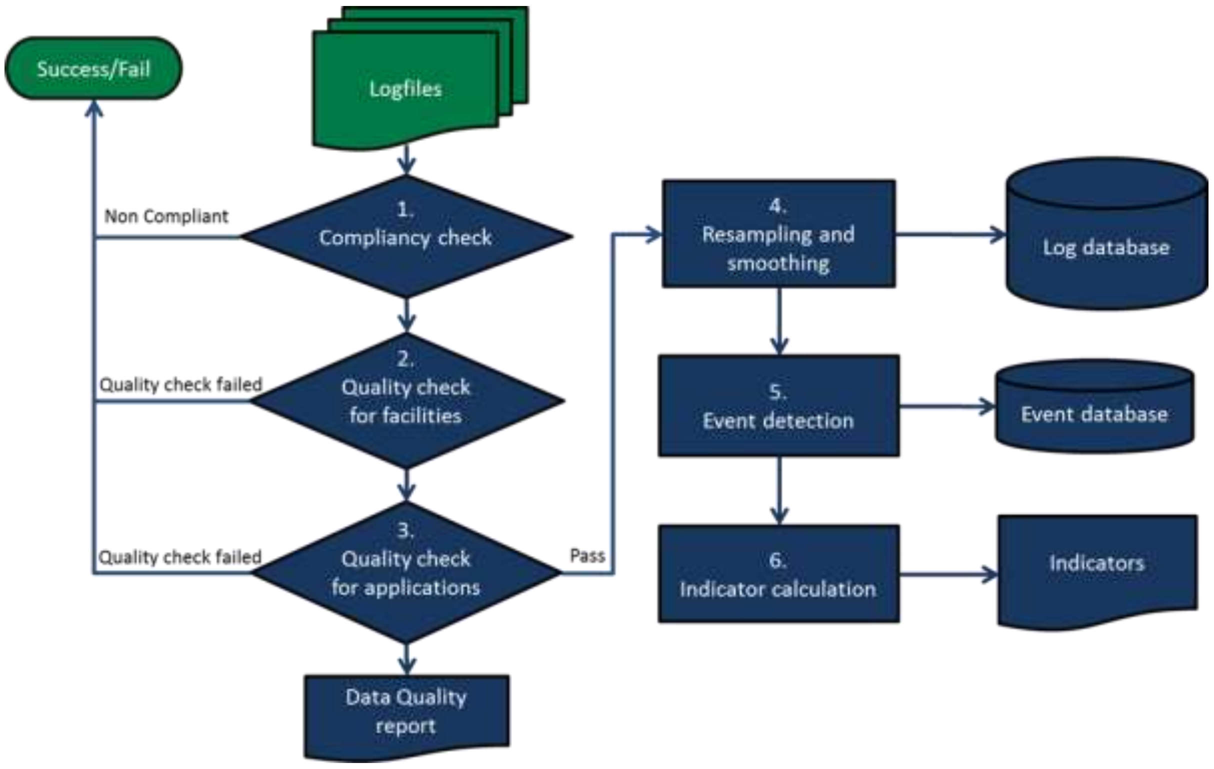


Figure 20: Brainport PSTS Data processing steps for Automated Data Analysis (ADA)

4.3.3.1 Quality check

The first three steps in Figure 20 are data quality checks.

1. The log data is first check on compliance to the parameter and file format specifications in section 3. This includes the availability of mandatory parameters, valid encoding of message payloads, value ranges.
2. The sanity and plausibility of basic station or device facilities are checked in the second step. These include the sanity or plausibility assessment of:
    a. Time offsets and time synchronisation between devices, components and logging units.
    b. Vehicle kinematics and the consistency between position, speed and accelerations from vehicle sensors and control systems, absolute (GPS) positioning systems, relative positioning systems, and target or object detection systems.
    c. Reconstruction of trajectories of devices and vehicles.
    d. Communication performance, such as the kinematic data in sent messages, relative positions of received messages, packet delivery ratios, effective communication ranges, time delays between generation, sent and reception timestamps.
3. The sanity and plausibility of application and control logic is checked. Known event locations are checked against vehicle trajectories to detect every vehicle passing. In the opposite direction every logged vehicle event is checked for consistency with internal and external event data. These checks are specific to applications and use cases.

The ADA processing tools do not attempt to detect any missing measurements other than missing mandatory data. ADA tools do not replenish any missing data.

4.3.3.2 Analysis
Step four in Figure 20 can be used to resample and smooth continuous parameters, and inserted as 'new' parameters in the repository, to ease further analysis. More often, however, such resampling is only executed with specific analysis scripts and not inserted into the repository.

4.3.3.3 Processing and enrichment
In step five, events and actions are detected from the logged application and control logic, and from the vehicle trajectories and vehicle passes detected in step three. Events, actions and situations are defined per use case and application.

In step six a basic set of predefined indicators are calculated for the events and actions detected in step five. Indicators that are calculated are, for example, the communication performance parameters from D4.1 such as communication latency, packet delivery ratio and effective communication range.

4.3.3.4 Metadata definition
Meta data are provided for the experiments, time periods, involved devices and their trajectories, events and indicators.

4.3.4 Data storage

Data is stored in PostgreSQL data bases per experiment. Data is stored in the structure defined in section 3. The data bases used for ADA analysis are not accessible by partners. The resulting database is provided as a database dump, including all logged data relevant to the ADA processing and the results of the ADA processing.

4.3.4.1 Raw data storage
Raw data, as mentioned in D3.6, is all logged data received from pilot experiments. All data from a single experiment is stored in a separate folder on the repository.

4.3.4.2 Processed data storage
Processed data is stored in a PostgreSQL data base per experiment.

4.3.5 Data access

Data access is controlled by access codes per pilot site, use case and partner.

Data can be accessed via the website (Figure 18) in the *"Experiment Results"* and *"Evaluation Results"* sections. The results can be accessed per experiment as database dump file for downloading and local processing, or in specific views, such as the data quality reports, animations, result summaries and data plots.

5 Livorno pilot site specificities

This section presents the Use Case managed by the Livorno pilot site, and the specific architecture of the pilot site components.

5.1 Architecture design rationale

This subsection presents the rationale of the global design, i.e. it describes and justifies the main decisions of design. The requirements and constraints define what the project expects from the architecture.

5.1.1 Non-functional requirements

Requirements below are the most important non-functional requirements this pilot site will comply (or adhere to), in order to ensure an optimal integration in the global AUTOPILOT project.

5.1.1.1 Design constraints

From an architecture standpoint, a constraint is an architectural design or implementation decision that has been selected to be treated as if it were a formal requirement.

Table 10: Livorno pilot site design constraints

ID	Title	Description	Source
NFRQ-DC-01	Networking	PSTS must be connected to the Internet.	D3.6
NFRQ-DC-02	Networking	PSTS must be connected to the PS VPN with a static IP address	D3.6
NFRQ-DC-03	IoT	PSTS must be able to send HTTP/GET request to the oneM2M platform.	D3.6
NFRQ-DC-04	IoT	PSTS must be able to parse the response to the GET with a function like JSON.parse().	D3.6
NFRQ-DC-05	Data collection	PSTS must be able to store the data requested to the oneM2M platform.	D3.6
NFRQ-DC-06	Data collection	PSTS must be able to store the data sent from devices with local storage (OBU/RSU, etc.) via FTP.	D3.6
NFRQ-DC-07	Data collection	PSTS can enable manual collection of test data using hard drive, flash drive, USB key	D3.6
NFRQ-DC-08	Data Storage	PSTS must use a database to store test data descriptions.	D3.6
NFRQ-DC-07	Data Transfer	PSTS must allow sending stored data to the CTS by FTP.	D3.6

5.1.1.2 Architecture quality goals

This section presents two kinds of quality requirements:

- Performance and Scalability requirements
- Availability and Reliability requirements

Table 11: Livorno pilot site non-functional quality requirements

ID	Title	Description	Source

NFRQ-QR-01	Data Interface	PSTS must send the Test Data according to interface defined by CTS (tar file and description file)	D3.6
NFRQ-QR-02	Data Storage / Backup	PSTS must provide a secured storage and backup to avoid data loss.	D3.6
NFRQ-QR-03	Data Storage	PSTS must provide enough storage space.	D3.6

5.1.1.3 IoT platforms requirements

This section presents the requirements related to the IoT platforms.

Note that these requirements mostly concern what is expected from the IoT platform. Some of them will drive the way the platform is used.

The security aspects of oneM2M platform are based on the following features:

- Services and APIs oneM2M are exposed with SSL (HTTPS)
- Authorisation mechanism based on credentials (username/password) of a specific user (tenant)
- Creation of Access Control Policy (ACP) for each Application Entity (AE)

Table 12: Livorno pilot site IoT platform requirements

ID	Title	Description	Source
NFRQ-IOT-01	compliance	The IoT platform must be compliant to the OneM2M standard - Release 2	D2.3, D3.6
NFRQ-IOT-02	Network	Resources must be identified by URI in separate way from IP addressing	D2.3
NFRQ-IOT-03	Network	The IoT platform should be IP based (irrelevant the version, IPv4 or IPV6)	D2.3
NFRQ-IOT-04	Network	The IoT platform must be network independent	D2.3
NFRQ-IOT-05	SW architecture	The IoT platform must support the REST approach	D2.3
NFRQ-IOT-06	interoperability	The IoT platform interfaces towards the applications (REST APIs) should be compliant with oneM2M standard	D2.3
NFRQ-IOT-07	management	IoT platform must support full device and subscription management	D2.3, D2.5
NFRQ-IOT-08	Protocols	The IoT platform should implement HTTP/COAP/MQTT transport protocols	D2.3
NFRQ-IOT-09	Data management	The IoT platform should implement the “Store and Share” paradigm with data historisation	D2.3, D3.6
NFRQ-IOT-10	Security	The IoT platform should implement some levels of assurance for authentication	D2.3, D3.6
NFRQ-IOT-11	Privacy by design	Identifiers used for communication in the M2M System should not be directly related to the real identity of either the	D2.3, D3.6

		device or its user.	

5.1.1.4 Other non-functional requirements

This section includes:

- Security requirements
- Legal requirements

Table 13: Livorno pilot site non-functional requirements

ID	Title	Description	Source
NFRQ-NF-01	Security	The PSTS should implement some levels of assurance for authentication	D1.9
NFRQ-NF-02	Legal	The PSTS should meet GDPR requirements	D4.9

5.1.2 Design rationale

This section presents the major design decisions for the Livorno PS.

5.2 Global architecture

This section presents the essentials of Livorno pilot site system architecture, including main API components, services and functionalities.

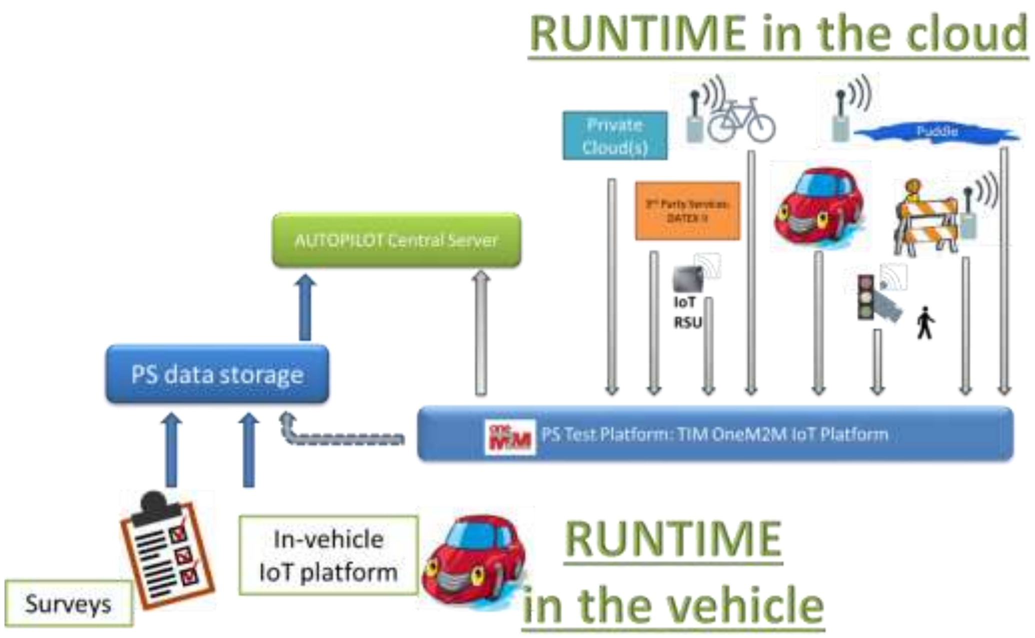


Figure 21: Data Management Global architecture at Livorno pilot site

5.3 Components architecture

This section presents the detailed components architecture.

5.3.1 Data upload components

The data collection components and raw data acquired are described in the following subsections.

The focus is about how devices, platforms and other sources data are uploaded to the test server platform.

5.3.1.1 IoT platforms

Since the Livorno PS IoT Platform is compliant to the OneM2M standard, it is based on a *"Store and Share"* resource-based paradigm. Thus, data produced during the piloting may be made available on the platform to the other applications, including uploading service of raw data in the PSTS.

The data are downloaded from the IoT platform with a customised function that combines REST methods (notably GET) with filters able to select time intervals, use cases and devices.

The IoT raw data are permanently stored in the IoT platform and can be accessed and downloaded any time. The downloaded data after filtering and quality checking are uploaded to the PSTS. No local storage between these two ICT infrastructures is maintained. The overall operations of data transferring will be performed by a human operator.

The data inside the PS IoT platform are structured according to the resource tree shown in Figure 22, Figure 23 and Figure 24.

All the devices with a role in the experimentation have a virtual representation on the oneM2M platform, notably the associated container and sub-container resources have specific attributes. Those attributes are both metadata describing the digital object itself, and the values of the variables of that object, which are called *"content"*.

Every time an IoT device publishes new data on the OneM2M platform a new *"content instance"* is generated, representing the actual status of that device.

All the *"content instances"* are stored in the internal database with a unique resource ID. They can be retrieved from other consumers, including PSTS with simple REST methods (notably GET, see D3.6).

In the Figure 22, Figure 23, Figure 24 the *"label"* metadata are represented by blue tags. Those tokens are used to add meta-information to resources that can be used for example for discovery purposes when looking for particular resources that one can "tag" using that label-key.

Note that structuring the data, as described here, is essential for IoT platform conception and configuration, and also for applications consuming data published by the devices into the IoT platforms. These inputs must be used as requirements for IoT platforms and applications and represent also a proposal for the semantics for the *"smart roads"* vertical domain.

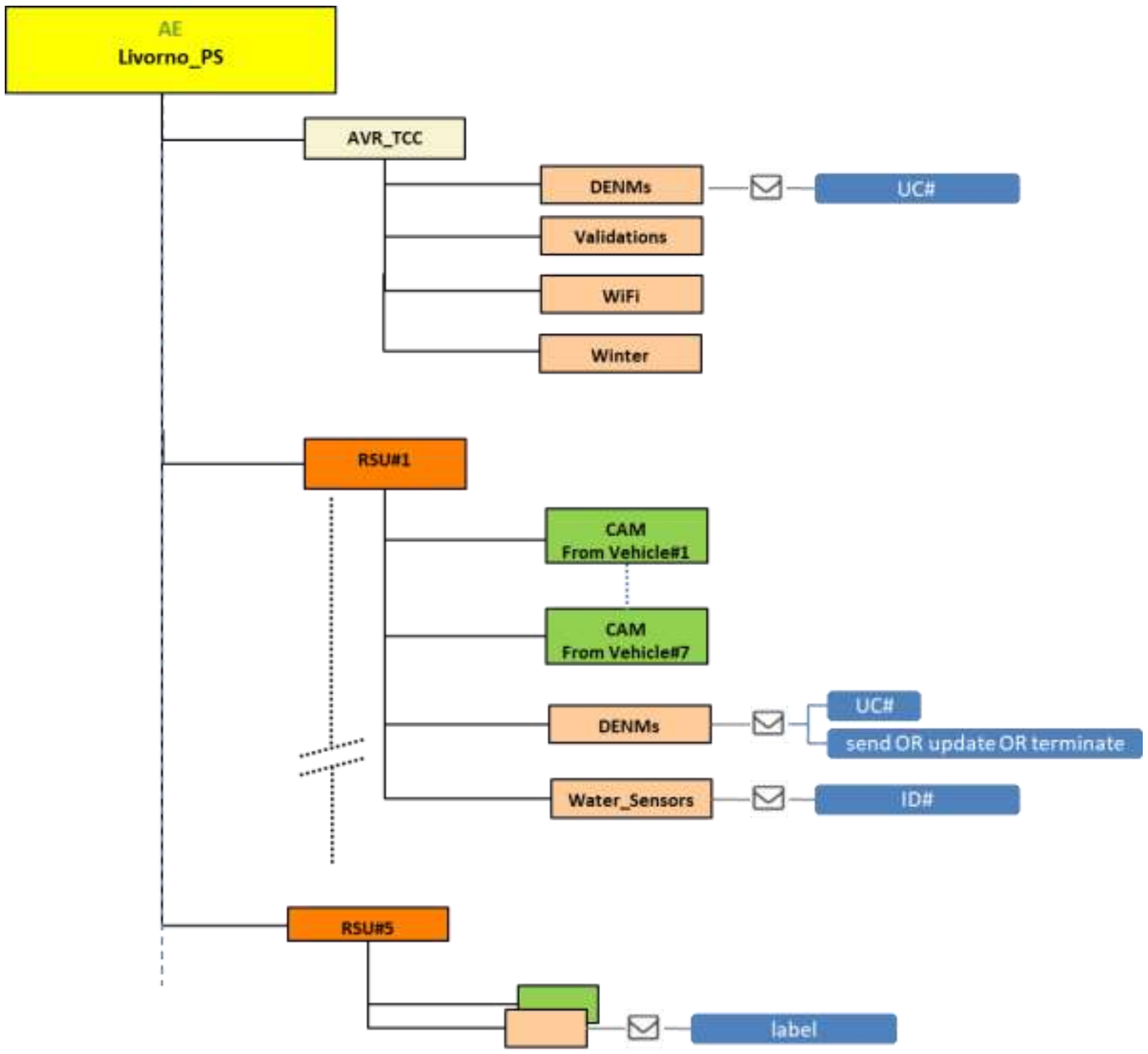


Figure 22: Data tree structure of IoT platform at Livorno pilot site (TCC and RSUs virtual entities)

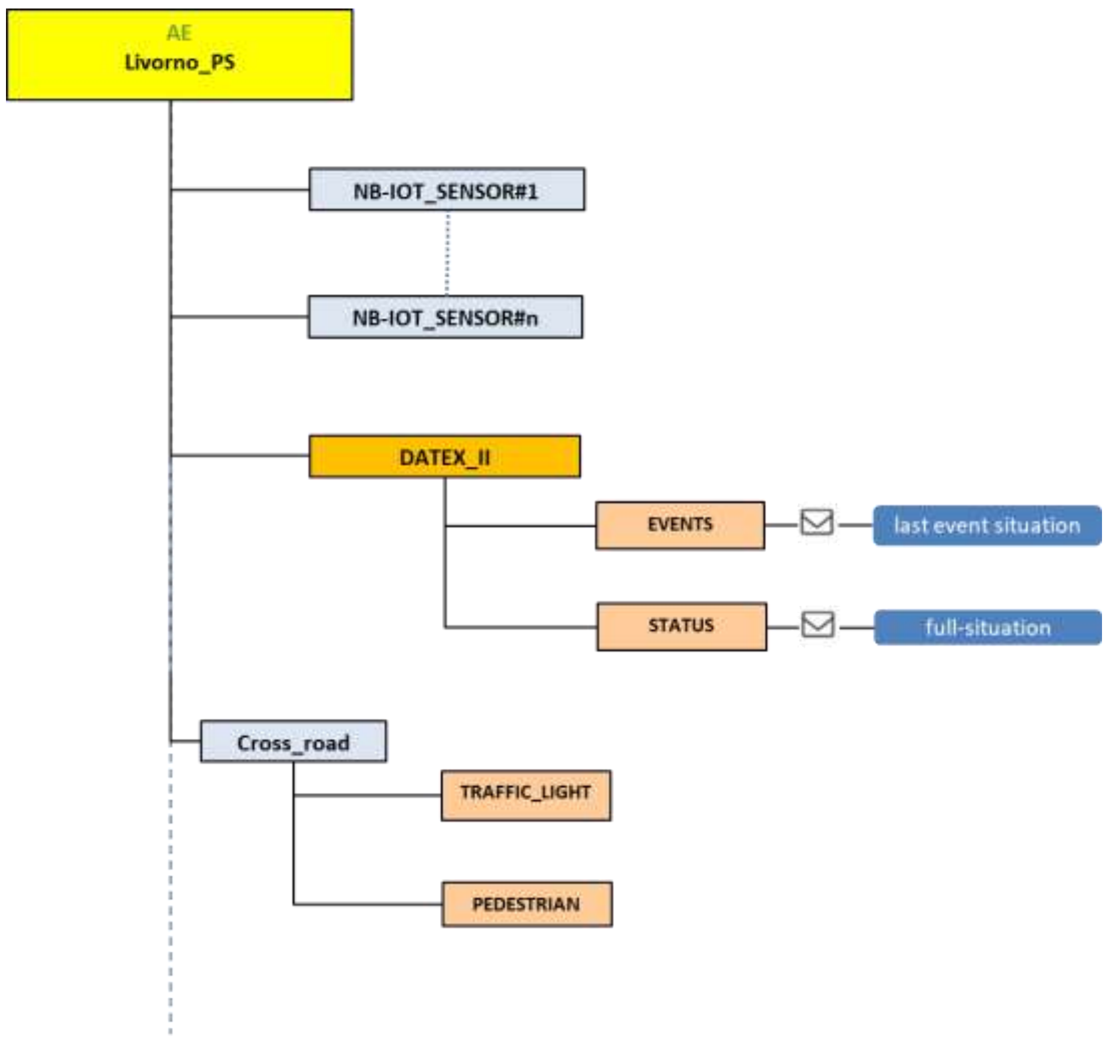


Figure 23: Data tree structure of IoT platform at Livorno pilot site (NB-IoT DATEX2 and Crossroad virtual entity)

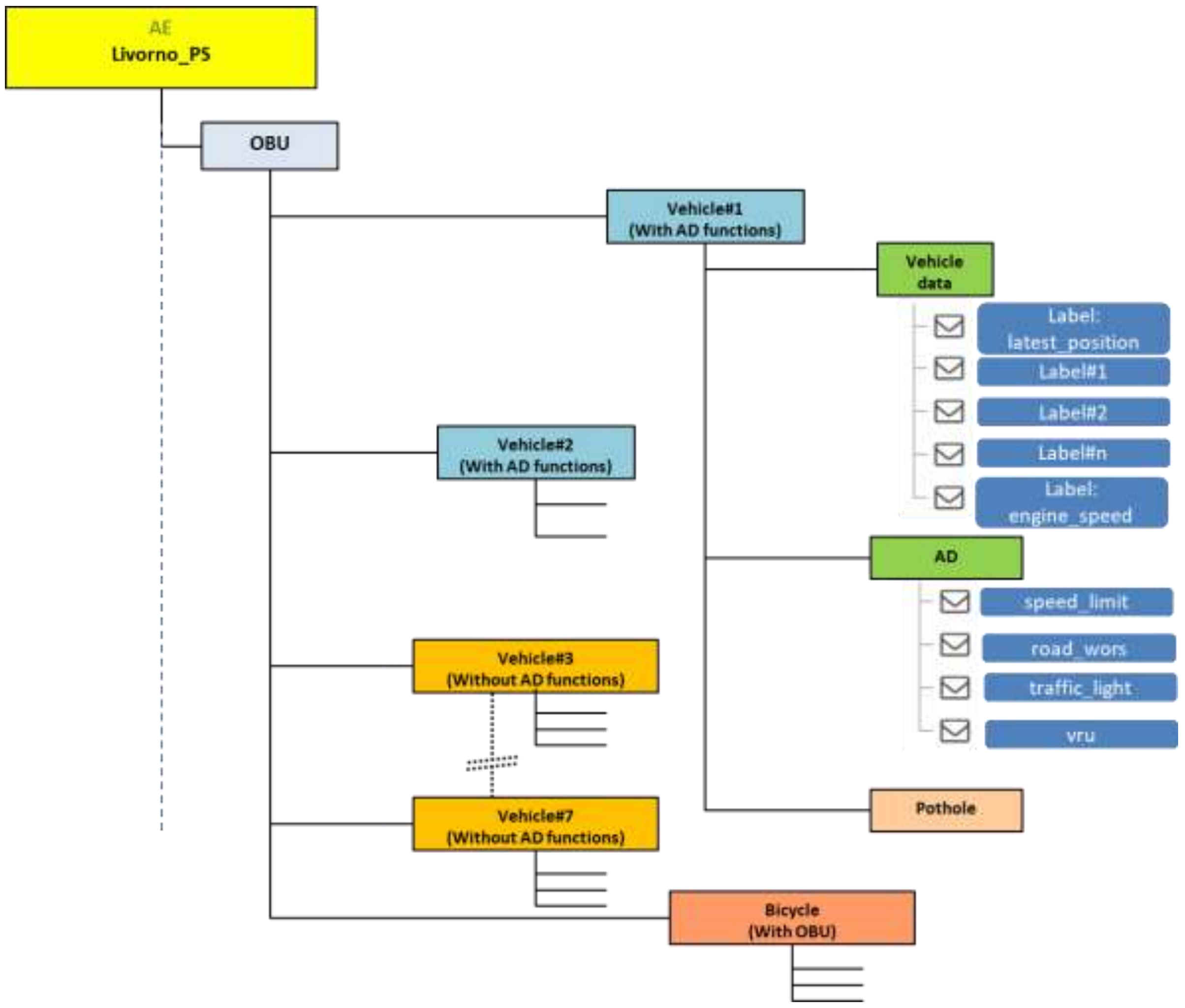


Figure 24: Data tree structure of IoT platform at Livorno pilot site (OBUs virtual entity)

5.3.1.2 Vehicle data

The vehicle data are stored during the piloting in the local in-vehicle IoT platform. In order to overcome the storage limitations of the OBUs, the logging is performed in a binary format so called *"protobuf"*.

At the end of a piloting session a specific API will decode the logs and send them to the PSTS in JSON format using SFTP transfer over the PS VPN.

5.3.1.3 Surveys data

For the survey data, the Livorno PS will use the tools and methods indicated by WP4, once available.

5.3.1.4 V2X messages data

The V2X message are generated and collected by both OBUs and RSUs: the former will store the collected CAMs and DENMs locally inside the in-vehicle IoT platform, the latter will publish at runtime the DENM and CAM messages on the oneM2M platform. At the end of the piloting session V2X messages from both kinds of ITS stations will be uploaded to the PSTS, according the abovementioned procedures.

5.3.2 Data processing components

In Figure 25 the raw data processing is shown: the different sources of raw data are filtered according to device ID, UC and time interval, then data quality is checked: only the data that are free from errors or inconsistencies are uploaded to the PSTS. The next step involves metadata generation using the tools provided by AKKA and finally the uploading to the CTS, where the data are stored and available to the evaluators.

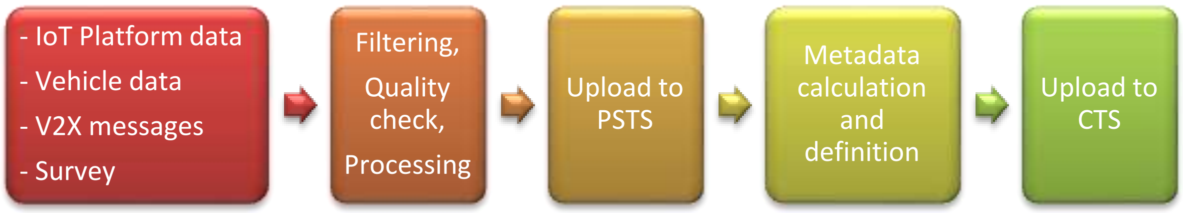


Figure 25: Data processing components at Livorno pilot site

5.3.2.1 Analysis

The data are filtered according to time intervals, use cases and devices. The data received from the oneM2M platform with GET methods are parsed with a JSON.parse() function in order to convert the "con" field in JSON objects array without '\' characters.

5.3.2.2 Quality check

The quality check operations will be performed manually before the uploading on the PSTS: the operations will include:

- Assessing and quantifying missing data
- Controlling data values and units of measure
- Checking that the data dynamic over time
- Guaranteeing that data fulfils specific hypotheses requirements

5.3.2.3 Processing and enrichment

The processing activities will include:

- Events detection
- PI calculation
- Data aggregation

5.3.2.4 Metadata definition

Metadata required by the CTS database are produced using the API provided by AKKA.

5.3.3 Data storage

In this section data storage components are described: filesystems, databases and data organisation.

5.3.3.1 Raw data storage

IoT raw data published on the TIM oneM2M platform are stored on a cluster of servers with MariaDB, so called MariaDB Galera Cluster (see Figure 26**).** It is fully read-write scalable, comes with synchronous replication, allows multi-master topologies, and guarantees no lag or lost transactions. Some of its features & benefits are listed below:

- Synchronous replication

- Active-active multi-master topology
- Read and write to any cluster node
- Automatic membership control, with failed nodes dropped from the cluster
- Automatic node joining
- True row-level parallel replication
- Direct client connections, native MariaDB/MySQL look & feel

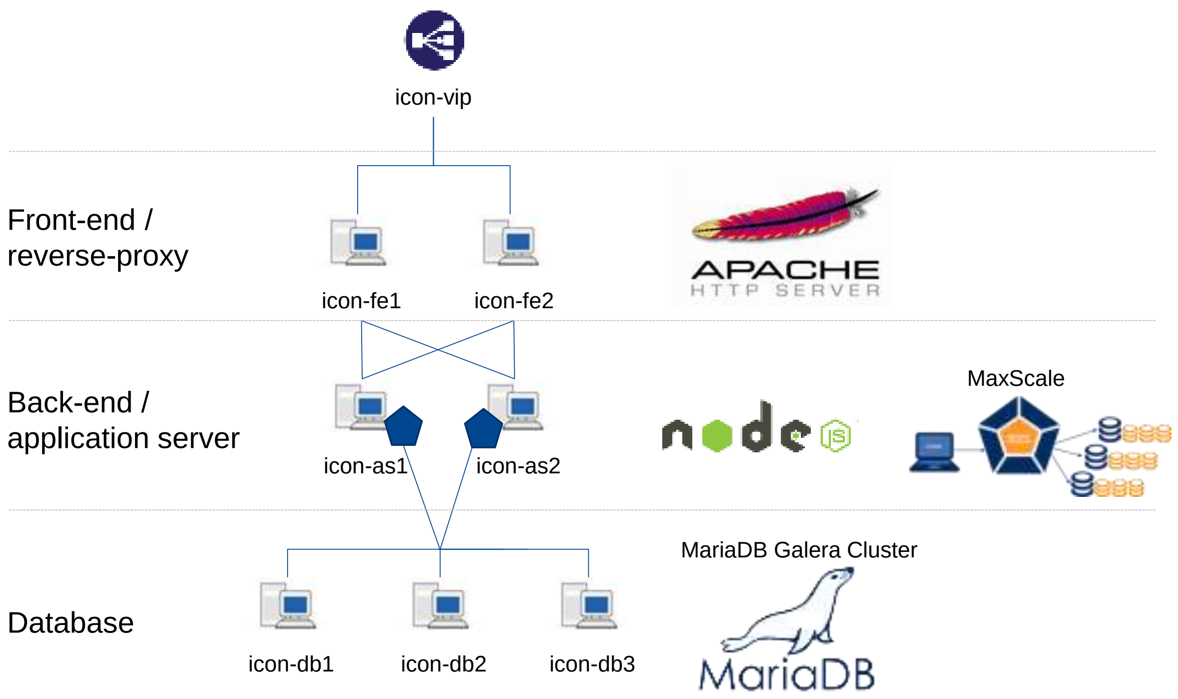


Figure 26: Data Three-tier architecture and technologies (Source TIM Internal documentation) at Livorno pilot site

5.3.3.2 Processed data storage

The processed data (i.e. data ready for transfer to CTS) are stored in sever NAS, notably a Network-attached storage. It is a file-level computer data storage server connected to a computer network providing data access to a heterogeneous group of clients. The system contains more storage drives, arranged into logical, redundant storage containers or RAID. It is configured in such a way, a single bad block on a single drive can be recovered completely via the redundancy encoded across the RAID set. The data are organised in archives generated by the tool provided by AKKA, which at the same time create the archive and the metadata descriptor.

6 Tampere pilot site specificities

This section presents the data management of the Tampere pilot site, and the related architecture.

6.1 Architecture design rationale

This subsection presents, describes and justifies the main design decisions. The following requirements and constraints define what the project expects from the architecture.

6.1.1 Non-functional requirements

Requirements stated below are the most important non-functional requirements this pilot site will adhere to, in order to ensure an optimal integration in the AUTOPILOT project.

6.1.1.1 Design constraints

Table 14: Tampere pilot site design constraints

ID	Title	Description	Source
NFRQ-DC-01	Networking	PSTS must be connected to the Internet	D3.6
NFRQ-DC-02	Networking	PSTS can enable manual collection of test data using hard drive, flash drive, USB key	D3.6
NFRQ-DC-04	Data Storage	PSTS should use a database to store test data descriptions.	D3.6
NFRQ-DC-06	Data Storage	PSTS must have a storage filesystem.	GA – DMP, D3.6
NFRQ-DC-03	Data Transfer	PSTS must allow sending stored data by FTP to the CTS	D3.6
NFRQ-DC-08	Test Data Interface	PS must send data that are compliant to the provided description.xml	D3.6 decision reused at PS level

6.1.1.2 Architecture quality goals

This section presents two kinds of quality requirements:

- Performance and scalability requirements
- Availability and reliability requirements

Table 15: Tampere pilot site non-functional quality requirements

ID	Title	Description	Source
NFRQ-QR-01	Data Interface	PS Test Server must send the Test Data according to interface defined by CTS (tar file and description file)	D3.6
NFRQ-QR-02	Data Storage / Backup	PSTS must provide a secured storage and backup to avoid data loss.	D6.9, D3.6
NFRQ-QR-03	Data Storage	PSTS must provide enough storage space.	D3.6
NFRQ-QR-05	Data Storage	PS Test Server must store IoT, vehicle and survey data as they are provided	D4.1
NFRQ-QR-08	Data synchronisation	All PS data sources must be synchronised with each other (within 1 second to enable analyses)	D4.1 & D3.6

6.1.1.3 IoT platforms requirements

This section presents the requirements related to the IoT platforms.

Table 16: Tampere pilot site IoT platform requirements

ID	Title	Description	Source
NFRQ-IOT-01	Platform Availability	The IoT platform must be available when running test sessions.	D2.3 & D3.2
NFRQ-IOT-02	IoT Platform standard	The IoT platform must provide standard oneM2M implementation	D2.3
NFRQ-IOT-05	Platform Storage Capabilities	The IoT platform must provide storage means until data are collected by the pilot site	D3.6
NFRQ-IOT-06	Data logging	The IoT platform must provide logging of events	D4.1
NFRQ-IOT-07	Data synchronisation	The IoT platform must be synchronised with the other devices at the pilot site	D4.1
NFRQ-IOT-08	Data synchronisation	The IoT platform must add a timestamp upon data/measure reception	D4.1

6.1.1.4 Other non-functional requirements

Table 17: Tampere pilot site non-functional requirements

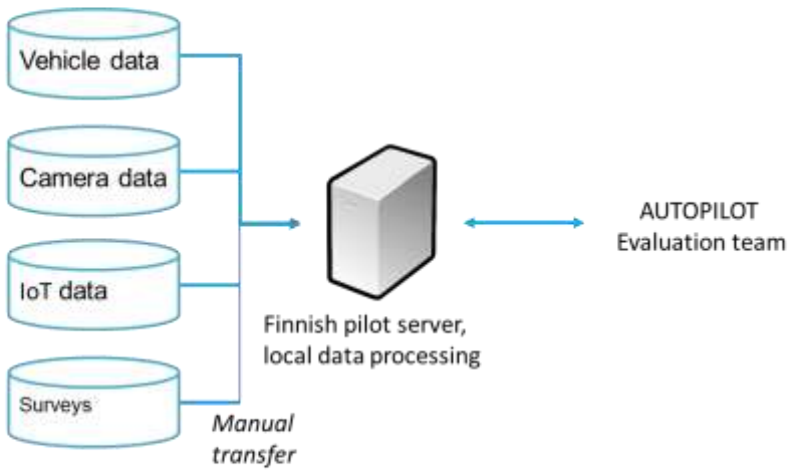
ID	Title	Description	Source
NFRQ-NF-01	Data privacy, legal	Collected data, especially surveys, must be anonymised before transferring it to the central storage.	D6.9

6.1.2 Design rationale

Data collection at the pilot site enables later analysis of key performance indicators from vehicle & IoT data. It also covers carrying out user surveys. The vehicle and IoT data are logged as .csv files of defined formats. The files will be collected manually to a test site server, where they are checked for quality and made available for further evaluation.

6.2 Global architecture

Figure 27 shows the architecture of the Finnish test data architecture. The data will be collected locally by the different components, and then transferred manually to the Test Site Pilot Server, where quality checks are performed, and from there it is transferred to the AUTOPILOT Central Test Server.



Vehicle data
Camera data
IoT data
Surveys
Manual transfer
Finnish pilot server, local data processing
AUTOPILOT Evaluation team

Figure 27: Tampere pilot site Test Data architecture

6.3 Architecture components

6.3.1 Data upload components

The data collection components and raw data acquired are described in the following subsections. The focus is about how devices, platforms and other sources data are uploaded in the test server platform.

6.3.1.1 IoT platforms

The data at the IoT platform will be stored in a CSV file, and downloaded at the end of each piloting session to the PSTS.

6.3.1.2 Vehicle data

A component has been developed that stores the vehicle data made available over DDS (Data Distribution Service – internal communication between vehicle components) in csv files. Log data files will be transferred manually at the end of each piloting session to the PSTS.

6.3.1.3 Survey data

For the survey data, the Tampere pilot site will use the tools and methods indicated in WP4, once available. In case paper surveys are deemed necessary, the pilot site will transfer the data to the selected electronic/online tool, in order to collect all data in a harmonised format (e.g. csv or Excel).

6.3.1.4 V2X messages data

V2X messages are generated and collected by the in-vehicle ITS-G5 OBUs. The in-vehicle OBU makes the data available for other applications through DDS. The V2X message data are hence included in the vehicle data logs. However, no CAM and DENM messages are planned to be used in the Tampere pilot.

6.3.1.5 Other data

The mobile roadside unit, on which the traffic camera is installed, has a similar in-vehicle IoT platform as the vehicle. A similar data collection component as for the vehicle will be used to store the data from the roadside unit. Only events will be stored, not the actual camera images and video. Event data is anonymous.

6.3.2 Data processing components

The data of the different data sources will be transferred manually to the PSTS using physical means such as USB sticks and hard drives. The data will be stored in different directories, with names based on the day of measurement. Additionally, the .csv data will be imported to a PostgreSQL database. Several quality checks, both manual plots and database queries, will be performed in order to provide a clean and ready-to-use data for evaluation.

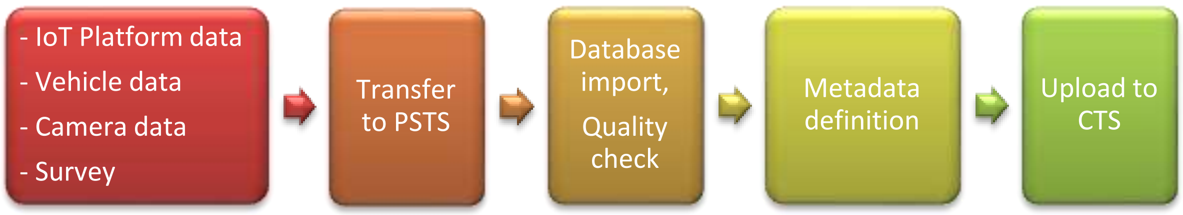


Figure 28: Data processing components at Tampere pilot site

6.3.2.1 Quality check

Quality checking at the pilot site ensures that

- Test equipment has not experienced glitches after pre-tests and data collection continues to work well
- Data does not contain values indicating strange system behaviour, e.g. vehicle speed above 300 km/h. If the data contains such values, such behaviour will be documented.
- Data is mainly continuous instead of having missing periods.

Additionally, during pre-testing, quality checking ensures that:

- Data cross-references exist (IDs match between tables) and data is synchronised to a reasonable level of accuracy to enable analyses over various logs
- Data plots show consistent system and log behaviour, capturing test periods
- Example indicators such as mean speed or specified distances can be calculated.

Quality checking during pre-testing requires collaboration with analysts.

Quality testing during actual tests includes running defined database scripts after test data has been first imported to PostgreSQL. This includes e.g. calculating logging frequencies for different signals and seeking values outside defined normal ranges. Selected variables will also be plotted manually to visually inspect data. Quality checking should be performed frequently during user test weeks to ensure that data is not missed and tests do not have to be repeated.

6.3.2.2 Processing and enrichment

At Tampere pilot site, processing and enrichment is limited to providing mandatory data and documentation to describe its format.

6.3.2.3 Metadata definition

At the level of the pilot site, metadata definition will be limited to providing a complete description of the Test Data described in D3.6

6.3.3 Data storage

Both the raw and the imported PostgreSQL data will be stored at VTT's servers.

6.3.3.1 Raw data storage

Regarding security, the test equipment will almost continuously be monitored by test site personnel. Additionally, collected data poses no specific confidentiality (product or company) risks. The data includes personal data, but of no sensitive nature, since the tests are controlled tests in specified test areas, and the test subjects will have signed a consent form accepting scientific use. Therefore, no specific measures are necessary to protect data during its collection.

After each test day, data is collected and sent to a test site server and deleted from logging equipment (both vehicle and roadside). The data server resides within VTT's premises, which are not accessible to the public, and only named persons have access rights to operate the computer. After the project ends, data will additionally be encrypted, so that even if the computer would be hacked or otherwise accessed, the data would be extremely difficult to open.

6.3.3.2 Processed data storage

Processed data storage at the test site covers mainly the data imports to a PostgreSQL database and cleaned raw logs, made available for evaluation. After the project ends, related database tables will be backed up as files and stored together with evaluation/raw data and general test site documentation. Depending on analyses, such databases could also contain key performance indicators derived from raw data.

6.3.3.3 Metadata storage and user consent forms

Test site documentation is stored together with data that is made available for analysts. The documentation is public within the consortium.

Consent forms signed by the test subjects are stored in a secure location, separate from log data. The forms are kept as long as the stored log data contains personal data. In case the data is anonymised or destroyed, the consent forms would be destroyed as well.

7 Vigo pilot site specificities

This section presents the use case managed by Vigo pilot site, and the specific architecture of the pilot site components.

7.1 Architecture design rationale

This subsection presents the rationale of the global design, i.e. it describes and justifies the main design decisions. The requirements and constraints define what the project expects from the architecture.

7.1.1 Non-functional requirements

Requirements below are the most important non-functional requirements this pilot site will comply (or adhere to), in order to ensure an optimal integration in the global AUTOPILOT project.

7.1.1.1 Design constraints

Table 18: Vigo pilot site design constraints

ID	Title	Description	Source
NFRQ-DC-01	IoT	PSTS must be connected to the Watson IoT Platform™.	D2.3
NFRQ-DC-02	Data collection	PSTS must be able to store data going through the IoT platform.	D3.6
NFRQ-DC-03	Data Storage	PSTS must use a database for storing and accessing parking test data.	D4.1 & D3.6
NFRQ-DC-04	Data collection	PSTS must provide tools for cleaning stored parking data	D4.1 & D3.6
NFRQ-DC-05	Data collection	PSTS must guarantee internal consistency of the data	D4.1 & D3.6
NFRQ-DC-06	Data Transfer	PSTS must allow subscribing to the parking data coming from the Watson IoT Platform™.	D2.3
NFRQ-DC-07	Data Access	PSTS must allow access parking data via the REST API based on unique identifiers of the parking data sources	D2.3
NFRQ-DC-08	Data Access	PSTS must allow access parking data via the REST API based on geospatial queries	D2.3
NFRQ-DC-09	Data Format	Data must be stored in the server in a readable format following the requirements in D4.1	D4.1 & D3.6
NFRQ-DC-10	Data model	CAM, DENM and SPAT must be stored in agreed data format and data model: ITS G5	D4.1 & D3.6
NFRQ-DC-11	Data Transfer	PS must send the data collected in the local server to the centralised server of the project	D4.1 & D3.6

7.1.1.1.1 Design constraints of the Parking Spot Service

This section describes design constraints of the Vigo PSTS related to the Parking Spot Service. These constraints guarantee a correct collection and storage of test data. The Watson IoT Platform™

should be used as a basic IoT platform and it should be connected to the IBM CloudAnt™ database for data storing. The overview of the PSTS is presented in Figure 29. First, all data from IoT devices are being sent to the Watson IoT Platform™. The IBM Parking Server registers itself as a subscriber to the parking messages with the IoT platform. Every message published by IoT devices passes through the IoT platform and is received by the server. The server checks quality and consistency of the message and stores correct messages in the IBM CloudAnt™ database. The server also provides interfaces for accessing existing messages.

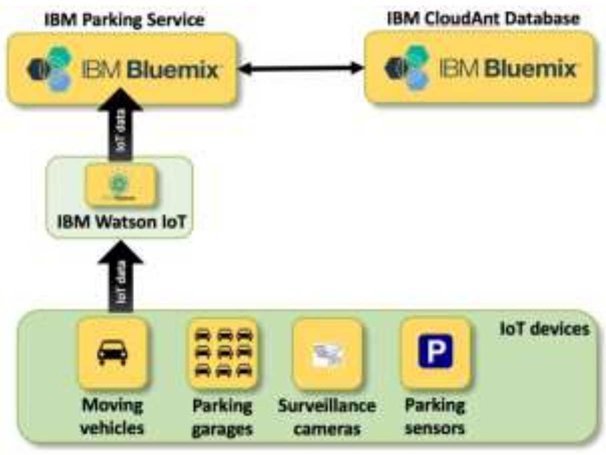


Figure 29: Architecture of the Vigo Parking Spot Testing Server

7.1.1.2 Architecture quality goals

This section presents two kinds of quality requirements:

- Performance and Scalability requirements
- Availability and Reliability requirements

Table 19: Vigo pilot site non-functional quality requirements

ID	Title	Description	Source
NFRQ-QR-01	Data Storage	PSTS must provide secure storage for the services information	D3.6
NFRQ-QR-02	Data Storage	PSTS must provide enough storage space to save the services information	D3.6
NFRQ-QR-03	Data collection	PSTS must support different IoT devices	D2.3
NFRQ-QR-04	Data collection	PSTS must support data handling with a regular time interval of five seconds	D2.3

7.1.1.2.1 Architecture quality goals of the Parking Spot Service

This section describes non-functional quality requirements of the Parking Spot Service. These requirements should guarantee the correctness, stability and scalability of the Parking Spot Service. These requirements are aligned with the parking service use case definition. In particular, the PSTS must allow different data sources (e.g., parking garages) to send parking data within regular time intervals (five seconds between consecutive calls from one IoT device). All parking data must be securely stored.

7.1.1.3 IoT platforms requirements

This section presents the requirements related to the IoT platforms.

Table 20: Vigo pilot site IoT platform requirements

ID	Title	Description	Source
FRQ-IOT-01	Platform Availability	The IoT platform must be available during all test sessions	D2.3
FRQ-IOT-02	Platform Availability	The IoT platform must be available for one week before test sessions for parking data collecting for development purposes.	D3.2
FRQ-IOT-03	Platform Availability	The IoT platform must remain available after test sessions for parking data collecting	D2.3
FRQ-IOT-03	Platform Storage Capabilities	The IoT platform must provide enough storage until parking data are collected	D2.3
FRQ-IOT-04	Data logging	The IoT platform must provide logging of events.	D2.3
FRQ-IOT-05	Data synchronisation	The IoT platform must be synchronised as any IoT device.	D2.3
FRQ-IOT-06	Data synchronisation	The IoT platform must add a time stamp upon data/measure reception.	D2.3
FRQ- IOT-07	Data Transfer	The IoT platform must provide a way to register IoT devices acting as data sources for the parking information	D2.3
FRQ- IOT-08	Data Transfer	The IoT platform must provide a way to publish data to the parking data topics for registered IoT devices	D2.3
FRQ- IOT-09	Data Transfer	The IoT platform must provide a way to subscribe for parking information topics	D2.3
FRQ- IOT-10	Data Transfer	The IoT platform must provide a way to receive parking information based on existing subscriptions for the parking data topics	D2.3

7.1.1.3.1 IoT platform requirements of the Parking Spot Service

This section describes the requirements of the Watson IoT Platform™ which has been selected as the basic IoT platform for the Parking Spot Service. The IoT Platform should guarantee a safe, reliable and scalable way for publishing and subscribing to the parking information coming from the IoT devices.

7.1.1.4 Other non-functional requirements

This section might include for example:

- Operational and Environmental requirements
- Security requirements
- Legal requirements

Table 21: Vigo pilot site non-functional requirements

ID	Title	Description	Source
NFRQ-NF-01	Data privacy	Collected data, especially surveys, must be anonymised before storage.	D6.9
NFRQ-NF-02	Data privacy	All private information must be removed from the parking data message before publishing to the Watson IoT Platform™.	D6.9

7.1.1.4.1 Other non-functional requirements of the Parking Spot Service

The parking PSTS assumes all parking information messages do not contain any private information and are anonymised before being sent to the Watson IoT Platform™.

Table 22: Vigo pilot site non-functional requirements of the Parking Spot Service

ID	Title	Description	Source
NFRQ-NF-01	Data privacy	All private information must be removed from the parking data message before publishing to the Watson IoT Platform™.	D6.9

7.1.2 Design rationale

In the Vigo pilot site there are two use cases: urban driving and automated valet parking. These two services are going to be tested in controlled test environments. This type of testing clearly conditions the type of logging. The part of the services and systems deployed that are in the vehicle are going to be registered predominantly using a CAN format. The files obtained are going to be collected manually from the vehicle and sent to the pilot site server. This server is going to also receive the V2X log files from the infrastructure and the IoT platform. Once this data is in the server the different files are going to be processed in order to prepare the data for the analysis defined in WP4.

7.2 Global architecture

This section presents the essentials of Vigo pilot site system architecture, including main API components, services and functionalities.

The following chart represents the main items participating in the data collection in the Vigo pilot site. These different items will upload the data through different interfaces to the CTAG central server.

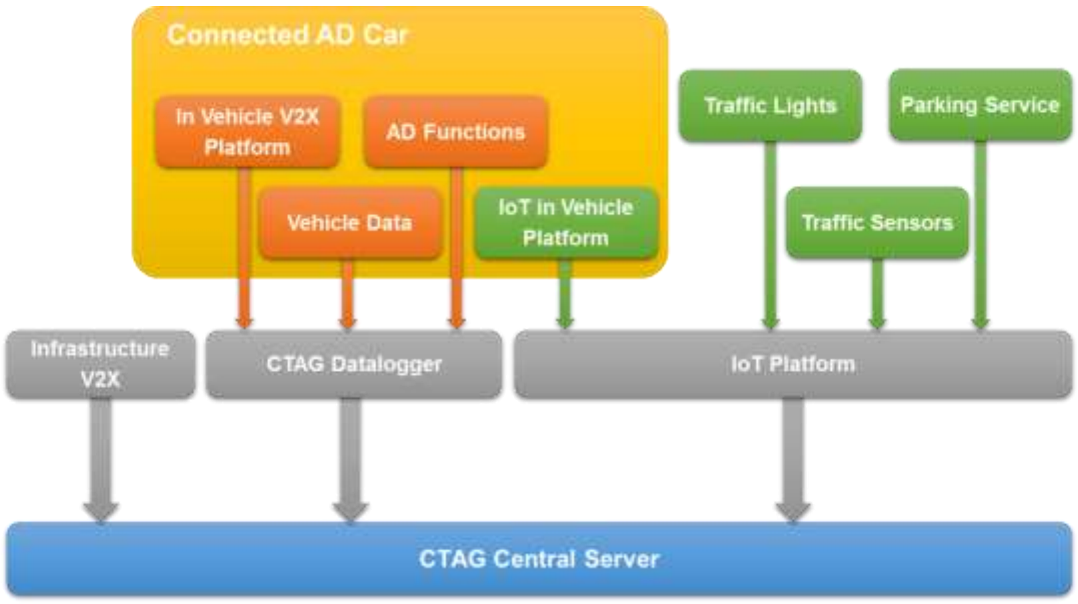


Figure 30: Vigo pilot site Test Data architecture

Once the data is uploaded to the central server, a harmonisation and quality check process is launched according to the project requirements.

After this process, the data can be sent to a project central server to be analysed by the evaluators.

7.3 Components architecture

The following pictures describe the main components in the architecture of the two services deployed in the Vigo pilot site.

Figure 31 describes the data management process for the service *"Urban driving"*. In this case it can be observed how the main actors (camera, traffic light, traffic server and vehicle) are connected to the IoT platform.

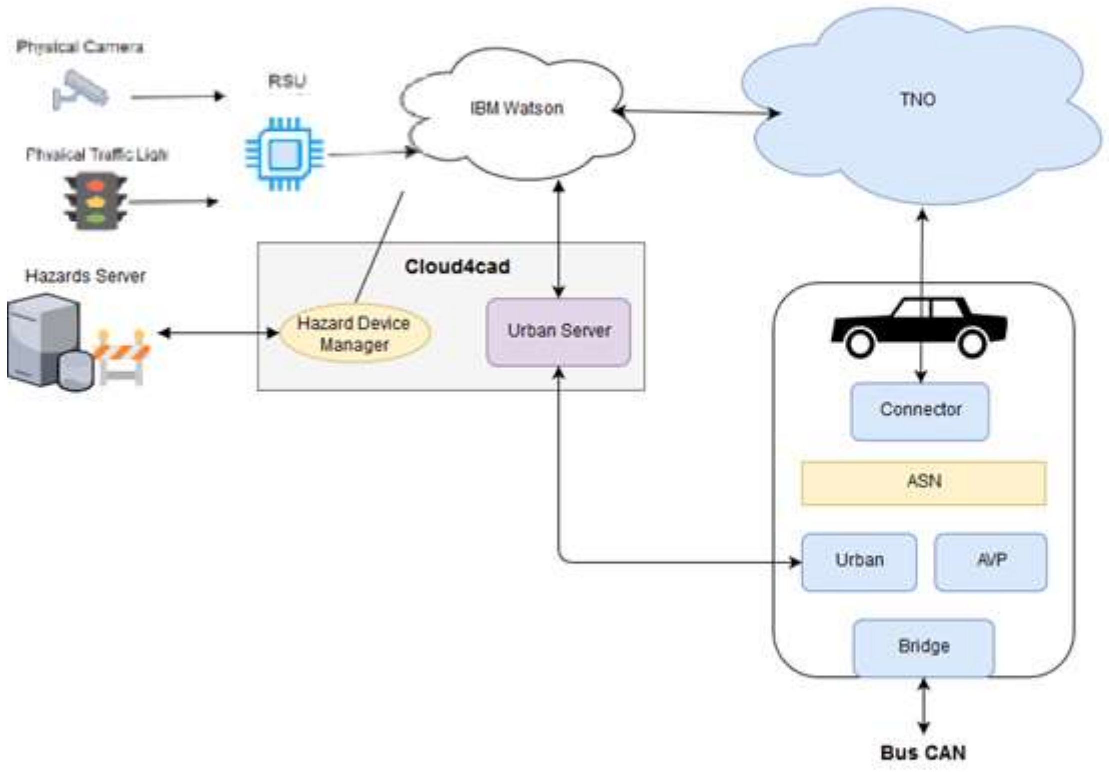


Figure 31: Data Management architecture at Vigo pilot site for urban driving

Figure 32 shows the data architecture for the service *"Automated valet parking"*. In this case the main actors are the parking management service, the mobile application and the vehicle.

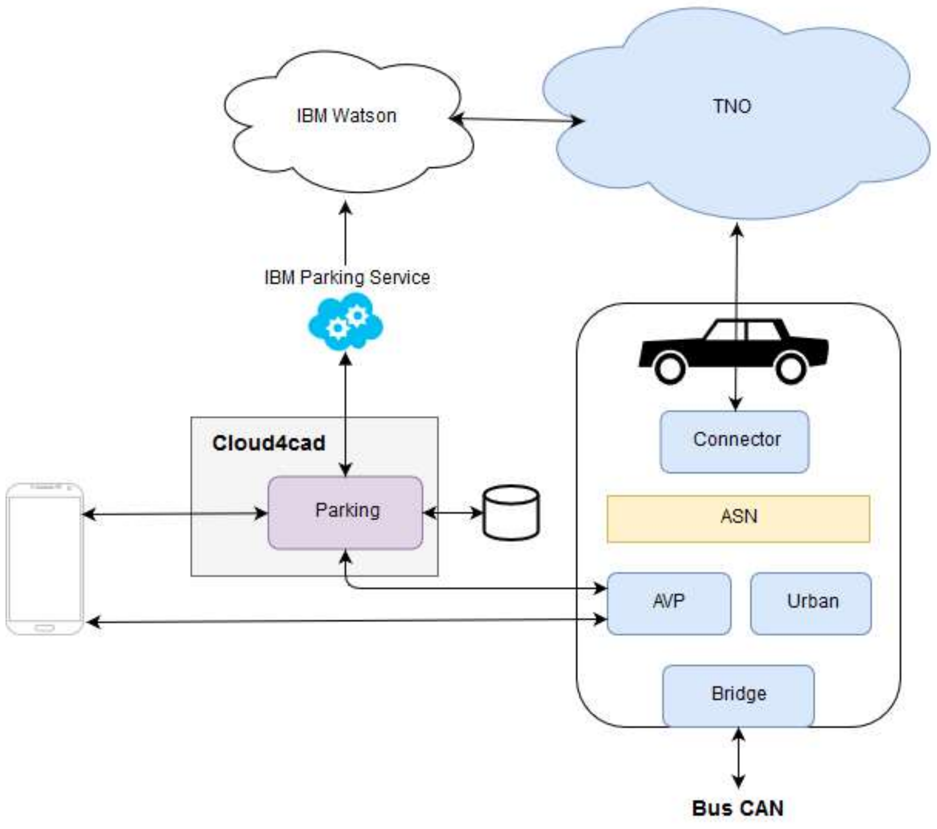


Figure 32: Data Management architecture at Vigo pilot site for AVP

7.3.1 Data upload components

This section describes the main components used for uploading data for the services in the Vigo pilot site.

7.3.1.1 IoT platforms

The Parking Spot Service uses the Watson IoT Platform™ for communication between parking information sources and the parking spot server. The AUTOPILOT project uses the Watson IoT platform owned by the IBM Ireland project member. The Watson IoT Platform™ assigns a unique identifier for each registered data provider or data subscriber. This identifier is used for publishing and subscribing to the messages coming from/to the IoT platform.

In order to use the Watson IoT Platform™ one should create a new type of an IoT device which acts as a source of parking information. This operation allows real parking IoT devices at the Watson IoT Platform™ to be registered. Because of this operation, each IoT device gets an authentication token and an authentication key. These values are used by the Watson IoT Platform™ for authentication of all agents which can publish and subscribe messages.

In the case of the Parking Spot Service the Vigo pilot site parking garages act as a source of parking data, and publish information to the IoT platform. From the other side the parking spot server subscribes to all published parking messages. These messages should be verified and stored in the

IBM CloudAnt™ database. The following table contains settings which should be used for publishing and subscribing the parking messages.

Table 23: Vigo Watson IoT Platform settings

ID	Title	Description
1	IBMReParkingTable	This type of IoT devices is used for all messages which contain information about parking tables.
2	IBMReParkingStatus	This type is of IoT devices used for all messages which contain information about occupation statuses of a parking table.
3	authentication token / authentication key	This parameter is used to publish / to subscribe for messages within the Watson IoT Platform™.
4	ID of an IoT device	This is a unique identifier of an IoT device of a certain type.

The Parking Spot Service should be subscribed to parking messages, check consistency of these messages and store them in a database. In addition, the PSTS should provide a RESTful API to access parking data. All parking messages should be aligned with the DATEX II data model format.

The parking server should support two types of data models: parking tables and statuses of parking tables. The first model describes a group of parking spots, their location and properties. The second type consists of a link to the parking table and its occupancy status.

The RESTful API functionality of the testing server should allow adding, updating, getting and deleting parking tables. In addition, it should provide interfaces for adding and retrieving parking table statuses. Access to the parking tables or statuses must be possible via unique identifiers or geographical locations.

The architecture of the parking database should be optimised for efficient data storage and fast querying of parking information. The database should store all original messages and build additional geospatial indexes for geographical queries.

7.3.1.2 Vehicle data

All the data required from the vehicles will be accessed through a CAN interface and stored in the local server.

7.3.1.3 Surveys data

All surveys data will be collected through individual forms and stored manually in the local server complying with the General Data Protection Regulation.

7.3.1.4 V2X messages data

Both in the infrastructure (RSU) and vehicles (OBU) the V2X data will be stored following the ETSI G5 standard. Every type of message will generate a file containing all the data fields described in the standard.

All this text files will be synchronised with the local server.

7.3.2 Data processing components

The following image describes the data processing components in the Vigo pilot site:

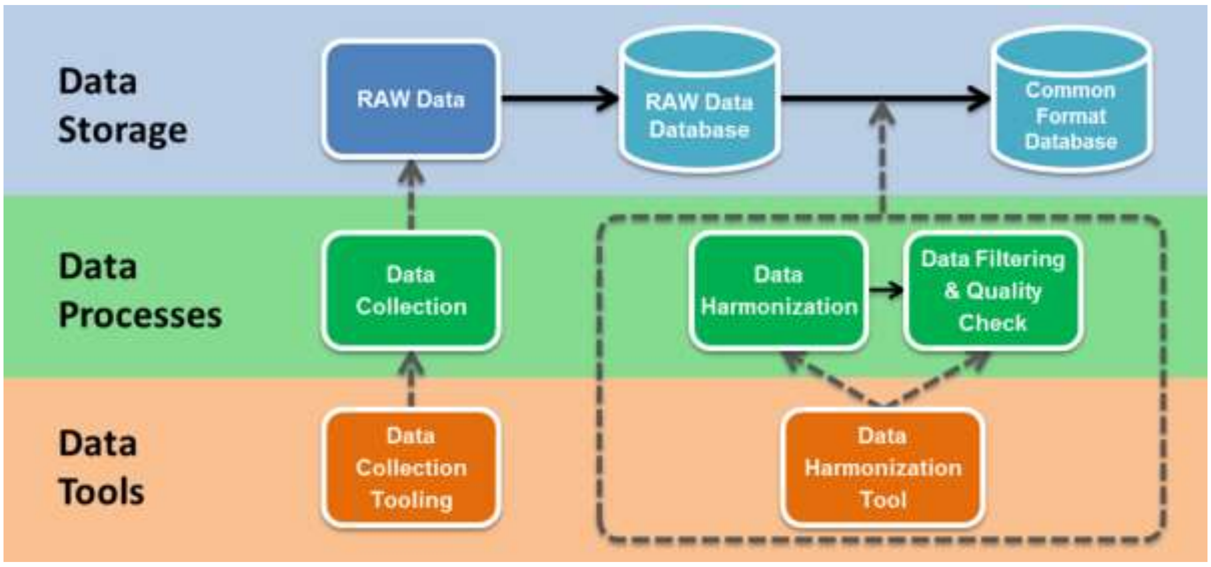


Figure 33: Vigo Data Processing components

7.3.2.1 Analysis
The data stored in the pilot site will be compliant with the OneM2M model for the IoT data and the ETSI ITS- G5 standard for V2X data.

7.3.2.2 Quality check
The following items will be checked according to project requirements:

- Naming convention
- Data resolution and precision
- Data frequency for each measure and message stored
- ID convention established for the pilot site
- Define and check data ranges according to previous definitions. Specially focused on GPS data

7.3.2.3 Processing and enrichment
No additional processing is going to be performed in the pilot site.

7.3.2.4 Metadata definition
The metadata will include the description of each type of test carried out and the description of the established IDs in the pilot site.

7.3.2.5 Analysis and quality check of the Parking Spot data messages
This section describes steps for parsing, filtering and verification of the data messages used by the parking spot service. We assume all messages are aligned with the DATEX II data format and are represented as JSON documents. The PSTS should check the format of an input message and its compatibility with the DATEX II data format. Parking table occupancy messages also contain links to the corresponding parking tables. The server should check all links to other documents and guarantee their consistency. All non-compliant data should be rejected by the server and not saved in the database.

Injection of a new parking table should initiate an update of the current geospatial index. This index keeps geographical locations of all existing parking spots and their groups. The index allows retrieving unique identifiers of parking tables by their geographical location.

A deletion of an existing parking table should force a deletion of all related documents – status messages for this parking table and its representation within the geospatial index.

7.3.3 Data storage

7.3.3.1 Raw data storage

Depending on the data sources the raw data will be collected in different ways. All the raw data is going to be sent to the central local server.

7.3.3.2 Processed data storage

The processed data will be stored in the central server according to the project requirements. This includes a file system with all the collected data accessible by the evaluators in the project.

8 Centralised Test Server architecture

8.1 Architecture design rationale

This subsection presents the rationale of the design, i.e. it describes and justifies the main decisions of design. The requirements and constraints define the expectations about the architecture.

8.1.1 Non-functional requirements

Requirements below are the most important non-functional requirements the CTS must comply with in order to ensure an optimal interoperability with pilot sites and integration in the AUTOPILOT project.

8.1.1.1 Design constraints

From an architectural standpoint, a constraint is an architectural design or implementation decision that has been selected to be treated as if it were a formal requirement.

Table 24: Centralised Test Server design constraints

ID	Title	Description	Source
NFRQ-DC-01	Data transfer	The CTS must provide a SFTP link to store the test data	D3.6 decision at PS level
NFRQ-DC-02	Data transfer	The CTS must provide a SFTP link to store the evaluation results data	D3.6 decision at PS level
NFRQ-DC-03	Data browsing	The CTS must provide an interface to search uploaded test and evaluation result data	D3.6 & D4.1
NFRQ-DC-04	Data storage	The CTS must provide a data storage space to store test data and evaluation results	D3.6
NFRQ-DC-05	Data upload monitoring	The CTS should provide a monitoring system that will enable the data uploading process	D3.6
NFRQ-DC-06	CTS REST API	The CTS must provide a REST API that gives access to Search, Upload and Download of test data and evaluation results.	D3.6

8.1.1.2 Architecture quality goals

This section presents two kinds of quality requirements:

- Performance and Scalability requirements
- Availability and Reliability requirements

Table 25: Centralised Test Server non-functional quality requirements

ID	Title	Description	Source
NFRQ-QR-01	Data validation	CTS must check and validate that uploaded test data complies with the predefined test description interface (test data xsd interface)	D3.6
NFRQ-QR-02	Data validation	CTS must check and validate that uploaded evaluation data complies with the predefined test description interface (evaluation result xsd interface)	D3.6

NFRQ-QR-03	Data availability	CTS must be available during the AUTOPILOT project execution from M18 to the end of M36	GA
NFRQ-QR-04	Data accessibility	CTS must provide access to all the evaluator partners	D3.6
NFRQ-QR-05	Data	CTS will receive the data (test data, evaluation result) in a zip or tar file that must be unzipped to retrieve the metadata information that will be used to search through the CTS web interface	D3.6

8.1.1.3 Other non-functional requirements

This section includes:

- Operational and Environmental requirements
- Security requirements
- Legal requirements

Table 26: Centralised Test Server non-functional requirements

ID	Title	Description	Source
NFRQ-NF-01	Data access	CTS must provide an authentication and authorisation access to stored data	D3.6
NFRQ-NF-02	Data processing	Test data will be stored without any transformation or conversion	D3.6, D3.1, D4.1
NFRQ-NF-03	Data processing	Before uploading data must be anonymised	D3.6
NFRQ-NF-04	Data processing	Evaluation result will be stored without any transformation or conversion	D3.6 & D4.1
NFRQ-NF-05	Data accessibility	Test Data and evaluation result will be provided through web interface	D3.6
NFRQ-NF-06	Data storage	Test data and evaluation result are stored locally in the CTS environment	D3.6
NFRQ-NF-07	Data storage	A PostgresSQL database is used to store the content of the description file associated to any test data or evaluation result	D3.6

8.1.2 Design rationale

The design decisions are guided by the use cases of the CTS:

- Uploading data from pilot sites
- Uploading data by evaluators
- Browsing, fetching, querying, searching among data
- Storage of various type of data
- Monitoring upload tasks
- Users and profiles management
- Downloading data manually (WEB API)
- Downloading data automatically (REST API)

8.2 Centralised Test Server architecture

This section presents the schematic system architecture, including interfaces of services and functionalities.

Figure 34 represents the deployment of the CTS, on four specific servers:

- Application server: hosts the front-end (web interface and REST API), and the back-end (CTS main application)
- File system server: stores the test data files
- Database server: stores the test data description
- SFTP server: receives test data uploaded by pilot sites

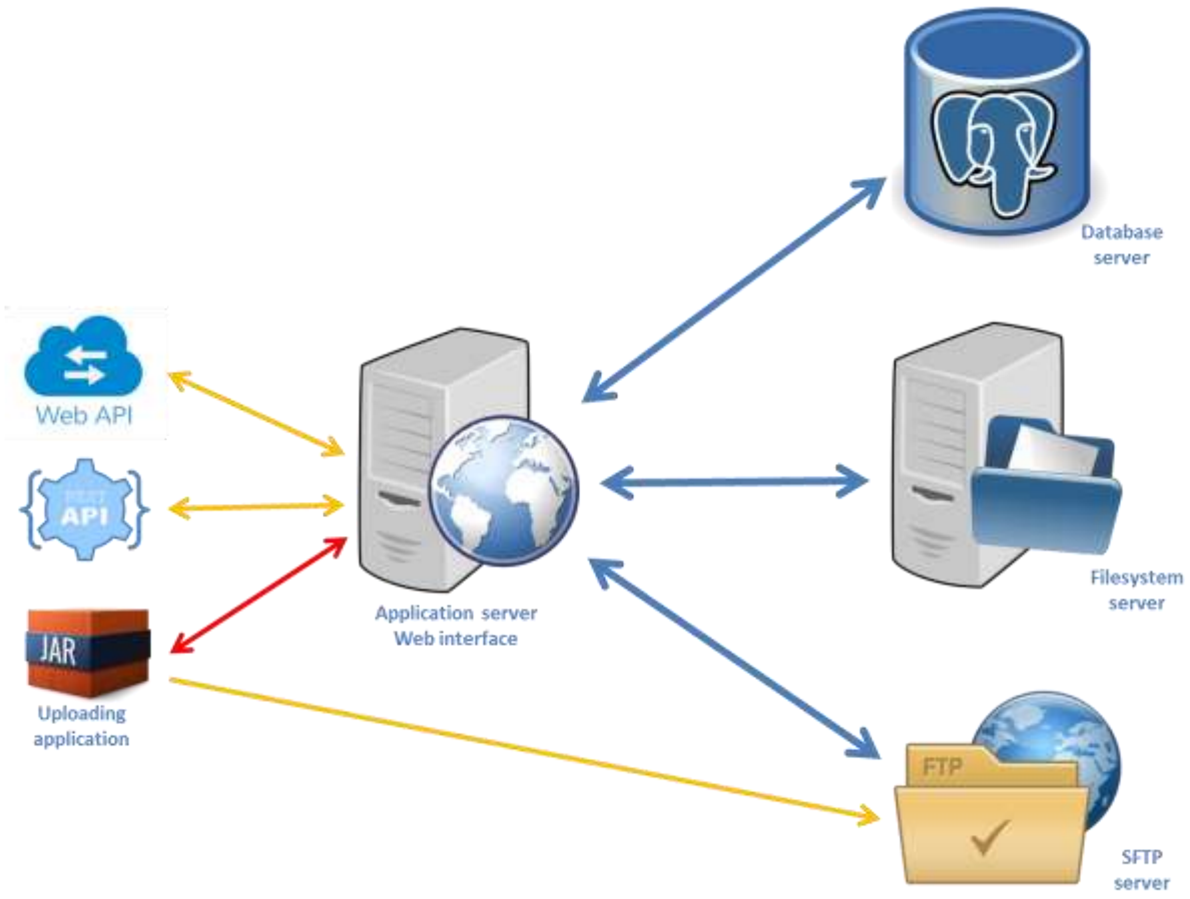


Figure 34: Centralised Test Server architecture

Web client and REST API are used to browse and download test data from CTS. An uploading application is used to upload test data or evaluation results. The red link between uploading application and application server represents the authentication process, the uploading task creation and the sftp information acquisition. Once sftp information is retrieved, the uploading application connects and uploads test data files to the sftp server.

8.2.1 Functional description

This section presents a functional view of the components of the application server.

Figure 35 shows the main Use Cases (logical software modules) of the application server of the CTS. Some modules can be triggered by an API, either from web interface, REST interface, or uploading application, other modules are internal.

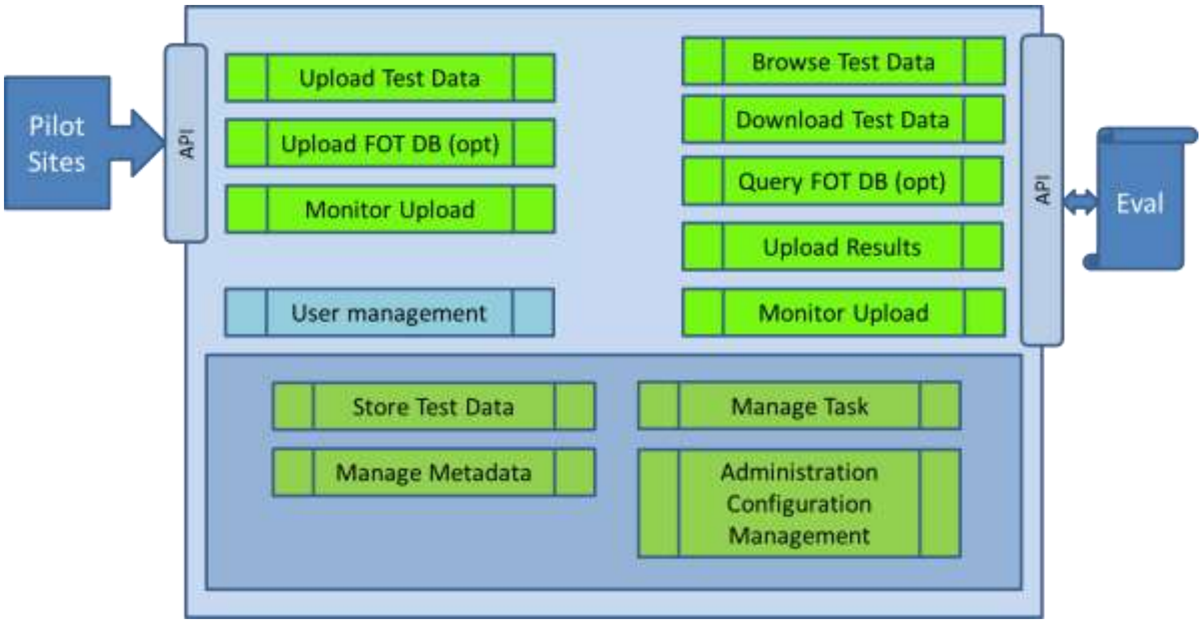


Figure 35: Centralised Test Server components

Here under is the layers view showing the data flow when uploading pilot site data to the CTS, when accessing or downloading data for evaluation.

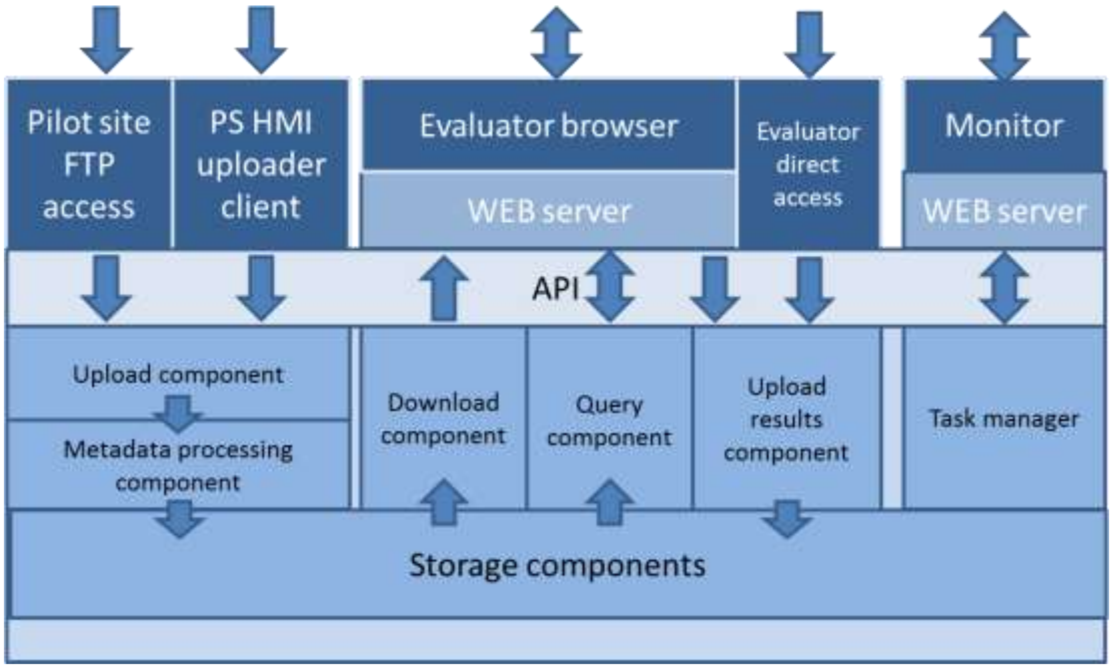


Figure 36: Components layers

As seen in the D3.6, the logic of the storage data flow is shown below. Received data archives are processed by the upload component in the application server. The first step is to analyse the description file of the archive. Its content is used to populate a database with uploaded tests and evaluation results descriptions.

Data are then stored as test data files in a dedicated filesystem.

FOT database creation is under study. This database could be populated using uploaded dumps.

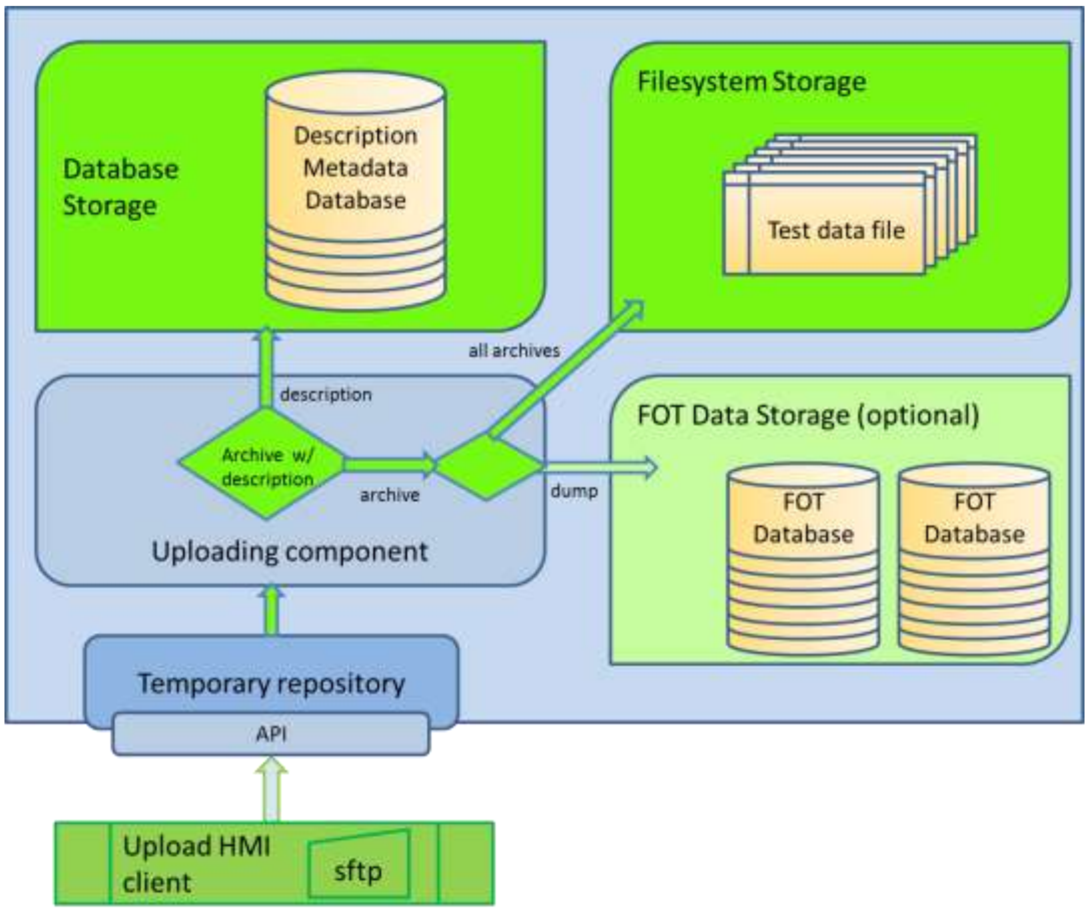


Figure 37: Schematic storage flow

8.2.2 HMI Interfaces

The following diagram summarises the interfaces with the CTS:

- Uploading data will be achieved using applications
- Accessing and managing data will be done using a web client

Figure 38 shows the applications used to upload test data and evaluation results, and details the main pages of the web client.

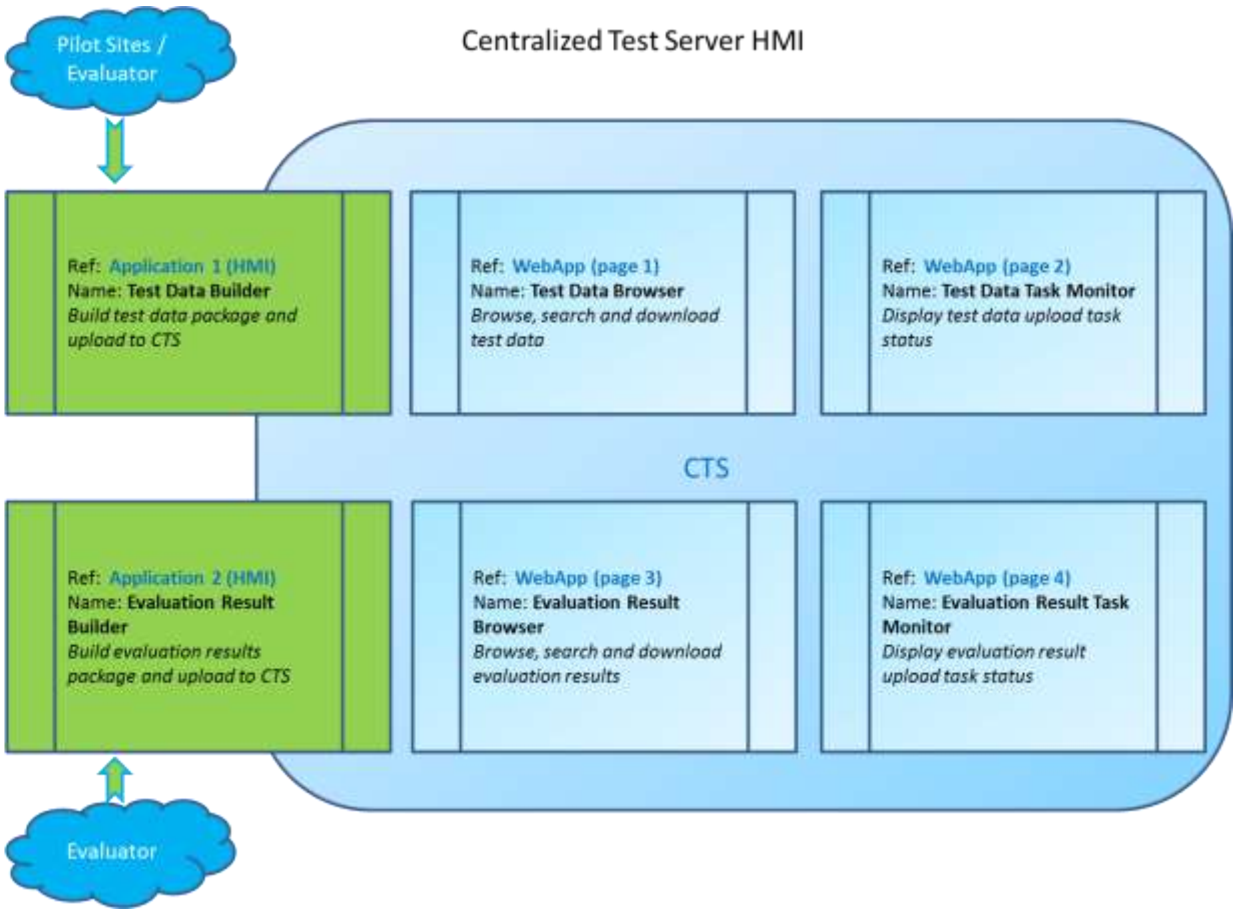

Centralized Test Server HMI
Pilot Sites / Evaluator
Ref: Application 1 (HMI)
Name: Test Data Builder
Build test data package and upload to CTS
Ref: WebApp (page 1)
Name: Test Data Browser
Browse, search and download test data
Ref: WebApp (page 2)
Name: Test Data Task Monitor
Display test data upload task status
CTS
Ref: Application 2 (HMI)
Name: Evaluation Result Builder
Build evaluation results package and upload to CTS
Ref: WebApp (page 3)
Name: Evaluation Result Browser
Browse, search and download evaluation results
Ref: WebApp (page 4)
Name: Evaluation Result Task Monitor
Display evaluation result upload task status
Evaluator

Figure 38: CTS HMI summary

8.2.2.1 Interfaces for data upload

The procedure for uploading test data and evaluation results to the CTS is to use two applications. Application 1 (Figure 38) will be provided to the pilot sites and the evaluators, and Application 2 (Figure 38) will be provided to the evaluators.

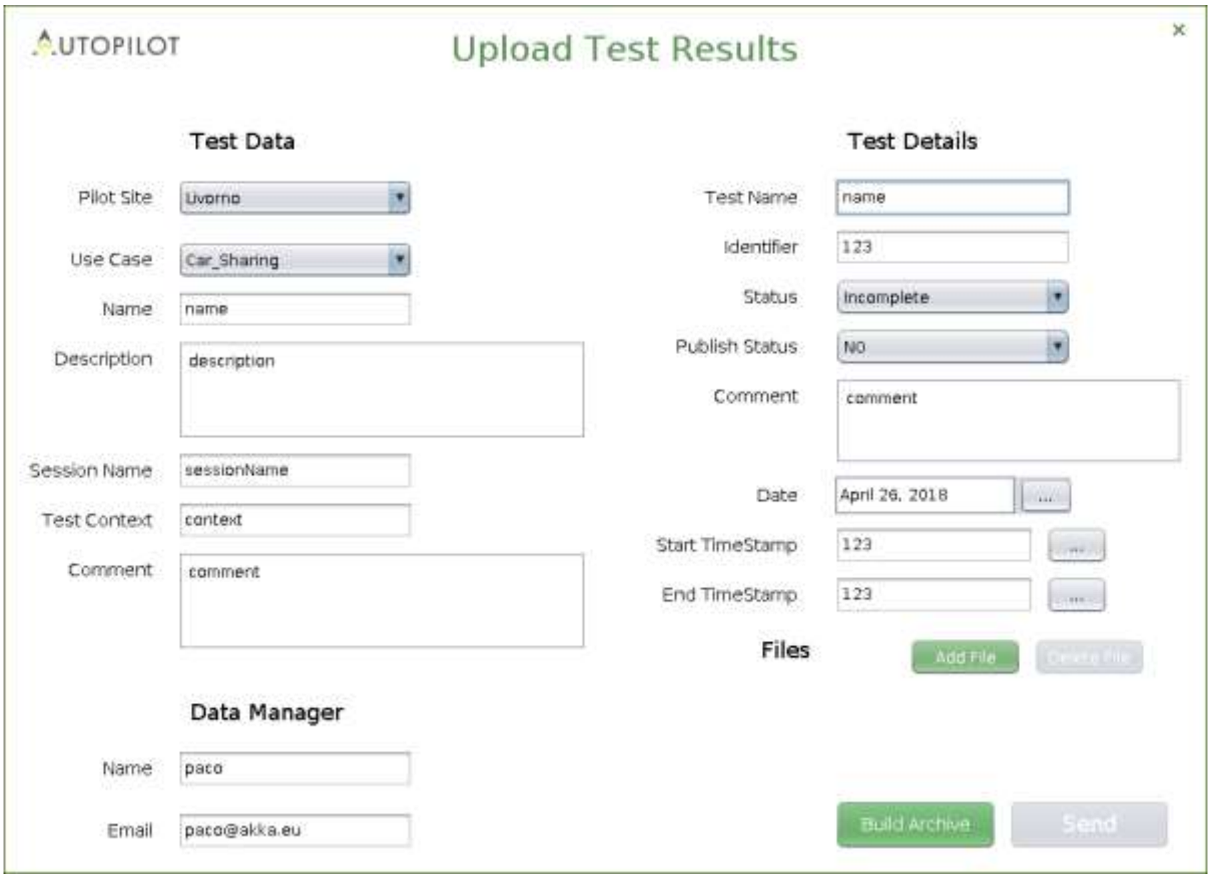
These applications allow selecting files, filling metadata, creating and storing archives and uploading them to the CTS. They also allow for uploading stored archives (previously created with the application) to the CTS.

These applications, as with the web client, require user authentication. Credentials will be provided to the responsible at each pilot site. Administrators will be able to create new users with specific access rights.

Note: evaluators will have the possibility to upload reworked and enriched test data (as new data), in order to share their work, produced by modifying downloaded test data, using application 1.

Figure 39 shows the main page of Application 1 (Test data builder). All fields are mandatory to create the test data description file. An “Add file” button opens a new page to select test data files by browsing the filesystem and also fill the description fields associated with each test data file.

When all the information has been collected and the test data files added, the archive can be created and stored or sent to the CTS.



Test Data

Pilot Site	Livorno
Use Case	Car_Sharing
Name	name
Description	description
Session Name	sessionName
Test Context	context
Comment	comment

Data Manager

Name	paco
Email	paco@akka.eu

Test Details

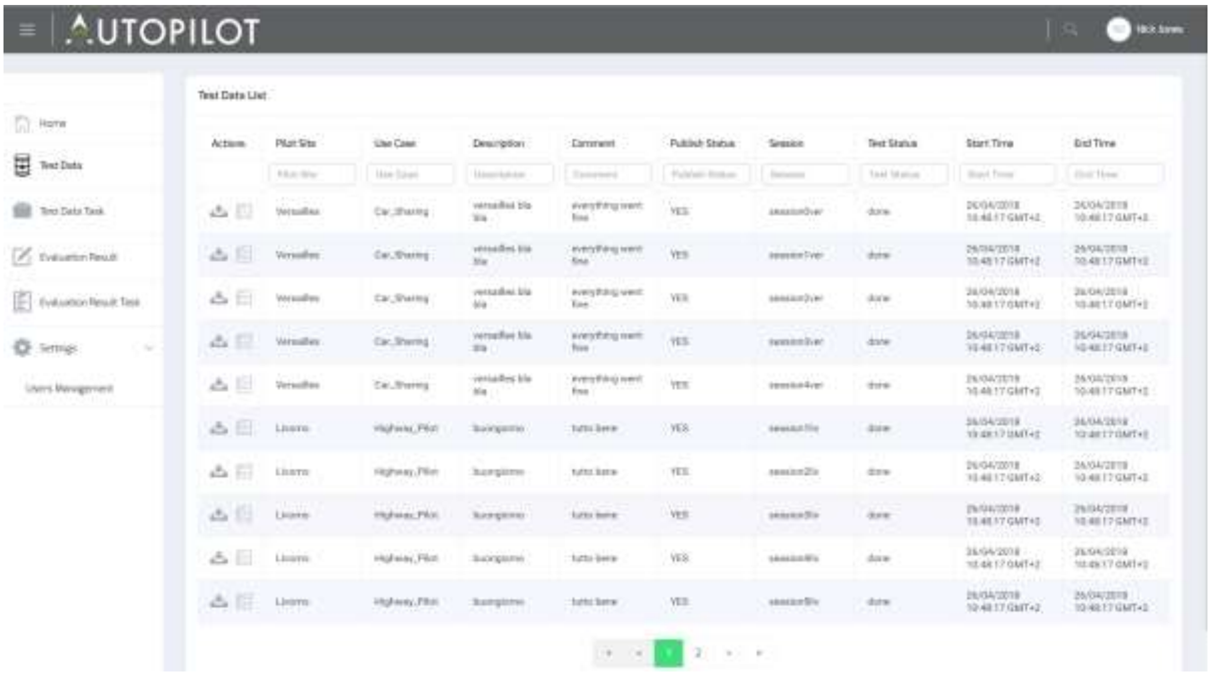
Test Name	name
Identifier	123
Status	Incomplete
Publish Status	NO
Comment	comment
Date	April 26, 2018
Start TimeStamp	123
End TimeStamp	123

Files

Figure 39: Test Data Builder HMI

8.2.2.2 Interfaces for data access

CTS collected data are accessible using a web browser. Figure 40 shows how it is displayed on the test data browser page (WebbApp page 1 in Figure 38).



Test Data List

Actions	Pilot Site	Use Case	Description	Comment	Publish Status	Session	Test Status	Start Time	End Time
	Versailles	Car_Sharing	versailles bla bla	everything went fine	YES	session0ver	done	26/04/2018 10:48:17 GMT+2	26/04/2018 10:48:17 GMT+2
	Versailles	Car_Sharing	versailles bla bla	everything went fine	YES	session1ver	done	26/04/2018 10:48:17 GMT+2	26/04/2018 10:48:17 GMT+2
	Versailles	Car_Sharing	versailles bla bla	everything went fine	YES	session2ver	done	26/04/2018 10:48:17 GMT+2	26/04/2018 10:48:17 GMT+2
	Versailles	Car_Sharing	versailles bla bla	everything went fine	YES	session3ver	done	26/04/2018 10:48:17 GMT+2	26/04/2018 10:48:17 GMT+2
	Versailles	Car_Sharing	versailles bla bla	everything went fine	YES	session4ver	done	26/04/2018 10:48:17 GMT+2	26/04/2018 10:48:17 GMT+2
	Livorno	Highway_Pilot	buongiorno	tutto bene	YES	session1liv	done	26/04/2018 10:48:17 GMT+2	26/04/2018 10:48:17 GMT+2
	Livorno	Highway_Pilot	buongiorno	tutto bene	YES	session2liv	done	26/04/2018 10:48:17 GMT+2	26/04/2018 10:48:17 GMT+2
	Livorno	Highway_Pilot	buongiorno	tutto bene	YES	session3liv	done	26/04/2018 10:48:17 GMT+2	26/04/2018 10:48:17 GMT+2
	Livorno	Highway_Pilot	buongiorno	tutto bene	YES	session4liv	done	26/04/2018 10:48:17 GMT+2	26/04/2018 10:48:17 GMT+2
	Livorno	Highway_Pilot	buongiorno	tutto bene	YES	session5liv	done	26/04/2018 10:48:17 GMT+2	26/04/2018 10:48:17 GMT+2

1 2

Figure 40: CTS Web application HMI

8.2.3 REST API

In addition to the CTS Web application which enables users to search and query available data, a RESTfull API makes it possible to give access to all the public resources available in the CTS. This interface is based on a REST API standard and is documented by and can be discovered through the swagger provided tools (swagger.io).

8.3 Centralised Test Server components architecture

This section presents the detailed architecture of the main components from section 8.2.

8.3.1 Data upload

This component provides the functionality to upload data and unzip the archives. Once files are readable, it will check the completeness of the description file and its consistency with the uploaded files. Then files are handed over to the metadata processing component.

8.3.2 Metadata processing

This component provides the functionality to process and store metadata information describing the upload currently in process. These metadata are stored in the description database.

8.3.3 Data storage

This component provides the functionality to create repositories for each upload, to store files in their dedicated repository, and to store the original uploaded archive in a backup repository. Information about the path to retrieve data (repository) will be added in the description database.

According to data type, it may optionally dump the test data into FOT databases.

8.3.4 Evaluator browser

This component offers an interface to the evaluator for browsing available test data, using various criteria (use case, pilot site, date, test point, various metadata fields), retrieve and download test data.

According to data storage policy, it may optionally enable the evaluator to execute queries on the data.

8.3.5 Download component

This component will build an archive according to an evaluator data request and make it available for download.

8.3.6 Query component

This component will run queries built in the evaluator browser.

8.3.7 Upload evaluation results

This component provides the functionality to upload evaluation results using the evaluation result upload application provided to the evaluators. The evaluation results are stored in the CTS database designed for this purpose.

8.3.8 Monitor, Task manager

This component will provide a way to monitor the upload progress and status, and to check the status of previous operations.

9 Conclusion

This deliverable describes the software architecture that will be put in place by each pilot site to implement the data collection process used for the management of the test data produced by each pilot site during the piloting activities.

A formal description of all the non-functional requirements and constraints is provided. Each pilot site has built its own architecture, according to the use cases it will run and the specificities of the partners involved in. Consequently, the workflows to collect and process data may differ, but the collected data must be provided in a common data format. A harmonisation effort was done in D3.6 to share common data formats and data models, but specificities remain and are implemented in each pilot site on a distributed platform called *"Pilot Site Test Server"*.

The AUTOPILOT project proposes a unique platform called a *"Centralised Test Server"* to upload, share, store and browse many sorts of data, allowing the evaluators to work with harmonised data and a convenient interface. All the test data will be uploaded into the centralised server using the same interface. This interface unifies the way the test data and evaluation data will be transferred and guarantees that all mandatory metadata will be provided in order to identify precisely any shared data created during piloting activities.

The current document will be used by the architects and developer teams to define the targeted distributed and centralised platforms. The CTS platform will be documented in the deliverable D3.8 *"Implementation of Test Data Management Platform"* due in M21.