Grant Agreement Number: 768953

Project acronym: ICT4CART

Project full title: ICT Infrastructure for Connected and Automated Road Transport

D7.7

Collected and Aggregated Data

Due delivery date: 31 December 2021

Actual delivery date: 14 January 2022

Deliverable type: OTH

Organisation name of lead participant for this deliverable: IBM-IE

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)	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 768953

Document Control Sheet

Deliverable number:	D7.7
Deliverable responsible:	IBM IE
Workpackage:	WP7 – Test Sites Integration and Data Collection
Editor:	Anton Dekusar, IBM IE <adekusar@ie.ibm.com>

Author(s) – in alphabetical order		
Name	**Organisation**	**E-mail**
Gottfried Allmer	ASFINAG	Gottfried.Allmer@asfinag.at
Michael Buchholz	UULM	michael.buchholz@uni-ulm.de
Martin Dirnwöber	ATE	Martin.Dirnwoeber@austriatech.at
Edoardo Bonetto	LINKS	Edoardo.bonetto@linksfoundation.com
Anton Dekusar	IBM-IE	adekusar@ie.ibm.com
Martin Herrmann	UULM	martin.herrmann@uni-ulm.de
Richard Lutz	NOKIA	richard.lutz@nokia.com
Alessandro Marchetto	CRF	Alessandro.marchetto@crf.it
Rodrigo Ordonez Hurtado	IBM-IE	Rodrigo.Ordonez.Hurtado@ibm.com
Anastasia Petrou	BMW	anastasia.petrou@bmw.de
Jan Strohbeck	UULM	jan.strohbeck@uni-ulm.de
Markus Wimmer	NOKIA	markus.wimmer@nokia.com

Document Revision History			
Version	**Date**	**Modifications Introduced**	
		Modification Reason	**Modified by**
V0.1	29.10.2021	Document structure	Anton Dekusar
V0.2	24.11.2021	Initial draft	Anton Dekusar
V0.3	10.12.2021	Added contributions from the partners	Anton Dekusar
V1.0	14.12.2021	Finalised Version	Anton Dekusar
V1.1	10.01.2022	An updated version based on a review by UULM	Anton Dekusar
V1.2	14.01.2022	Added contributions from the partners after the review	Anton Dekusar
V1.3	14.01.2022	Submitted version	Anton Dekusar

Abstract

This document describes the data collection and aggregation methodology used in ICT4CART, together with a summary of the state of the art on tools needed for data acquisition, validation, transmission, database structure and storage. The developed methodology is impacted by the other tasks of WP7, where the test site requirements and deployments are specified, as well as the task T8.1 "Evaluation Methodology and Plan". The test sites have provided their own data specifications of the collected data in each use case.

Legal Disclaimer

Abbreviations and Acronyms

Acronym	Definition
5G	5th Generation (Mobile Communication Network)
AD function	Autonomous driving function
ADAS	Advanced driver-assistance systems
ASCII	American Standard Code for Information Interchange
CAN	Controller Area Network
C-ITS	Cooperative Intelligent Transport Systems
CLI	Command line interface
CPM	Collective Perception Message
CSV	comma separated values
DOP	dilution of precision
EC	European Commission
ECU	Electronic control unit
ETRS	European Terrestrial Reference System
GA	Grant Agreement
GNSS	Global Navigation Satellite System
ICT	Information and communications technology
ITS	Intelligent Transport System
ITS-G5	European standard for vehicle communication based on IEEE 802.11p
ITS-S	ITS Station
JSON	JavaScript object notation
KPI	Key performance indicator
LTE	Long Term Evolution
MEC	Multi-Access Edge Computing
NMEA	National Marine Electronics Association
OBU	On-board unit
PO	Project officer
RSU	Road-Side Unit
RTK	Real-Time Kinematic
SCN	Scenario
SPATEM	Signal Phase and Timing Extended Message
UART	Universal Asynchronous Receiver Transmitter
UTC	Coordinated Universal Time
UTM	Universal Transverse Mercator
V2X	Vehicle-to-everything: communication technology
WP	Work Package

Table of Contents

List of Figures

List of Tables

Executive Summary

This document describes the data collection and aggregation methodology used in ICT4CART, together with a summary of the state of the art on tools needed for data acquisition, validation, transmission, database structure and storage. The developed methodology is impacted by the other tasks of WP7, where the test site requirements and deployments are specified, as well as the task T8.1 "Evaluation Methodology and Plan". The test sites have provided their own data specifications of the collected data in each use case.

The data collection and aggregation methodology is described in the Section 2, and defines a common process that should be used by all the test sites and for each use case to produce required data with the right quality for the evaluation tasks in WP8. This deliverable presents insights into collected data with focus on the WP8 requirements and specificities per test site. In addition, the document explains the data management methodology and procedures, and defines the features of the centralised data management in ICT4CART which enables the storage and sharing of collected data across the consortium partners.

The current deliverable is a public document and provides a complete picture of the data management methodology developed the task T7.7 "Data Collection and Aggregation" in ICT4CART. In this task, the involved partners have developed a common methodology for data acquisition and centralised data management, which is tailored to the needs of collection of semi-structured data formats that may significantly vary from a test site to a test site.

All collected data are stored in a centralised data management system. The design and architecture of such a system are explained in Section 5 of this document, which provides a deep understanding of the interfaces provided to the evaluators and to the test sites to collect data.

1 Introduction

1.1 Aim of the Project

The main goal of the ICT4CART project is to design, implement and test in real-life conditions a versatile ICT infrastructure that enables the transition towards higher levels of automation (up to L4) addressing existing gaps and working with specific key ICT elements, namely hybrid connectivity, data management, cyber-security, data privacy and accurate localisation. ICT4CART builds on high-value use cases (urban and highway), which are demonstrated and validated in real-life conditions at the test sites in Austria, Germany, Italy, and at Italy/Austria cross-border. Significant effort was also put on cross-border interoperability, setting up a separate test site at the Italian-Austrian test site.

1.2 Purpose of the Document

This document represents the outcome of the task T7.7 "Data Collection and Aggregation". The document presents a methodology and the tools for collecting and managing data at the test sites, as well as processing and integrating data needed for the evaluation purposes in WP8.

This deliverable is structured in four sections:

- Section 2, which describes the common data management methodology applied in the project.
- Section 3, which describes what type of raw data the test sites collect, and how they store and process it. This section provides an insight into details of data collection at the test sites.

- Section 4, which describes what types of raw data were post-processed by the corresponding test site, and what is available for evaluation purposes. This section provides a reference to the data that are used by other partners in the project if they intend to analyse it.
- Section 5, which describes the centralised data management system, the main goal of this task. In this section, a detailed architecture of the system is presented with a detailed description of the developed software tool to interact with the system, and the APIs exposed to the project partners.

1.3 Targeted Audience

This document is a public deliverable of the ICT4CART project. It is primarily targeted to the project partners involved in the data collection and aggregation at the test sites. Nevertheless, the document may also be of interest to any reader that is interested in large-scale collection and management of test data that are highly diverse and spatially-and-temporally varying. Therefore, this document also covers tools needed for data acquisition, transmission, database design data storage.

2 Data Management Methodology

In this section, the data management methodology applied in the ICT4CART project is presented. For the data collection purposes, the methodology is based on a two-stage approach. First, the proposed methodology needs to be compliant with the WP8 evaluation requirements. Since the required data for evaluation include various measurements and metrics, the data management methodology must be flexible enough to express potentially unseen requirements. On the other hand, it must satisfy the project goals. The data to be collected must handle all the test sites aspects: vehicles, infrastructures, IoT devices, IoT platforms, standards, structures, and test scenarios. In order to meet these requirements, the data structure definition should not be fully fixed and provide a great extent of flexibility to the partners which are responsible for data collection and to the partners which are involved in the evaluation tasks. Second, the data management methodology does not limit the project partners what type of data must be collected and how it must be processed locally. The project partners are responsible for taking into consideration the requirements of the data analysis in WP8 and providing these data through a centralised data management platform. Thus, an overall data management process has a distributed nature, where requirements are defined by the evaluation team, raw data collection performed by the project partners and, in overall, data are collected in the centralised data management system. This is illustrated in Figure 1.

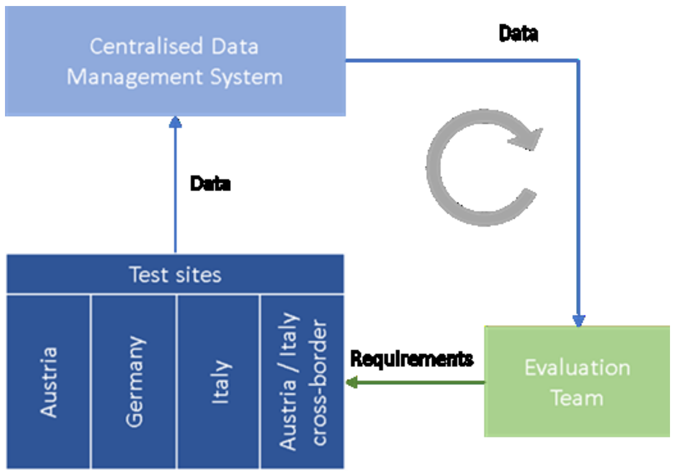


Figure 1. General data flow.

The data collection process consists of a sequence of steps that forms a loop, which can be iterated several times depending on the requirements and collected data. This process can be initiated by either a partner from the evaluation team or by a partner at a test site and is defined by the following steps.

- First off, the evaluation team defines requirements for evaluating KPIs and metrics. This was done in the task T8.1 "Evaluation Methodology and Plan".
- Next, the partners at the test sites investigate the requirements and prepare a corresponding infrastructure and services for data logging and post-processing. Here, any data collection tools can be used, and collected data may be present in any digital form.
- Once collected data at the test sites are post-processed and ready for submission, they should be uploaded to the centralised data management system to be available to the evaluation team and other project partners who are interested in data analysis. Here, a common data format and structure arise, which are described later in this section.
- Finally, available data in the centralised data management system can be used for evaluation purposes and for elaboration of more detailed requirements for data collection, thus closing the data management loop.

Considering that the described methodology does not cover the ways data should be collected and processed at the test sites, then data at the test sites can be in any form. The centralised data management system cannot allow data to be uploaded and stored in an arbitrary format. Since in the data analysis tabular data representation is widely used, the centralised data management system accepts data in tabular format, especially in the comma separated values (CSV) format. According to Wikipedia [Wiki-CSV], a CSV file is a delimited text file that uses a comma to separate values. Each line of the file is a data record, and each record consists of one or more fields separated by commas. So, each CVS file contains tabular data, where metadata are usually present on the first line of the file and describe all column names present in the file. Thus, this format is convenient for computer systems due to the simplicity of the structure and it is still readable by humans. The data collection and exchange must be done using CSV format at the centralised level.

Despite the versatility of the tabular data format, some limitations should be imposed on the collected data. In fact, the collected data are used for evaluation purposes, so the evaluators, at least, should be aware where data were collected, what the use case was, and some additional hints that may be helpful to distinguish different test runs. Therefore, a minimal set of columns must be present in any CSV data in the project to contain:

- **Test site.** This describes the test site the data are collected. There are four test sites in the project: in Germany, Austria, Italy and at the Italy/Austria border. For the test site description, please refer to the deliverable [D7.1].
- **Use case.** This describes the use case of the project. There are four use cases in the project: smart parking, dynamic adaptation, intersection crossing, and the cross-border use case. For the use case description, please refer to the deliverable [D2.1].
- **Evaluate.** This is a Boolean property that is either set to true or false. If it is set to true, then the data should be considered in the evaluation, otherwise the data are collected for informative purposes only and may be discarded in the evaluation.

The rest of the data types is either not declared or declared optional for the centralised data management system. Optional properties are:

- **Session name.** This is for the data collectors and evaluators convenience. This allows to distinguish several test sessions that may be run at a test site. For instance, a session may be a set of set tests that are run at a single day.

- **Run name.** This property is also for the data collectors and evaluators convenience. This is an additional layer in the collected data hierarchy and may describe a certain test that was held at a test site. For instance, at each session, there may several runs of the same test, so this property allows to distinguish such similar tests.

The test sites may collect various measurements, which may differ from the measurements collected at another test site. Despite these possible differences in the measurements, the centralised data management system must be able to process data with such a semi-defined data structure.

Concerning the data provisioning to the centralised data management system, this task defines interfaces that cover data sharing and provisioning. Indeed, all test data collected and pre-processed at the test site level must be made available to other partners via the centralised data management system. Thus, the system must provide interfaces to upload and download data. These interfaces are described in more details in the Section 5. Access to the system is restricted to the project partners.

Additionally, some test sites may collect confidential data. As a result, the centralised data management system does not include such data and the partners must provide results of the data analysis carried out on their own.

3 Raw Data

This section describes what types of data are collected at each test site in the ICT4CART project, with a specific focus on the subsequent evaluation of the collected data. In each subsection, the project partners provide insights into the raw data, such as data types, structures, and formats used at the test sites. They also provide analysis of the quality, comparability, and consistency of the collected data at the pilot tests if deemed necessary. The data described in this section do not need to be fully uploaded to the centralised data management system. There may be a situation when collected data are pre-processed at a test site before making it available to the project partners, e.g., data can be anonymised first.

3.1 Raw Data at the Austrian Test Site

AustriaTech carried out tests with its Mobile Lab test platform. For each test run, log files have been recorded as a BLF file, which is a proprietary log file format of Vector CANoe software tool used. Please refer to the deliverable [D7.5] for a description of the test site. The files contain:

- collected C-ITS messages (DENM, IVIM, CAM),
- GNSS information (time and location information),
- other variables for post processing: timestamp of the message, security context, and a sender station identifier.

The data are stored locally at AustriaTech. They are used for evaluation and test report generation, which are then shared with relevant partners. In addition, it is possible to convert these files to the pcap format (generated by the pcap API used by Wireshark for network package capturing, see [Wireshark] and [pcap] for details) containing the received C-ITS messages, which can be shared with partners if needed.

Furthermore, a CSV file is generated during the test drive. The file may also be generated in post-processing as well. The file contains for each DENM received the following information:

- ActionID (orgSID + seqNum),

- StationID,
- time of message reception,
- time of message generation,
- channel used for transmission,
- message size,
- position of message reception.

3.2 Raw Data at the Germany Test Site

The data in the German test site differ between the four scenarios, as described in the following.

3.2.1 SCN 1.1 – Smart Parking and IoT Services

The evaluation of the scenario SCN 1.1 “Smart Parking and IoT Services” use case is based on the Ride-Hailing Simulator. Focus here lies on determining if the ICT4CART specific infrastructure influences the overall profit for the fleet manager.

The simulator gathers various data every time a test run is completed. The data are is processed and used for the evaluation internally. These data are made available to the project partners, but since there is no contribution to the calculation of any KPI, the provided data are marked as for non-evaluation purposes.

The data gathered for each simulation run and each vehicle are:

- vehicle-id – unique identifier of the vehicle,
- revenue – total revenue produced by the vehicle,
- cost – total cost for the vehicle, including consumption, service, and park costs,
- idleTime – total time the vehicle has been in an idle state, that means without a customer,
- travelTime – total time the vehicle has been travelling/moving with or without a customer,
- travelTimeEmpty – total time the vehicle has been travelling/moving without a customer,
- mileage – total distance the vehicle has travelled with or without a customer,
- mileageEmpty – total distance the vehicle has travelled without a customer,
- parkTimeEmpty – total time the vehicle has been parked.

The above-mentioned data are processed to match in units. At the end, cost and revenue are measured in euros, time parameters in minutes and mileage parameters in kilometres.

3.2.2 SCN 3.1a – Virtual Mirror

For the scenario SCN 3.1a, one critical aspect is latency. When measuring latency, it is important to differentiate between the transport latency and the application latency. The transport latency is the latency incurred by the transport of the information from the generating ITS-S to the receiving vehicle ITS-S. Because this data transport occurs via hybrid communication, data are collected from communication via only LTE/5G, only ITS-G5, as well as via a sequence of LTE followed by ITS-G5 (data relay).

Application Latency means the latency incurred by the processing of the raw data by the ICT4CART system until arriving in the vehicle (including transport latency). This latency is much higher than only the transport latency, since, in SCN 3.1a, raw sensor data must be processed and fused in a centralised location (MEC server) before the fused result can finally be transmitted to the vehicle.

The raw data that are recorded for the KPI evaluation are:

- **For transport latency:** Upon receiving a message in the vehicle, the timestamp that the message was received at, the encoding timestamp that is included in the message header, and the mode of transport (e.g., LTE/5G, ITS-G5, LTE+ITS-G5). Further details about the recorded data can be found in [D8.1] under "TE-KPI1-End-to-End Transport Latency".
- **For application latency:** Upon receiving a message in the vehicle, the timestamp that the message was received at, the timestamp in the message that indicates the time of the last sensor measurement that was used in the fusion process, and the mode of transport (same as for the transport latency). Further details about the recorded data can be found in [D8.1] and [D8.2] under "TE-KPI2-End-to-End Application Latency".

Before uploading the raw data, the data are filtered to remove occurrences of network unavailability, because these might heavily skew the usage of the data for the KPI evaluation.

Note that any raw or processed data from vehicle sensors or infrastructure sensors, that might be recorded for development and/or internal evaluation purposes, is out of scope of this deliverable and thus not further discussed.

3.2.3 SCN 3.4 – Precise Positioning in Urban and Highway Location

The scenario SCN 3.4 "Precise Positioning in Urban and Highway Location" focuses on the precise positioning of vehicles. Details are described in deliverable D7.4. RTK corrections as a means to improve the precision of localisation was at the heart of the use case. There are two measurement cases: (i) in the static case, where the measurement is conducted at a fixed, non-moving reference point; and (ii) in the dynamic case, where the measurement is done in a vehicle during a test drive.

The position information is provided by the RTK receiver in a mix of NMEA-0183 format (ASCII) and the u-blox proprietary binary format "UBX". The UBX format is superior, as it provides more detailed information. The logged data were configured at the receiver in a per-message type. The data volume of logged data needed to be limited to avoid an overflow of the receiver's UART transmission capacity to the host computer. The Windows program "u-center" provided by the company u-blox was used to log the data.

The binary format of all interesting data elements was converted into separate ASCII files (per-message type). For the purpose of data collection, these files have been merged into a single CSV file. Otherwise, no post-processing on the logged data has been performed to allow future evaluation of the raw data. This relates to both the static and dynamic measurements. During the dynamic measurement, the data from the vehicle's ground truth receiver were additionally collected. The RTK receiver logs and the reference system locations were stored in separate files, as any combining will result in data losses.

The collected data are slightly different between static and dynamic measurements: during static measurements, some values are not of interest (e.g., speed, heading); during dynamic measurements, the number of activated message types was lower to limit the data output, as an increased measurement rate of 5Hz was required to get sufficient measurements results. In the static case, the measurement rate was 1Hz.

3.3 Raw Data at the Italian Test Site

In the Italian test site, LINKS developed a software module, hosted in the infrastructural components of the pilot (i.e., RSU and MEC server), for collecting raw data about: (i) context and the environment in which the vehicle prototype of the pilot is operating; and (ii) the performance of the ICT/communication technology adopted. Additionally, both LINKS and CRF developed software modules, hosted in the vehicle components of the pilot (i.e., vehicle V2X OBU and ADAS ECU), for collecting raw data about the vehicle dynamics and the driver behaviour. For additional details about the Italian test site architecture and complements, see Section 4 of the deliverable "D7.3 - Italian test site" [D7.3].

The following lists detail the set of raw data collected by the developed software modules in the RSU, MEC server, vehicle V2X OBU and ADAS ECU.

List of raw data collected in the RSU:

- For each C-ITS message sent, it is logged:
    - content of the message,
    - timestamp at which the information provided in the message is made available at the application layer,
    - timestamp at which the message is sent,
    - communication channel (i.e., 5G and/or ITS-G5) used to send the C-ITS message,
- time synchronisation of the RSU based on Network Time Protocol (NTP) using a stratum 3 NTP server (i.e., NTP server on the MEC server) as a reference,
- round-trip time of the ping command from the RSU to the MEC server.

List of raw data collected in the MEC server:

- For each C-ITS message sent, it is logged:
    - content of the message,
    - timestamp at which the information provided in the message is made available at the application layer,
    - timestamp at which the message is sent,
- time synchronisation of the MEC server based on NTP using a stratum 2 NTP server as a reference.

List of raw data collected in the vehicle V2X OBU:

- For each C-ITS message received, it is logged:
    - content of the message,
    - timestamp at which the message is received,
    - communication channel (i.e., 5G and/or ITS-G5) used to receive the C-ITS message,
    - timestamp at which the C-ITS message is written in the CAN,

    - timestamp at which the event identified in the message is relevant to the vehicle,
- time synchronisation of the OBU based on NTP using GNSS clock as a reference (i.e., synchronised with stratum 0),
- round-trip time of ping command from the OBU towards the MEC server,
- geographic position of the vehicle (i.e., latitude and longitude).

List of raw data collected in the ADAS ECU:

- (relative) timestamp of the test execution,
- geographic position of the vehicle in terms of latitude and longitude,
- communication channel (i.e., 5G and/or ITS-G5) used to receive the C-ITS message,
- type of the received C-ITS message,
- distance of the event object of the C-ITS message,
- value of the longitudinal vehicle acceleration (m/s^2),
- pressure/activation of the braking pedal (e.g., not pressed, pressed),
- value of the acceleration (gas) pedal position,
- status of the lateral AD function actuation (e.g., not available, available, enabled, activated, level of the actuation),
- status of the longitudinal AD function actuation (e.g., not available, available, enabled, activated, level of the actuation),
- (relative) timestamp at which the warning message is elaborated by the ADAS ECU and provided to the driver,
- value of the vehicle speed (m/s).

The listed raw data are processed to produce log files that contain the information to be logged for the different KPIs as described from Section 3 to Section 9 in deliverable D.8.2 “Technical evaluation – Version 1” [D8.2].

3.4 Raw Data at the Cross-border Test Site

As in the Italian test site, LINKS and CRF developed software modules, hosted in the vehicle components of the pilot (i.e., vehicle V2X OBU and ADAS ECU), to collect raw data about the vehicle dynamics and behaviour. For additional details about the Italian test site architecture and complements, see the deliverable “D7.3 - Italian test site” of the WP7 (Section 4) [D7.2].

Raw data collected in the vehicle V2X OBU at the cross-border test site are the same as at the Italian test site. In addition to them, connection requests performed from the OBU to AMQP message brokers are logged together with the message brokers’ details for identifying the message brokers.

Raw data collected in the ADAS ECU are the same as at the Italian test site. The generated raw data are processed in the same fashion as at the Italian test site.

AustriaTech used the same Mobile Lab configuration for tests at the cross-border site as for the Austrian test site. Therefore, the structure of the collected raw data is the same as described in Section 3.1.

4 Data Collection

4.1 Categories of Data Generated

This section describes what kind of data may be collected at the test site level. **REMARK:** This section is provided as a reference and copied from [D1.3].

In the ICT4CART project, the following three main categories of data are produced and handled:

- **Context Data**: Data related to the environment in which a vehicle is operating.
- **Vehicle Data**: Data collected by in-vehicles sensors and devices.
- **ICT Data**: Data collected regarding the performance of ICT technologies.
- **Aggregated Data**: Summarised data obtained by analysing or merging any of the above data categories, to infer new knowledge.

4.1.1 Context Data

Context data are related to the environment in which vehicles operate. These include:

- **Static Data:** Data that rarely change in time, such as maps, road network data, parking spot locations and accessibility information, fuelling/charging stations, information about and positions of deployed infrastructure and sensors.
- **IoT Data:** Real-time and near-real-time data produced by (infrastructure) sensors and devices deployed outside of the vehicles and made available to the project. Examples of such sensors and devices include, but are not limited to, road-side cameras, traffic lights, parking spot detectors, etc. IoT data are typically exchanged through IoT platforms.
- **Public Authority Data:** Data and information from public authorities, such a city councils and transport control centres. These include but are not limited to traffic status, planned and real-time traffic events and road works. Public authority data may be generated by IoT devices, but generally from IoT devices typically beyond the reach **of the project partners.**

4.1.2 Vehicle Data

Vehicle data are those collected by vehicles either using their original in-car sensors or sensors added for the ICT4CART project purposes. Vehicle data include the following sub-categories.

- **Vehicle Dynamics:** Measurements that describe the mobility of the vehicle, such as longitudinal speed, longitudinal and lateral acceleration, yaw rate, and slip angle.
- **Driver Actions:** Measurable driver actions on the vehicle commands such as steering wheel angle, pedal activation or HMI button press variables.
- **In-vehicle Systems State:** Accessed by connecting to the embedded controllers. It includes continuous measurements like engine RPM or categorical values such as ADAS and active safety systems activation.
- **Environment Detection/Perception:** Environment data that can be obtained by advanced sensors like RADARs, LIDARs, cameras and computer vision, or simpler optical sensors. For instance, luminosity or presence of rain, but also characteristics and dynamics of the infrastructure (lane width, road curvature) and surrounding objects (type, relative distances

and speeds) measured from within a vehicle, e.g., via dedicated short-range communication (DSRC) devices.

- **Vehicle Positioning:** The geographical location of a vehicle as determined by, among others: satellite navigation systems (e.g., GPS, DGPS, A-GPS), dead reckoning, map matching, and cooperative positioning (e.g., multilateration).
- **Media:** Mostly images and videos, but also index files used to synchronise the other data categories.

Some of the vehicle data may be shared with other vehicles or with the infrastructure or IT platform, while others may not.

4.1.3 ICT Data

ICT data are the result of analysing the performance of the communication technologies and devices used in the project. Examples of ICT data include (but are not limited to):

- Connectivity: the available connectivity options for a device at any given time,
- Bandwidth consumption: the bandwidth consumption of a device/service,
- Latency: the latency experienced by a service at any given time,
- Computation time: time needed to perform a given computation or provide a given service,
- Energy consumption.

4.1.4 Aggregated Data

Aggregated data are the results of analytical, statistical, or machine learning methods applied to any of the above data to extract knowledge. Examples of aggregated data include, but are not limited to, the following.

- Traffic jam locations and intensities derived from vehicle speed data,
- Statistics on parking space occupancy over time and space,
- Risk maps derived from historical driving data and environment conditions.

Aggregated data typically have the benefit of higher anonymity level compared to the source/raw data. Therefore, they tend to be less prone to privacy issues and easier to share and reuse.

4.2 Data Collection at the Austrian Test Site

Together with ASFINAG, AustriaTech measures with its Mobile Lab test platform the following KPIs as described in detail in [D8.2]:

- End-to-End Application Latency (TE-KPI2)
- Information Reliability (TE-KPI3b)

The value for latency is directly calculated during the test run and saved in the CSV file described in Section 3.1, while information reliability is obtained by evaluation of the logfiles.

4.3 Data Collection at the German Test Site

4.3.1 SCN 1.1 – Smart Parking and IoT Services

The data described in 3.2.1 were gathered for the SCN 1.1. For each test run, the simulator has been run twice, once with the parking API on and once with it off. In both cases, the same types of data were collected. In total there were 20 test runs and 40 simulator runs with different configurations of input parameters. The data have been then processed and evaluated internally. For more details, see [D7.4] Section 2.2 Evaluation of SCN 1.1.

4.3.2 SCN 3.1a – Virtual Mirror

For SCN 3.1a, the raw latency data gathered are uploaded to the centralised data management system, allowing it to be accessed by all partners. It is used by UULM to calculate the KPI metrics for [D8.3], namely TE-KPI1-End-to-End Transport Latency and TE-KPI2-End-to-End Application Latency.

The transport latency data were gathered for four cases (see [D8.2]):

- SPATEM data, communicated via ITS-G5 directly from an RSU to the vehicle.
- SPATEM data, relayed to the MEC server via LTE, then received by the vehicle via LTE/5G.
- CPM data, calculated on the MEC server, communicated to the vehicle via LTE/5G.
- CPM data, calculated on the MEC server, communicated to an RSU via LTE/5G, which relays the data via ITS-G5 to the vehicle.

The above allows for a computation of TE-KPI1-End-to-End Transport Latency for each recorded communication path, and, hence, for a comparison of the latencies when using hybrid communication.

For the application latency, data were gathered in two variants:

- CPM data, calculated on the MEC server, communicated to the vehicle via LTE/5G.
- CPM data, calculated on the MEC server, communicated to an RSU via LTE/5G, which relays the data via ITS-G5 to the vehicle.

However, since the application latency is largely determined by the data processing on the sensor side and the fusion on the MEC server, major differences between the two variants were not seen, so that either communication path is feasible for the vehicle, when hybrid communication is available (see [D8.2]).

4.3.3 SCN 3.4 – Precise Positioning in Urban and Highway Location

The data were uploaded to the centralised data management system for both the static and dynamic measurement cases. For KPI determination (TE-KPI4- Position Accuracy), only the static case was used.

The static measurement has been taken against the geodetic reference point in the city of Neu-Ulm, whose location information is not contained in the stored logs. Information about the geodetic reference point in Ulm can be found at [Geo-Ulm]. The following location information is provided for the Neu-Ulm geodetic reference point:

Latitude 48° 23.62275' = 48° 23' 37.3649'' = 48.39371246° N
Longitude: 10° 0.17186' = 10° 0' 10.3118'' = 10.00286439° E
UTM32: 32574237.75 E / 5360547.06 N
(all ETRS89)

For the static measurement, the following data were collected:

Date Type	Data Format
Date/Time	UTC format
Longitude/Latitude	decimal degree
Height (above ellipsoid and mean sea level)	m
Estimated horizontal / vertical accuracy	m
DOP values (Horizontal, Positional)	-
Number of satellites used in sum (and GPS, GLONASS, Galileo, Beidou in detail)	# (# per GNSS type)
Type of Fix as defined in NMEA:	1 - GPS fix, 2 - Differential GPS fix, 3 - PPS fix, 4 - Real Time Kinematic, 5 - Float RTK, …
Age of correction data	Seconds

Table 1: Static measurement data elements.

Longitude and latitude are sufficient in the static case to determine the KPI position accuracy, and any other data are available for additional evaluation purposes.

In case of the dynamic measurement, the following data were collected by the reference system:

Date Type	Data Format
Unix timestamp	value
Adma Accuracy Status	8 (= highest accuracy – can be taken as ground truth), 4, 2, 1
Latitude/Longitude	decimal degrees
UTM East/North coordinate	m
Orientation	radians
Velocity	m/s
Acceleration	m/s^2

Table 2: Dynamic measurement data elements - reference system.

The collected data of the dynamic measurement contain both tracks as outlined in [D7.4] at 50Hz measurement rate.

The RTK data were collected during the dynamic measurement case, which contained following data elements:

Date Type	Data Format
Date/Time	UTC format + time fraction in seconds
Longitude/Latitude	decimal degree
Height (above ellipsoid and mean sea level)	m
Estimated horizontal / vertical accuracy	m
DOP values (geometric, position, time, vertical, horizontal, north, east)	-
Number of satellites used in sum	#

(and GPS, GLONASS, Galileo, Beidou in detail)	(# per GNSS type)
Speed (2D)	m/s
Heading of motion	degree
Speed accuracy	m/s
Heading accuracy	degree
Baseline length	m
Type of Fix (NMEA format)	1 - GPS fix, 2 - Differential GPS fix, 3 - PPS fix, 4 - Real Time Kinematic, 5 - Float RTK, …

Table 3: Dynamic measurement data elements - RTK receiver.

For the two driven tracks, data were taken with a 5Hz rate and provided in separate files.

Evaluation of the position difference requires some pre-proecssing steps to align and interpolate the measurements from both receivers as if they have been taken at exact the same instant of time. It needs to be kept in mind that the reference system provided the position of the vehicle reference point, while the RTK receiver received its position from its own mounted GNSS antenna. While the arrangement of both points is of course fix, the difference in positions depends on the orientation of the vehicle. For the evaluation, only the deviation of the distance of both positions was measured by Nokia.

The provided CSV files contain only the raw data from the two receivers of the dynamic measurement case, not any derived information.

4.4 Data Collection at the Italian Test Site

LINKS and CRF collect data to evaluate the TE-KPIs described in deliverable D.8.2 "Technical evaluation – Version 1" [D8.2]. In particular, the collected data are used to evaluate following KPIs:

- End-to-End Transport Latency (TE-KPI1),
- End-to-End Application Latency (TE-KPI2),
- Communication Reliability (TE-KPI3a),
- Position Accuracy (TE-KPI4),
- Takeover/Vehicle level handover time gained (TE-KPI7), mainly in the motorway use-cases,
- Driver comfort (KPI-9).

For each KPI, the data collected correspond to the list of information to be logged as described in deliverable [D8.2]. Please refer to Section 2 of [D8.2] for details about the information logged.

4.5 Data Collection at the Cross-border Test Site

Data collection here is performed similarly as at the Italian test site (please refer to the previous section). The project achieved a common adherence to the C-ROADS 1.8 standard for data transmission, so that project vehicles could collect data on both sides of the Austrian-Italian border without having to alter anything in the on-board receivers (refer to [D7.6]).

5 Centralised Data Management

This section describes how data are collected and aggregated in the centralised data management system.

5.1 General Description

The centralised data management system is a set of services which aim to collect and store data from the test sites and allow the evaluators to access and download data. All the data stored in the system have previously been anonymised by the data providers. The data management system offers an API and a command line tool for test sites to upload their data. The API can be used directly in the applications and services developed in the project. In this is case, there is no need for manual data upload, but such a solution requires more efforts in the development of the applications. This API is discussed in more details in this section. Collected data can be also uploaded manually via a command line tool. Users can follow the uploading progress in the tool and see the final upload status of the data. The same command line tool can be used for downloading data and is intended to be used by the partners who are involved in the evaluation tasks in the project in WP8. This tool is available to all project partners to manage their collected data.

A direct access to the databases can be granted to the evaluators' convenience, in order to make it possible to query any relevant data for evaluation purposes.

5.2 Centralised Data Management System

In the Figure 2, an overall design of the centralised data management system and corresponding client tools is shown.

The centralised data management system interacts with the data collectors from the test sites and the project evaluators:

- The system collects, sorts, and stores received collected data from the collectors at the test sites.
- Evaluators can access and download data in order to analyse and evaluate the project.

Therefore, the goal of the system is to provide an interface to upload, aggregate, store and finally download data.

General requirements of the system are:

- Data must be described using additional information called metadata. These metadata are split into two parts: one is the required metadata, like a test site name and a use-case, and the second part that consists of a few optional properties. The rest of the data must be described in the CSV format. Thus, every column in the CSV format should properly labels.
- The system must verify that all required properties are set, process optional properties if present and verify that data in the CSV format are correct and can be parsed, uploaded, and stored in the database.
- The system must be used to collect required data from the test sites for the evaluation purposes.
- Before uploading any data to the system, data must be anonymised by the partners who collect data.

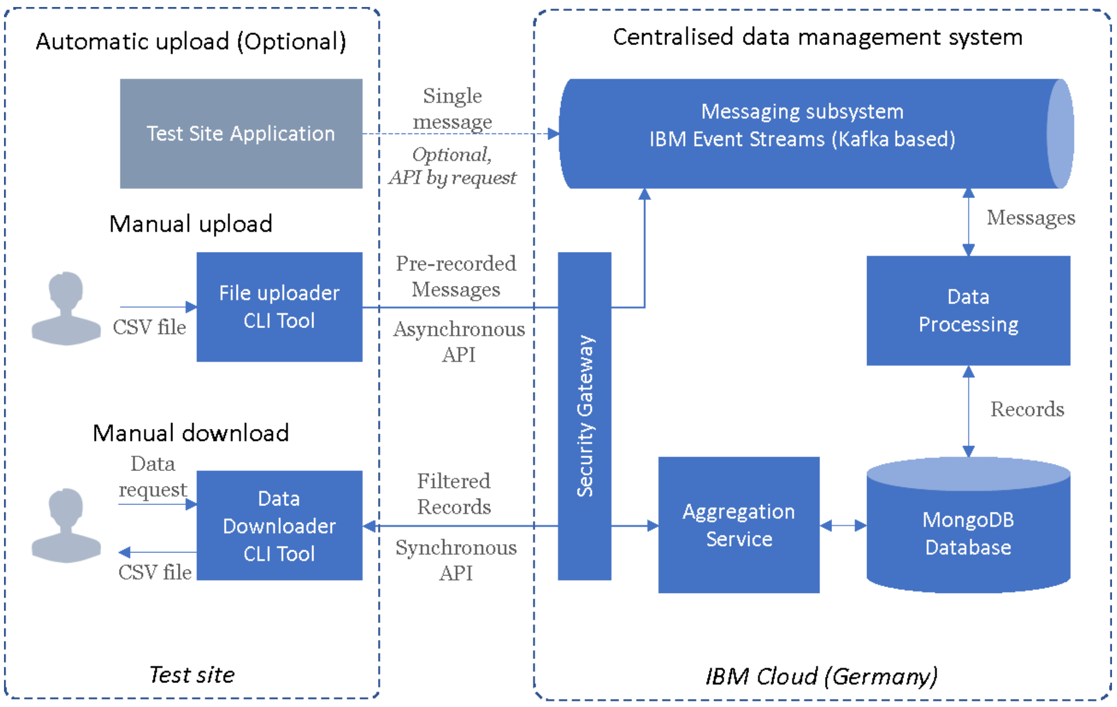


Figure 2. Architecture of the centralised data management system.

5.2.1 Data Flow

A general data flow in the system is shown in the Figure 3. The flow begins when any tests are run at a test site and corresponding data are generated in a test site application. Then, depending on the integration architecture of this application, the generated data can be either uploaded automatically from the application to the centralised data management system via a messaging subsystem or the generated data can be saved to a CSV file for post-processing and manual upload. In the first case, once an application generates a measurement or a set of measurements, they can be sent to the centralised data management system via the messaging subsystem. Thus, every message is a measured state of the application recorded at a certain time. A stream of such messages becomes a tabular dataset from which the evaluators may analyse and compute KPIs.

Certainly, the application may produce a file in a custom format and then it can be post-processed and converted to a CSV file, but for a sake of simplicity, it is assumed that the system is able to directly produce CVS files suitable for upload to the centralised data management system. Once a CSV file is obtained, it can be uploaded by a data collector at the test site. Once the data are available, a project evaluator may download the data based on their requirements, e.g., download all data from a concrete test site or data generated in a concrete use-case across all test sites.

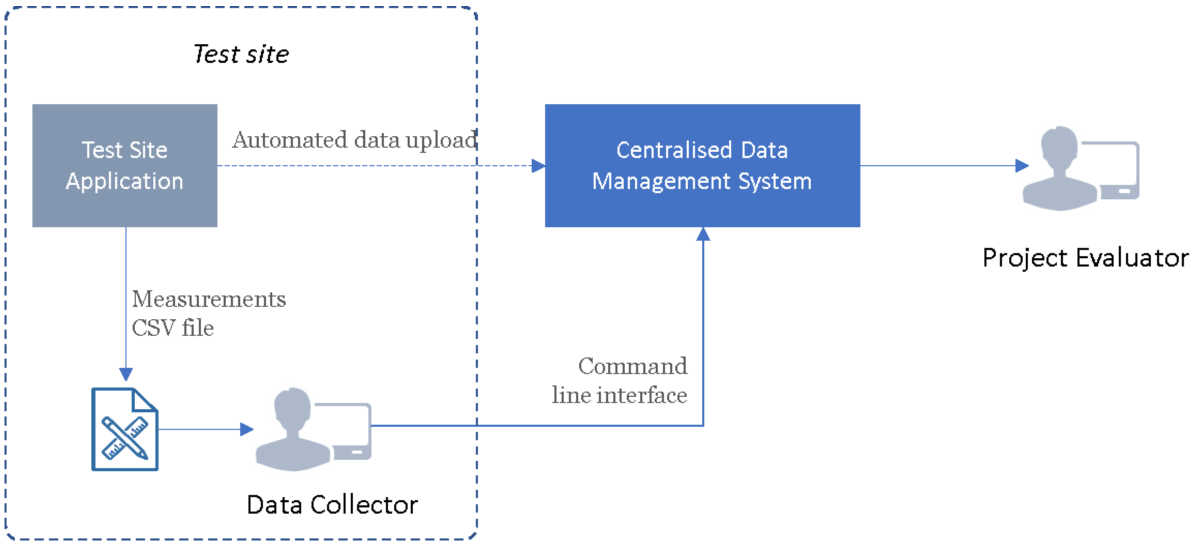


Figure 3. Data Flow for the Data Management System.

5.2.2 Functional Description

In this section, the main components and services of the centralised data management system, their role in the overall design, and properties are described following the components shown in Figure 2.

Messaging subsystem

In general, a messaging system is responsible for transferring data from one application to another in a transparent way for both applications so that applications deal with data and do not have to take care about actual algorithms used on the transmission layer. This simplifies the design of such applications and decouples the components of applications in a modular fashion.

In the centralised data management system, the goal of the messaging subsystem is to collect data record by record. Each record is one message or, in other words, one set of measurements at a given time. Messaging is very convenient for processing data in an asynchronous fashion, when a very small overhead is required or acceptable to emit a message and forget it. A very common asynchronous pattern is a fire-and-forget pattern. In this pattern, a message is sent from one application to another application via a messaging infrastructure with no attempt to track the results or response. Often with the asynchronous pattern, assured delivery of messages is required, which means that no loss of messages and no duplication of messages occur. These additional requirements bring additional overhead, but even in this case, performance of the messaging system is high.

The messaging interface is suitable for the integration into a system under evaluation. In this case, the system under evaluation generates messages when new measurements are recorded. Usually, messaging systems are very fast and thousands of messages per second maybe produced and emitted by a single system without a significant impact on overall performance of the system.

In the ICT4CART project, an IBM Event Steams [Streams] has been chosen for the messaging system/infrastructure. This is a cloud-based offering that has Apache Kafka [Kafka] as a backbone. Apache Kafka is a framework implementation of a software messaging bus using stream-processing. It is an open-source software platform developed by the Apache Software Foundation. The Apache Kafka project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. The goals of the Apache project perfectly fit to the needs of the ICT4CART project.

Any Kafka compatible client can connect to the messaging subsystem and produce message that will be processed and stored in the database. This API is provided by IBM-IE upon a partner’s request. The details of this API are provided later in this document.

The messaging subsystem is deployed on the IBM Cloud in the data centre in Germany as a separate service.

Data processing

Data processing is a separate service that receives messages from the messaging subsystem and stores them directly in the database. This service applies basic verifications of the received messages and does not provide an API.

The service is also deployed on the IBM Cloud in the data centre in Germany.

Database

This service provides the functionality to store all received messages uploaded to the system via the messaging subsystem and processed in the data processing service. Since the format of the messages may be flexible and only a few predefined properties are required, a chosen database should fulfil this requirement. Despite the data to be uploaded are in a tabular format, the structure of the format may vary across data and that is why classical relation databases may be a bad choice in this case, while NoSQL databases naturally fit into this requirement. Usually, NoSQL databases provide a mechanism for storage and retrieval of data that are modelled in means other than the tabular relations used in relational databases. They also may support flexible schemas for building modern applications. One of the popular NoSQL databases is MongoDB [Mongo], which has been chosen in the ICT4CART project. MongoDB is a document-oriented database.

Instead of using tables and rows for data storage as traditional relational databases do, MongoDB replaces them with collections and documents. A document consists of key-value pairs, where keys and values can be of any supported type. A document is the minimal amount of data the database can store. Documents can be considered as a direct analogy of rows in the relation databases. A set of documents forms a collection, which is an equivalent of a relational database table.

In the project setup, all messages uploaded to the system are stored in the single collection. This has a few benefits:

- The data are automatically aggregated. In this case, there is no need to store data from the test sites separately and then aggregate them in a manual process.
- Since the data are in one collection, they can be easily queried across all test site, all use-cases, etc.
- One collection is easier to maintain and support.

Summarising, the use of a NoSQL databases solves different issues and brings the required flexibility in terms of semi-defined data structure.

An instance of MongoDB is deployed on the IBM Cloud in the data centre in Germany.

Aggregation service

The aggregation service is a dedicated service that allows to query data in the database, aggregate data into the database, and download the results of the query. This service is a backend application that provides an API for the command line tool that is described later in the document. Thus, through the API, the evaluators can download documents stored in the database. Please note that this API operates documents that are stored in the database.

The service is also deployed on the IBM Cloud in the data centre in Germany.

Security gateway

The security gateway is a service that has the single goal: to verify credentials of a user and grant access either to upload data or download data. A user must provide a login and password to get access to the services provided by the centralised data management system. User credentials are provided upon request to the project partners by IBM-IE.

The service is deployed on the IBM Cloud in the data centre in Germany together with the aggregation service.

5.3 Command Line Interface

The command line interface (CLI) can be understood as a tool that operates in the operating system command prompt, where a user may type commands and interact with the tool. In the ICT4CART project, CLI is a tool designed for the data collectors and the project evaluators, and it provides access to the collected data stored in the centralised data management system. The tool is written in the Java language to be able to run the tool by users on different operating system without being rewritten. Hence, a Java Runtime Environment is required to be installed on the users' workstations. The minimal version that is required by the CLI is Java 11.

The CLI is packaged as a zip archive and available on the project website. Access to the CLI is restricted to the ICT4CART project partners only. There are a few files packaged in the zip archive.

Required files to run the tool:

- `data-collection-cli.jar` – the tool itself, packaged as a java application,
- `application.yml` – configuration file, must be edited before the first use,

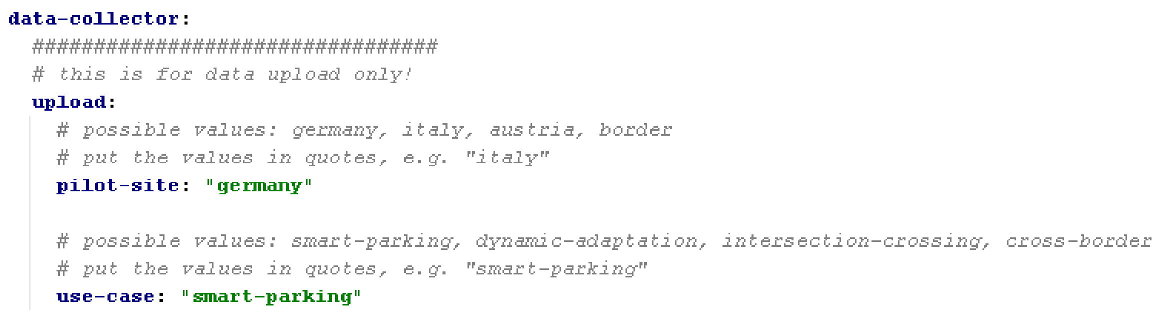
Optional files for convenience and testing:

- `generated-10.csv` – a sample CSV file with dummy data, *PROVIDED FOR EDUCATIONAL PURPOSES ONLY,*
- `upload.bat` / `upload.sh` – a sample script to upload the sample file,
- `download.bat` / `download.sh` – a sample script to download some data.

5.3.1 Configuration

Before a user can run the tool, the configuration file `application.yml` must be set up. This can be done in a text editor:

- Open `application.yml` with in a text editor. **REMARK:** Editors like Microsoft Word or WordPad cannot be used.
- Edit test site (a corresponding `pilot-site` property) and use case properties in the upload section, e.g., as shown in the example:

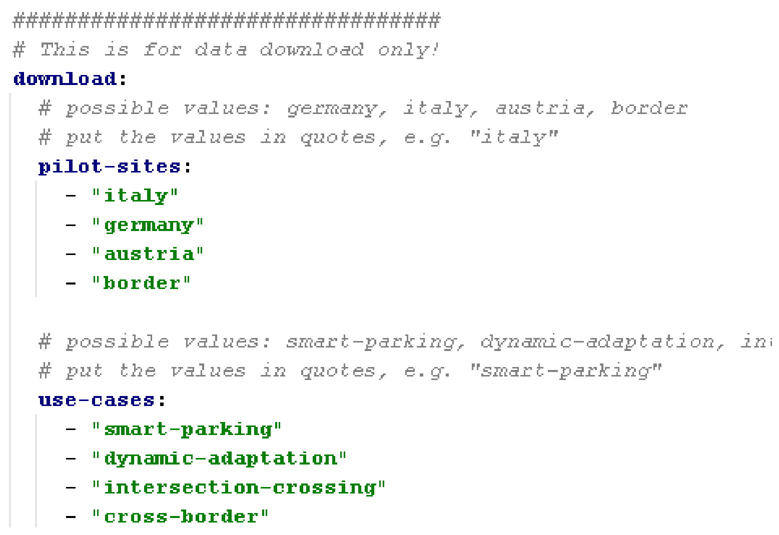


```
data-collector:
  ##################################
  # this is for data upload only!
  upload:
    # possible values: germany, italy, austria, border
    # put the values in quotes, e.g. "italy"
    pilot-site: "germany"

    # possible values: smart-parking, dynamic-adaptation, intersection-crossing, cross-border
    # put the values in quotes, e.g. "smart-parking"
    use-case: "smart-parking"
```

Figure 4. Upload configuration.

- These values will be used by the tool every time a user wants to upload their data. These properties may be changed if a user works at different test sites or on different use cases. While editing the file, the spaces in front of the property names must be preserved.
- To download data, edit test site (a corresponding `pilot-site` property) and use cases properties in the download section, e.g., as shown below:



```
##################################
# This is for data download only!
download:
  # possible values: germany, italy, austria, border
  # put the values in quotes, e.g. "italy"
  pilot-sites:
    - "italy"
    - "germany"
    - "austria"
    - "border"

  # possible values: smart-parking, dynamic-adaptation, in
  # put the values in quotes, e.g. "smart-parking"
  use-cases:
    - "smart-parking"
    - "dynamic-adaptation"
    - "intersection-crossing"
    - "cross-border"
```

Figure 5. Download configuration.

- These values will be used by the tool every time a user wants to download data. Values of test sites and use cases are used to filter a dataset required to be downloaded. In the example provided, any data for the specified test sites and use cases will be downloaded. If additional filters are required for the data to be downloaded, a user may add additional parameters in the command line, which will be described in the next section.
- Once the configuration setup is done and the file is saved, the tool can be run.

5.3.2 Data Upload

A user can start uploading data by running a command:

```
java -jar data-collection-cli.jar --login=<your_login> --upload --file=generated-10.csv --run=<your run name> --session=<your session name>
```

where:

- `--login` – user's login to the data collector tool, required,

- `--upload` – a parameter to upload a file, required,
- `--file=<path to file.csv>`, a path to a CSV file to upload, required,
- `--run=<trial name>`, a name of the trial, optional,
- `--session=<session name>`, a session, optional,
- `--evaluate`, if present, tells that the uploaded data must be evaluated.

Instead of running the above-mentioned command, a user may edit and run a shell script `upload.bat` / `upload.sh` provided for convenience.

A diagram showing a sequence of operations executed by the CLI for data upload is shown in Figure 6. This sequence starts from a command executed by a user on the command prompt. Then, the tool performs a few steps to upload the data specified by the user:

- First, the provided file is parsed and validated if this is a valid CSV file. This includes parsing the header of the file to get metadata and re-use column names to verify records and generate message.
- If the validation is successful, then the CLI verifies user credentials and get access to the messaging subsystem.
- If validation is not successful, then depending on the problem found in the file the execution path may vary:
  - If the file cannot be parsed for any reason, e.g., it is not a valid CSV file, the CLI tool stops with an error saying that the file cannot be parsed.
  - If the file can be parsed but the validator found inconsistent records (e.g., some values are not annotated), then the tool prompts the user to choose whether to proceed with the data upload or stop the process.
  - If the user decides to proceed then, this may lead to inconsistent data in the database that may cause problem to the project evaluators. This is not recommended to do except when the user tries to upload test data that are not suitable for evaluation.
- If the user has access to send messages, the CLI sends out a single message per each record in the CSV file as a set of key-value pairs.
- If the user does not have access, then the CLI tools stops.
- Once all messages are sent, the tool waits up to 3 minutes to get confirmations from the server that the messages are uploaded correctly and then stops.

The CLI tool writes down a log files with the actions it takes. The log file can be analysed in case of any problems.

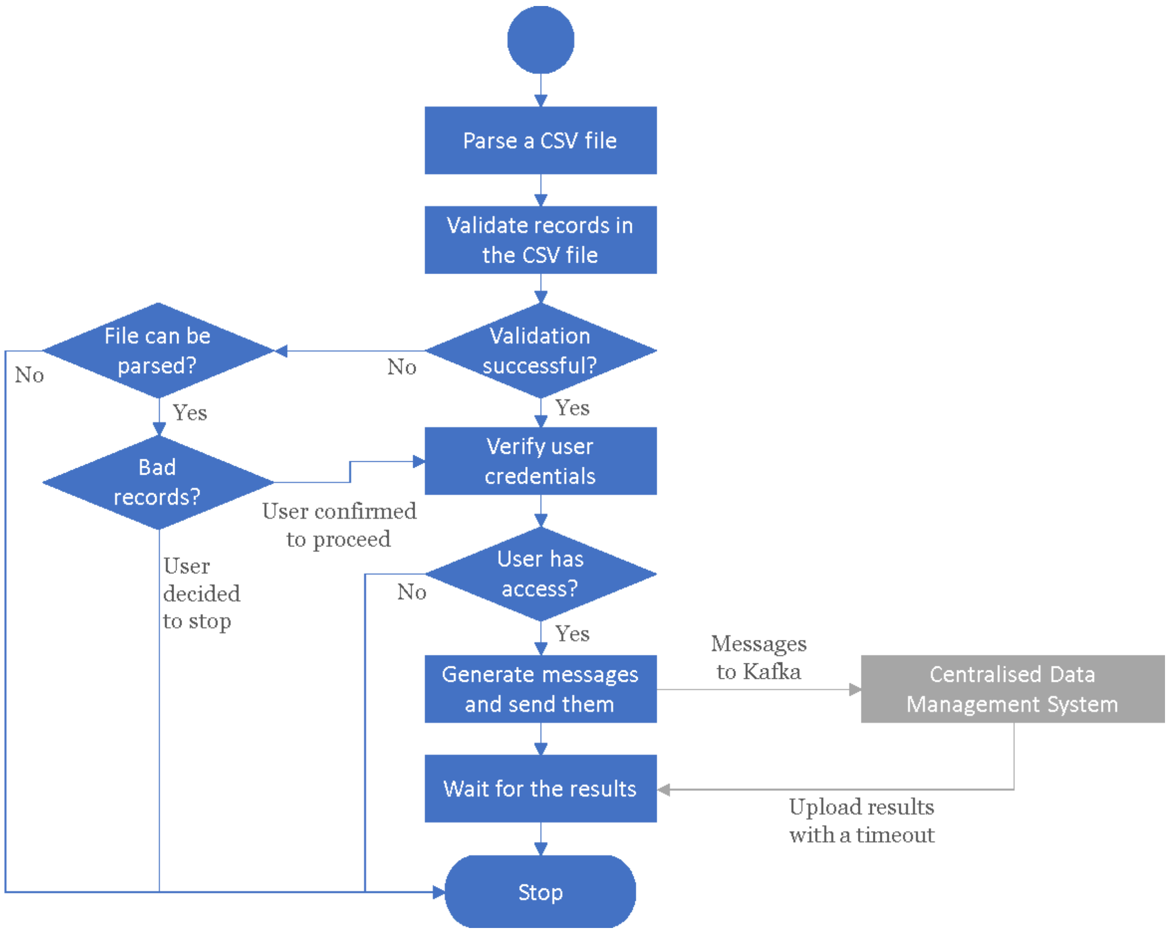


Figure 6. Diagram of actions taken for data upload.

5.3.3 Data Download

A user can download data by running a command:

```
java -jar data-collection-cli.jar --login=<your login> --download --
file=<file name.csv> --session=<your session>
```

where:

- `--login` – user's login to the data collector tool, required,
- `--download` – a parameter to download data, required,
- `--file=<file name.csv>`, a path to a CSV file to upload, required,
- `--session=<session name>`, a session, optional. The user may specify as many of these options as desired, e.g., to filter by two sessions pass: `--session=sess1 --session=sess2`.
- `--run=<trial name>`, a name of the trial, optional. The user may specify as many of these options as desired, same as for session.

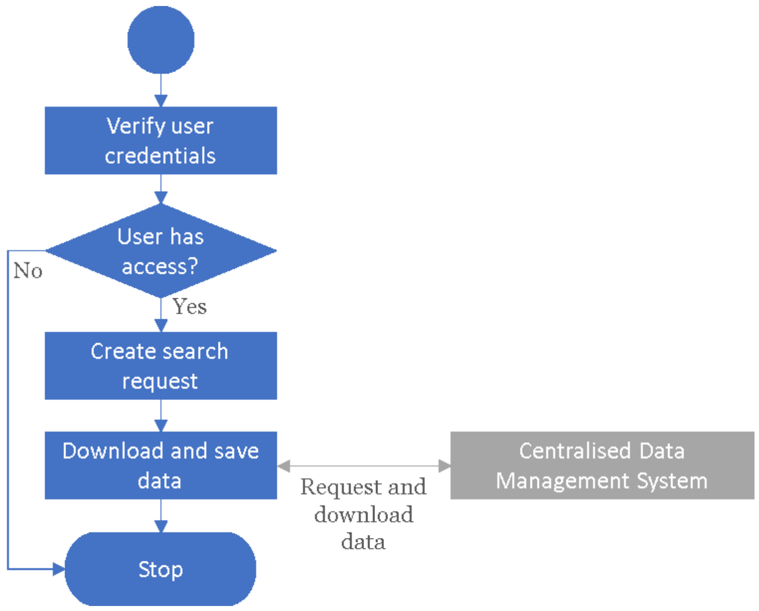


Figure 7. Diagram of actions taken to download data.

A diagram showing a sequence of operations executed by the CLI for data download is shown in Figure 7. This sequence starts from a command executed by a user on the command prompt. Then the tool performs a few steps to download the data specified by the user:

- The process starts with verification of the user's credentials and, if it is successful, the CLI creates a search request to download data. If the verification is not successful, the CLI terminates.
- Once the request is sent, the CLI waits for data, then converts the received data to the CSV format and saves it to the file provided.

5.4 Data API

There are two main APIs supported by the centralised data management system:

- Messaging Kafka based API. This API is solely used for data upload.
- Request/Response REST API. This API is used to get access to Kafka and download data.

5.4.1 Messaging API

Once an application has Kafka credentials, it can send logged messages via Kafka API. Please refer to [Kafka] for the documentation. In this section, only data collection specifics of the API are described. A message from an application must:

- Send a message to a one of the available topics: `germany`, `austria`, `italy`, `border`. As it stands from the topic names, there is one topic per test site.
- Use a use case name as a key value in the Kafka API. Possible values: `smart-parking`, `dynamic-adaptation`, `intersection-crossing`, `cross-border`. These values must be used without quotes.
- A message payload must be a string serialised JSON dictionary with a set of predefined keys that corresponds to the CLI configuration. These keys start with double underscore and end up with double underscore.
  - A key `__pilot-site__` corresponds to a test site where this message is logged. Value for this key must be one of `germany`, `austria`, `italy`, `border`.

- A key `__use-case__` corresponds to a use case in which this message is logged. Value for this key must be one of `smart-parking`, `dynamic-adaptation`, `intersection-crossing`, `cross-border`.
- A key `__session-name__` corresponds to a session name in which this message is logged. This key is optional and may have an arbitrary value.
- A key `__run-name__` corresponds to a run name in which this message is logged. This key is optional and may have an arbitrary value.
- A key `__evaluate__`, a Boolean value, must be set to `true` if the message should be used in evaluation, otherwise it should be set to `false`.

- Other key/value pairs may contain any values suitable for evaluation purposes. Custom keys cannot start and end up with double underscore.

Access for partners to the messaging API is provided upon request.

5.4.2 REST API

This API is used by the CLI tool to get access to the messaging API and to download data. The associated API requests are listed in Table 4 and Table 5.

Table 4. Get messaging API configuration request.

Description	Get messaging API configuration
Request method	`GET`
Path	`/api/kafka/sasl-jaas-config`
Header	Basic credentials
Response	A JSON dictionary with a single entry: the key is `sasl.jaas.config` and the value is a Kafka configuration parameter to be used.

Table 5. Download data request.

Description	Download data request
Request method	`POST`
Path	`/api/download`
Header	Basic credentials
Body	A JSON search data request. May contain properties: • `pilotSite`, a set of strings that correspond to the test site names to filter by. • `useCase`, a set of strings that correspond to the use case names to filter by. • `sessionName`, a set of strings that correspond to the session names to filter by. • `runName`, a set of strings that correspond to the run names to filter by. • `evaluate`, a Boolean value whether to look for data uploaded for evaluation purposes or not.
Response	A streamed list of dictionaries where each dictionary corresponds to a single message, the format of the message is described in the section 5.4.1.

6 Conclusion

The ICT4CART project proposes a methodology and a centralised data management system to collect, aggregate, store, and download data in the tabular format, allowing the evaluators to work with aggregated data. The methodology and the system ensure that data required for the evaluation are generated according to precise requirements and made easily available for technical evaluation and impact assessment.

This deliverable also defines a distributed data collection process that starts at the test sites where raw data are collected in the use-cases and shows what type of data were collected in the project. It is also introduced the centralised data management system that aggregates the collected data and stores them in a centralised environment, which was shared and made available to the evaluators. An overall architecture and implemented solutions which guarantee data provisioning of the right data for the right user are described in this deliverable.

All the collected data at the test sites are uploaded into the centralised data management system using a dedicated interface developed in this task and aggregated by the system. This interface unifies the way the collected data are transferred to the project evaluators and made available to all interested parties.

7 References

[D1.3] ICT4CART, "Deliverable D1.3 – Data Management Plan", February 2019. Access is restricted only for members of the consortium.

[D2.1] ICT4CART, "Specification of Use Cases", March 2019. https://www.ict4cart.eu/assets/deliverables/ICT4CART_D2.1_Specification-of-Use-Cases_v1.0_final.pdf

[D7.1] ICT4CART, "D.7.1 Test Sites Planning", August 2019. Access is restricted only for members of the consortium.

[D7.3] ICT4CART, "Deliverable D7.3: Italian test site", November 2021. Access is restricted only for members of the consortium.

[D7.4] ICT4CART, "Deliverable D7.4: German Test Site", November 2021. Access is restricted only for members of the consortium.

[D7.5] ICT4CART, "Deliverable D7.5: Austrian Test Site", December 2021. Access is restricted only for members of the consortium.

[D7.6] ICT4CART, "Deliverable D7.6: Cross-border Test Site", November 2021. Access is restricted only for members of the consortium.

[D8.1] ICT4CART, "Deliverable D8.1: Evaluation methodology", July 2020, https://www.ict4cart.eu/assets/deliverables/ICT4CART_D8.1_Evaluatuon-Methodology_v1.0_Final.pdf

[D8.2] ICT4CART, "Deliverable D8.2: Technical evaluation – Version 1", June 2021, https://www.ict4cart.eu/assets/deliverables/ICT4CART_D8.2_Technical_evaluation-Version_1.pdf

[D8.3] ICT4CART, "Deliverable D8.2: Technical evaluation – Final Version", not yet published, will be available on https://www.ict4cart.eu/hub/deliverables

[Geo-Ulm] Geodetic reference point in Ulm, https://www.ldbv.bayern.de/vermessung/satellitenpositionierung/referenzpunkte/neuulm.html

[Kafka] Apache Kafka, https://kafka.apache.org/documentation/

[Mongo]	MongoDB, https://www.mongodb.com/
[pcap]	Network packet capture interface, https://en.wikipedia.org/wiki/Pcap
[Streams]	IBM Event Streams, https://www.ibm.com/ie-en/cloud/event-streams
[Wiki-CSV]	Comma-separated values, https://en.wikipedia.org/wiki/Comma-separated_values
[Wireshark]	Wireshark is a free and open-source packet analyser, https://www.wireshark.org