Ecosystem for Services based on integrated Cross-sectorial Data Streams from multiple Cyber Physical Products and Open Data Sources

Cross-CPP MARKETPLACE DEVELOPER GUIDE

CPP DATA PROVIDER GUIDE

(CROSS-SECTORIAL ORIGINAL EQUIPMENT MANUFACTURER)

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 780167

Introduction

Data Provider Developer Guide offers data providers the information needed to connect a Company Backend system to the Cross-CPP Cloud Storage to share CPP data in Data Packages using Common Industrial Data Model (CIDM)

Purpose

This guide aims to help developers from an Original Equipment Manufacturer (OEM from now on) company about how to develop a Company Backend system capable of sharing data with the Cross-CPP solution.

Audience

This guide is meant for and solely for developers of OEM companies that wants to share data from CPP Owners through the Cross-CPP solution.

Scope

The content of this guide is meant to be taken into consideration only when developing a Company Backend system (CB from now on) looking to work with Cross-CPP Marketplace and will only cover functionalities meant to be used by those developers.

Cross-CPP team does not take responsibility on bad use of the application or the data provided when not following the instructions given in this guide.

Troubleshooting

For any questions or inquiries about the use of the Cross-CPP Cloud Storage API or SDK, or the contents of it or this guide, or if you find there is no content in this guide for some functionality please forward it to: marketplace-support@cross-cpp.eu.

Contact

Cross-CPP Project website: https://cross-cpp.eu

Cross-CPP Marketplace: https://datagora.eu

Marketplace support: marketplace-support@cross-cpp.eu

Context Monitoring and Extraction Module (CME): context-support@cross-cpp.eu.

Contents

Guide

Cross-CPP Marketplace Company Backend (CB) developers guide offers five distinct sections for Data Providers Company Backend (CB):

- Sending Data Process: complete workflow of data sharing
- Common Industrial Data Model (CIDM): Cross-CPP data model specification
- Cloud Storage API: REST API through which the data is sent to be stored in the Cloud Storage (CS) and made available for the Cross-CPP Marketplace.
- SDK
- Context Monitoring and Extraction (CME)

1. Sending Data Process

This process describes the required steps a CB needs to perform in order to send data to the CS.

1. Register Company Backend into the Cloud Storage
    a. CB must be CIDM compliant
    b. CB receives API key to operate with the CS
2. CPP Owners registration
    a. CPP Owners must register on their own
    b. Registered CPP owners must give write access
3. Data Storage

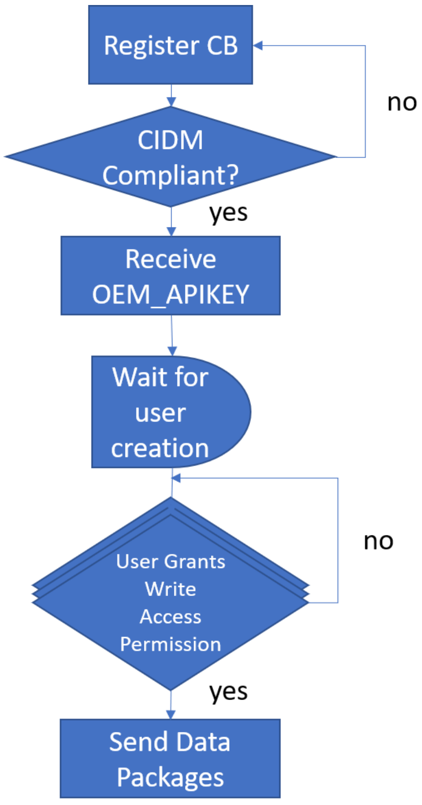


Figure 1. Sending Data process

1.1. Register the Company Backend into the Cloud Storage

The first step for the OEM is to register the Company Backend (CB) into the Cloud Storage (CS).

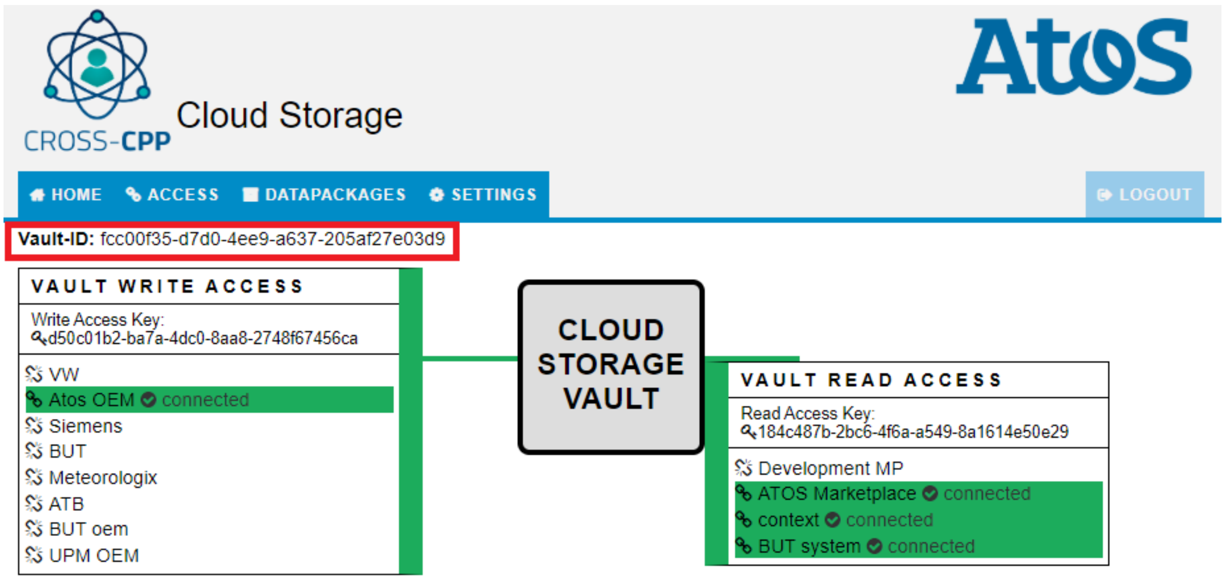
For this, the OEM have to provide the company name and a URL, that also will be used to receive notifications,

The administrator of the Cloud Storage will register a CIDM compliant Company Backend[1] by means of the Admin panel of the Cloud Storage. After the registration, the CB administrator will receive the **APIKEY** to configurate CB-CS communication.

1.2. CPP Owners registration

CPP Owners (vehicle or building owners) have the right to grant access to write their own data coming from the Company Backend into the Cloud Storage.

CPP Owners can grant write permission to the CB through the CS frontend: The screen capture shows the CPP Owner interface to grant write access permission. Once the permission is granted the name of the OEM Company is green. CB would receive the **vault-id** to start sending data to the CS. The vault id is the identification of the secure owner data space related to a vehicle or building.



Vault-ID: fcc00f35-d7d0-4ee9-a637-205af27e03d9

VAULT WRITE ACCESS

Write Access Key:
d50c01b2-ba7a-4dc0-8aa8-2748f67456ca

VW
Atos OEM connected
Siemens
BUT
Meteorologix
ATB
BUT oem
UPM OEM

VAULT READ ACCESS

Read Access Key:
184c487b-2bc6-4f6a-a549-8a1614e50e29

Development MP
ATOS Marketplace connected
context connected
BUT system connected

Figure 2. Cloud Storage Permissions View.

[1] A CIDM compliant Company Backend, is a Company Backend that implements the APIs provided by the Cloud Storage (CIDM compliant) and can therefore store CPP stream data in the Cloud Storage in the CIDM format.

1.3. Data Storage

Once the data owner grants the CB write permission, the CB can send data to the CS. The CB uses the Data Storage API endpoint to send data in the CIDM format. The request must provide the OEM_APIKEY in order to authenticate the CB and to check the authorization of the vault-id of the data-package.

This is the main request for Data Providers as is the entry point of data collected from their CPPs.

Note that the request body is always an array, and every package is treated separately, meaning the array can contain any amount of data from different signals and data types.

REQUEST	**PUSH DATA PACKAGE**	
Method	POST	
Url	https://cloudstorage-api.datagora.eu/api	
Endpoint	/datapackages	
Headers	Authentication	{{oem_apikey}}
	Content-type	application/json

Body

```
[
    {
        "cvim-version": "1.2.0"                                 // string
        "vault-id": "---",                                      // uuid
        "measurement-channel-id": "151",                        // string
        "type": "histogram",                                   // enumeration string
        "timestamp-start": "2019-05-20T17:01:32.402330Z", // date-time
        "timestamp-stop": "2019-05-20T17:18:00.402330Z", // date-time
        "data": [ * ],                                          // array of values
        "cpp-type": "vehicle",                                 // enumeration string
        "trip-id": "---",                                       // uuid
        "sumbmit-time": "2019-05-20T17:19:00.402330Z ",  // date-time
        "expiration-date": "2018-12-31T23:59:00Z",       // date-time
        "mileage-start": xx,                                    // number
        "mileage-stop": xx,                                     // number
        "room-id": "---",                                       // string
        "geo-bounding-box": {
            "latitude-max": 51.517008,                          // number
            "longitude-min": 7.4256,                            // number
            "longitude-max": 7.48278,                           // number
            "latitude-min": 51.501453                           // number
        },
        "location": {
            "latitud": 51.517008,                               // number
            "longitude": 7.4256                                 // number
        },
        "data-masiking-active": false                           // boolean
        "signatures": [    // array of objects about security signatures
            {
                "signatory": "OEM",                             // string
                "signature": "Rr---------4Zw==",            // encrypted string
                "checksum": "4f3-------d71"                     // string
            }
        ],
        "data-ownership-information": {
            "data-privacy-level": "public",             // enumerated string
                   ["public", "shared", "private"]
            "data-stakeholders": [
                {
                    "status": "Creator",                // string
                    "name": "CPP owner name"            // string
                },
                { ... }
            ],
            "copyright-stakeholders": [
                {
                    "status": "Creator",                // string
                    "name": "CPP owner name"            // string
                },
                { ... }
            ],
            "privacy-veto-rights": {
               "consent-level": "public",               // enumerated string
                   ["public", "shared", "private"]
               "data-format": "time-series",            // enumerated string
                   ["time-series", "histogram"]
               "jurisdiction": "Europe",                // enumerated string
                   ["Europe", "any"]
               "storage-constraint": "OEM storage"      // enumerated string
                   ["OEM storage", "Personal storage"]
            }
        }
    }
]
```

Table 1. Push Data Package request

Body explanation (for each data package):

- **vault-id**: identificatory of the cloud storage vault belonging to the data owner
- **measurement-channel-id:** channel to sample the data received
- **type**: type of data. Must be the same as the type of channel (time-series, histogram, geo-histogram, event-based, general-purpose, basic-cpp-information)
- **data**: array of data. The entities expected depend on the type of data
- **geo-bounding-box / location**: squared boundaries of the location, or exact location, in which the data was generated
- **signatures**: encrypted security signatures from the OEM
- **data-ownership-information**: optional object including the information relative of the nature of the data, including privacy levels and rights, involved stakeholders and copyright entities.

The body of the request is an array of CIDM data-packages. Even though the CIDM specification requires the "submit-time" and "datapackage-id" you don't need to provide them as the CS will provide those values automatically when the data-package is saved. The CS will validate that the data-packages are CIDM compliant and response OK.

Every data-package must provide the "vault-id" in order to assign the data-package to the proper storage space of the user. The "type" field specifies the type of data package, the main types are "time-series", "basic-cpp-information", "event-based", etc (see CIDM section 2 for further details). The "measurement-channel-id" specifies the signals that the data-package collects (you can check the list of measurement-channels available on the Marketplace Catalogue or ask the administrator to create a new one that fits to your needs). The format of the "data" object will depend on the "type" and the "measurement-channel-id" of the data-package.

1.3.1. Time-series

```
{
    "type": "time-series",
    "number-of-samples": 3                                  // integer (required)
    "data": [                                               // object array (required)
        {
            "timestamp": "2019-05-20T17:08:29.607343Z",     // date-time
            "value": [                                      // values array
                *                                           // value per channel signal
            ]
        },
        { … },
    ],
    "statistic-properties": {                               // object (optional)
        "min": 0,                                           // number
        "max": 3,                                           // number
        "average": 2,                                       // number
        "histogram": {                                      // object
            "measurement-channel-id": "2"                   // string
            "data": [                                       // array
                2,                                          // number
                ...
            ]
        }
    }
}
```

Listing 1. Time-series type data package

```
* "data" length must be equal to "number-of-samples" value
```

1.3.2. Histogram

```
{
    "type": "histogram",
    "data": [                           // number or number arrays array (required)
        1,                              // number
        …
    ]
}
```

Listing 2. Histogram type data package

1.3.3. Geo-histogram

```
{
    "type": "geo-histogram",
    "data": [                                                   // object array (required)
        {
            "tileY": 21790,                                     // number
            "tileX": 34123,                                     // number
            "histogram": [                                      // number array
                44,                                             // number
                ...
            ]
        },
        { ... },
    ]
}
```

Listing 3. Geo-histogram type data package

1.3.4. Event based

```
{
    "type": "event-based",
    "data": [                                                   // object array (required)
        "event-type": "real-time-event"                         // enumeration string *
        "event-data": {
            "event-time-stamp": "2019-05-20T17:08:29.607343Z",  // date-time
            "event-datapackages": [ DataPackage model ]         // DataPackage object array
            "value": ""                                         // string
        }
    ]
}
```

Listing 4. Event-based type data package

```
* "real-time-event", "trigger-event", "threshold-event"
```

1.3.5. General purpose

```
{
    "type": "general-purpose",
    "data": {                                                   // object (required)
        "key": "value"                                          // configured key: value
    }
}
```

Listing 5. General purpose type data package

1.3.6. Basic CPP information

```
{
    "type": "basic-cpp-information",
    "data": {                                                   // object (required)
        "VehicleColor": "red"                                   // configured key: value
    }
}
```

Listing 6. Basic CPP information type data package

Responses:

Code		Description
200	OK	Request successful
400	Bad Request	Query was malformed or incorrect
401	Unauthorized	Missing authorization token Unauthorized role Unauthorized user
404	Not found	Something requested does not exist
500	Internal Server Error	Something else went wrong

2. Common Industrial Data Model (CIDM)

Cross-CPP uses the CIDM as its data model. All data pushed into Cross-CPP Marketplace must follow this model.

2.1. Model Architecture

The CIDM architecture consist of three layers:

- The Signal layer consisting of the information provided by the CPP devices like vehicles or smart buildings. Signals are generated by sensors that observe the environment and produce data, as they detect physical and chemical phenomenon, for example, speed, temperature, charge state level, etc.
- The Measurement Channel layer providing signals data aggregation. The data needs to be pre-processed since raw sensor data exceeds the available storage and transferring capacity, to reduce the size of data down-sampling and histograms methods are provided.
- The data layer aggregating data inside data packages to store and transfer. One data package contains data from exactly one signal measured with one Measurement Channel. In addition to the actual data, Data Packages contain header information ("meta data"). This header information provides ownership of the data and gives quality of signal indications by OEM signatures or describes parameters of the measurement (e.g. time, rough position estimate, etc.).

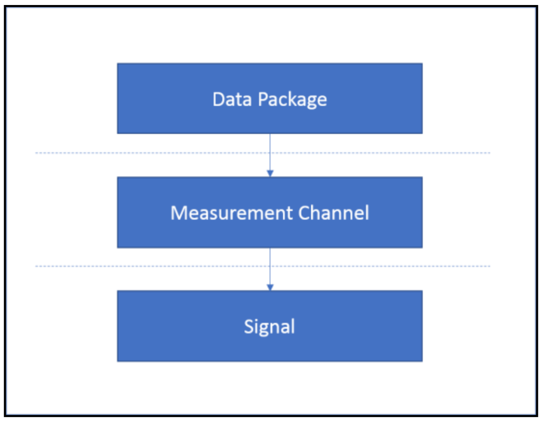


Figure 3: Layered High-level View of the Common Industrial Data Model (CIDM)

2.2. Signal Layer Specification

Sensors are the perception organs of CPP devices like vehicles and buildings. It is their main duty to detect physical phenomenon and chemical quantities by transferring them into electrical signals. The signal layers describe different types of signals and formats represented in the system. A new property is needed to group signals regarding the signal source type, cpp-type. Figure 4 shows the UML modelling of the signals for CIDM.

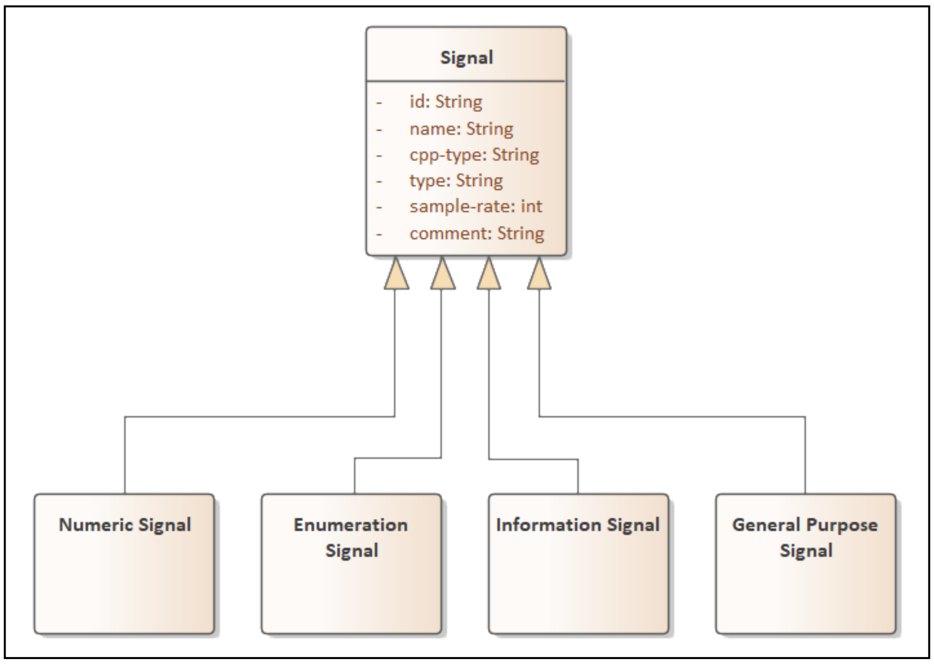


Figure 4. Signal UML Model.

The cpp-type is a required property that must be one of the two values, “vehicle” or “building”. The table below shows the complete definition of the Signal.

Property	Occurrence	Type	Format	Description
Common Properties				
id	Required	String		Unique Identifier of the Signal
name	Required	String		Name of the Signal
cpp-type	Required	String	one of: - vehicle - building	Type of the CPP
type	Required	String	one of: - numeric - enumeration - information - general-purpose	Type of the Signal
format	Optional	String		Signals representation format
sample-rate	Required	Numeric	double	Sample rate in Hz (Samples per Second). Must be larger than or equal to zero.
comment	Optional	String		Description of the signal
Numeric Signal				

Property	Occurrence	Type	Format	Description
type	Required	String	"numeric"	Type of the Signal needs to be numeric
format	Required	String	*<numeric formats>*	Signal's numeric representation (e.g. uint8, double, etc.)
min	Required	Number	<according to format>	Minimum Signal value
max	Required	Number	<according to format>	Maximal Signal value
resolution	Required	Number	<according to format>	Signals resolution
Unit	Required	String		Unit of the Signal (e.g. ºC)
Enumeration Signal				
type	Required	String	"enumeration"	Signal's type attribute needs to be "enumeration"
items	Required	Array	String	String array with possible Signal values
Information Signal				
type	Required	String	"information"	Signal's type attribute needs to be "information"
format	Required	String		Signals representation format (e.g. VIN, etc.)
General Purpose Signal				
type	Required	String	"general-purpose"	Signal's type attribute needs to be "general-purpose"
*	Optional	Any	No	May be extended with further attributes

Table 2. Signal property definition.

2.3. Measurement Channel Layer Specification

The measurement layer defines how sensor signals are captured and processed. One Measurement Channel describes how samples from one (or more - in the case of multidimensional histograms) sensor signal are aggregated and measured. Figure 5 shows the Measurement Channel UML model.

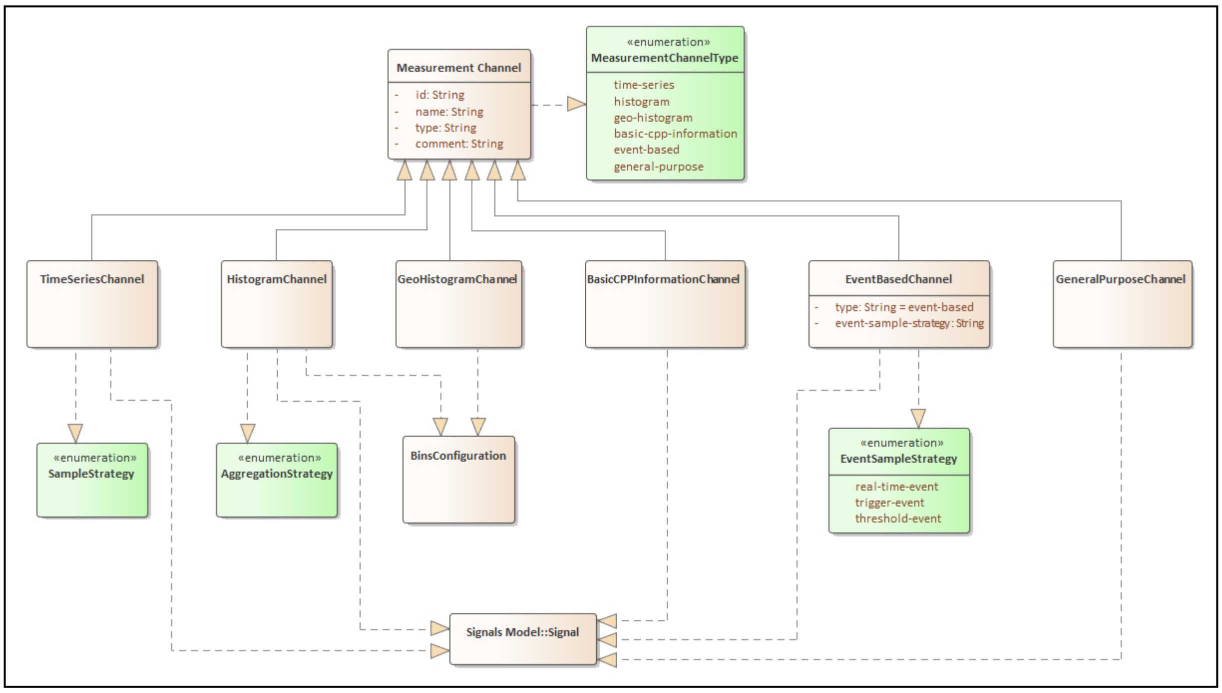


Figure 5. Measurement Channel UML Model.

The basic CPP information channel provides static information that is not measured by sensors but provides information about the CPP device, like the colour of a vehicle, the number of floors of a building, the identification number of a car, etc.

The event-based measurement channel provides information of events that occurs when the value of a measurement gets to a specific value (real-time-event) or when the value passes a specific threshold (threshold) and the last one that provides information about the event and the data-packages that have triggered the event. The table below details the measurement channel specification.

Property	Occurrence	Type	Format	Description
Common Properties				
id	Required	String	No	Unique Identifier of the Measurement Channel
name	Required	String	No	Name of the Measurement Channel
type	Required	String	one of: - time-series - histogram - geo-histogram - general-purpose - event-based - basic-cpp-information	Type of Measurement Channel
Comment	Optional	String	No	Description of the signal

Property	Occurrence	Type	Format	Description
Time Series Measurement Channel				
type	Required	String	"time-series"	Type of the Measurement Channel needs to be time-series
format	Required	String		Data type format of the samples
dimension	Optional	Number	uint32	Dimension of the Time-series. If dimension is not given, one-dimensional is assumed
capture-interval	Required when on-change is false	Number	double	Capture interval between two samples in seconds. Only required, when on-change is false.
on-change	Required	Boolean		Does Measurement-Channel only record changes in signal
sample-strategy	Required	String	one of: - min - max - average - last-known-value	Signal sampling strategy
signal	Required	Object	Array of Signal Object	See section 6.1.2 for Signal object definition
Histogram AND geo-Histogram Measurement Channel				
type	Required	String	one of: - histogram - geo-histogram	Type of the Measurement Channel needs to be histogram or geo-histogram
aggregation-strategy	Required	String	one of: - time - count - min - max	Histogram values aggregation strategy
capture-interval	Required	Number	double	Capture Interval of one Histogram. Needs to be larger than zero. +Infinity is valid (see IEEE 754).
dimensions	Required	Number	uint32	Dimensions of the Histogram
bins	Required	Array	Bin-Configuration Object	Array of bin configurations. Array

Property	Occurrence	Type	Format	Description
				needs to contain exactly one configuration for every dimension/axes of the histogram
Geo-Histogram Measurement Channel				
type	Required	String	"geo-histogram"	Type of the Measurement Channel needs to be geo-histogram
geo-resolution	Required	Numeric	double	Zoom level of the geo-histogram
Basic CPP Information Measurement Channel				
type	Required	String	"basic-cpp-information"	Type of the Measurement Channel needs to be basic-cpp-information
signal	Required	Object	Signal Object	See section for Signal object definition
Event Based Measurement Channel				
type	Required	String	"event-based"	Type of the Measurement Channel needs to be event-based
format	Required	String		Data type format of the samples
event-sample-strategy	Required	Event Sample	one of: - real-time-event - trigger-event - threshold-event	Event sampling strategy
comment	Optional	String	No	Description of Event Strategy
General Purpose Measurement Channel				
type	Required	String	"general-purpose"	Type of the Measurement Channel needs to be general-purpose
signal	Required	Any		See section 6.1.2 for Signal definition

Table 3: Measurement channel definition

2.4. Data Package Layer Specification

Data Packages contain the actual data of Signal measurements. As Signals are the information providers and Measurement Channels define the process of data acquisition from those Signals, Data Packages provide a structure for storing the data. In addition, they provide meta / header information containing time of recording, data ownership information, etc. Data Packages contain data from exactly one Measurement Channel. This leads to six different types of Data Packages that are defined similar as the Measurement Channels:

- Time Series Data Package
- Histogram Data Package
- Geo-Histogram Data Package
- Event based data Package
- Basics CPP information Data Package
- General Purpose Data Package

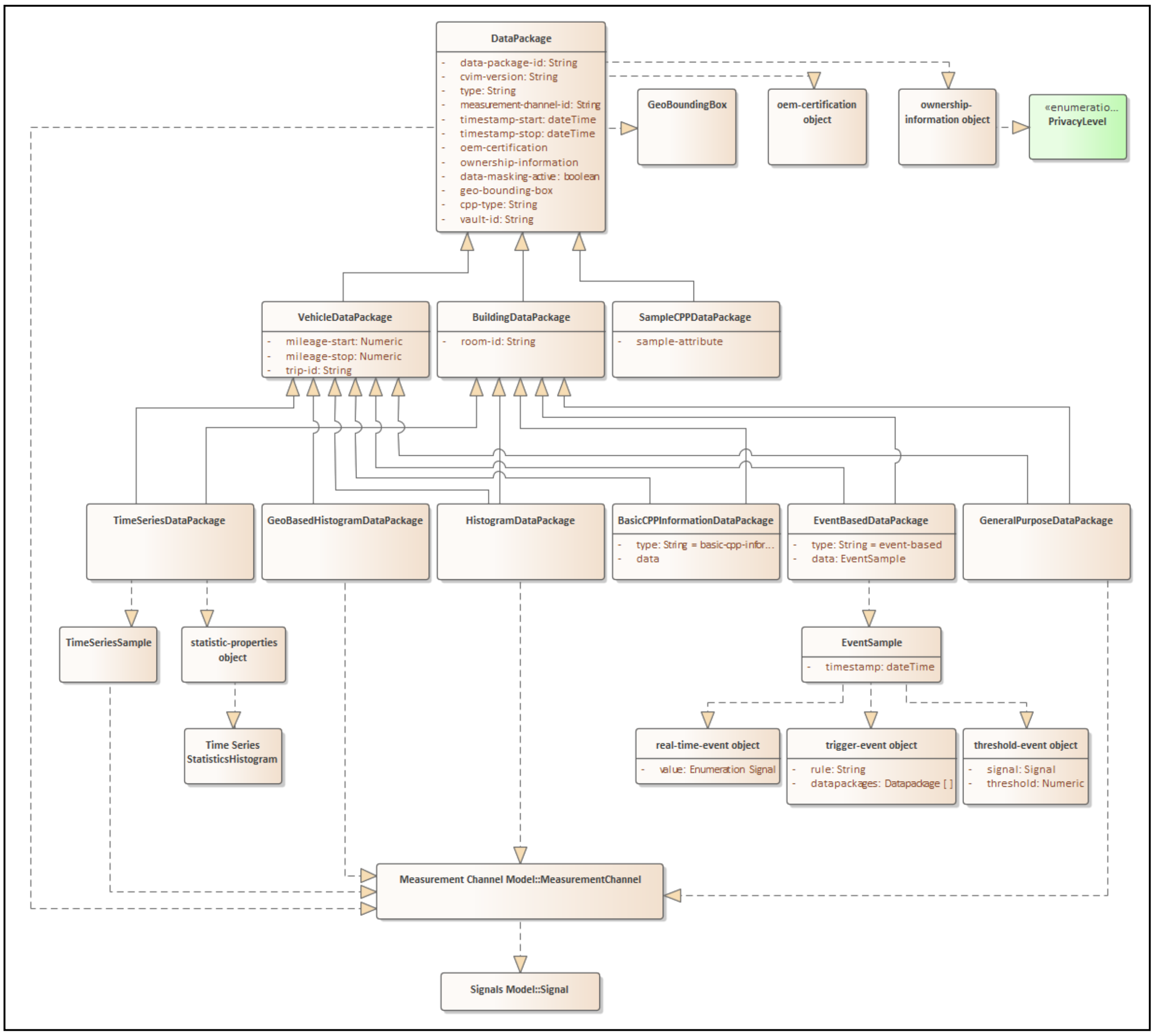


Figure 6. Data Package UML Model.

The data-package definition includes cpp-type to indicate is the data-package belongs to a "building" or a "vehicle" cpp and definition of data-package type, such as basics-cpp-information data-package and the event-based data-package.

The data of basics-cpp-information data-package depends of the signal type definition of the measurement channel - numeric, enumeration, information or general-purpose.

Event-based data-package has an additional property, "event-sample-strategy", to indicate three different type of event, real-time, trigger and threshold. According to the type of event, the "data" object has two mandatory properties, "timestamp" and "value", and an optional property named "datapackages" that is an array of the data-packages that triggers the "trigger-event".

The "building" CPP devices provides a set of sensors distributed along the rooms of the building, in order to identify the devices of the same room a new property has been included, "room-id".

Property	Occurrence	Type	Format	Description
Common Properties				
data-package-id	Required	String	UUID	Identifier of the Data Package. Unique per Cloud Storage Vault, Set by Cloud Storage Provider
cvim-version	Required	String	version	The name of the property is for backward compatibility with CVIM[2]. Must be set to 1.2.1
type	Required	String	one of: - time-series - histogram - geo-histogram - general-purpose - event-based - basic-cpp-information	Type of the Data Package
vault-id	Required	String	UUID	ID of the Cloud Storage Vault, where the data is stored in.
cpp-id	Optional	String	any	ID of the CPP
cpp-type	Required	String	one of: - vehicle - building	Type of the CPP
trip-id	Optional	String	any	Trip-ID of the User

[2] Common Vehicle Information Model: the basis for the current CIDM

Property	Occurrence	Type	Format	Description
room-id	Optional	String	any	ID of the room in a building where the measurement data was collected
measurement-channel-id	Required	String		Identifier of the Measurement Channel whose data is inside this data package
mileage-start	Optional	Number	double	Mileage at the start of the measurement in kilometres (km)
mileage-stop	Optional	Number	double	Mileage at end of measurement (km)
geo-bounding-box	Optional	Object	Geo-Bounding-Object	Geographic bounding box
location	Optional	Object	Location-Object	Single location including latitude and longitude
oem-certification	Optional	Object	OEM-Certification-Object	OEM Certification
data-ownership-information	Optional	Object	Ownership-Information -Object	Data Ownership Information
expiration-date	Optional	String	date-time	Data expiration date
data-masking-active	Optional	Boolean		Indicates status of data-masking (true = active)
Time Series Data Package				
type	Required	String	“time-series”	Type of the Measurement Channel needs to be time-series
timestamp-start	Required	String	date-time	Measurement start time
timestamp-stop	Required	String	date-time	Measurement stop time
number-of-samples	Required	Number	uint32	Number of samples that are stored in data
statistic-properties	Optional	Object	statistic-properties-object	Provides statistic properties about the data
data	Required	Array	time-series key-value-pair -object	Array of time-series-data Objects. The size of the array mist equal number of samples
Histogram Data Package				

Property	Occurrence	Type	Format	Description
type	Required	String	histogram	Type of the Measurement Channel needs to be histogram
timestamp-start	Required	String	date-time	Measurement start time
timestamp-stop	Required	String	date-time	Measurement stop time
data	Required	(Multi-dimensional) Array	Number	Array containing he bin counts. Size of array must match the dimension and bin configuration of the related Measurement Channel. Number format depends on Histogram aggregation-strategy
Geo-Histogram Data Package				
type	Required	String	geo-histogram	Type of the Measurement Channel needs to be geo-histogram
timestamp-start	Required	String	date-time	Measurement start time
timestamp-stop	Required	String	date-time	Measurement stop time
data	Required	(Multi-dimensional) Array	Number	Array containing the bin counts. Size of array must match the dimension and bin configuration of the related Measurement Channel. Number format depends on Histogram aggregation-strategy. The outer most dimension is the geo-dimension. It must match in its size the size of the geo-tiles array.
geo-tiles	Required	Array	Geo-Tile Object	Array of geo-tile objects. Only visited tiles are included.
Basic CPP Information Data Package				
type	Required	String	“basic-cpp-information”	Type of the Measurement Channel needs to be basic-cpp-information

Property	Occurrence	Type	Format	Description
Timestamp	Required	String	date-time	Measurement date time
data	Required	Any		Data depends on the type of signal of the measurement channel
Event Based Data Package				
type	Required	String	"event-based"	Type of the Measurement Channel needs to be event-based
Timestamp	Required	String	date-time	Measurement date time
event-sample-strategy	Required		one of: - real-time-event - trigger-event - threshold-event	Event sampling strategy
data	Required	Object	Event Sample Object	event-based data Object indicating an event
General Purpose Data Package				
type	Required	String	"general-purpose"	Type of the Measurement Channel needs to be general-purpose
Timestamp	Required	String	date-time	Measurement date time
data	Required	Any	time	Datatype depends on Measurement Channel

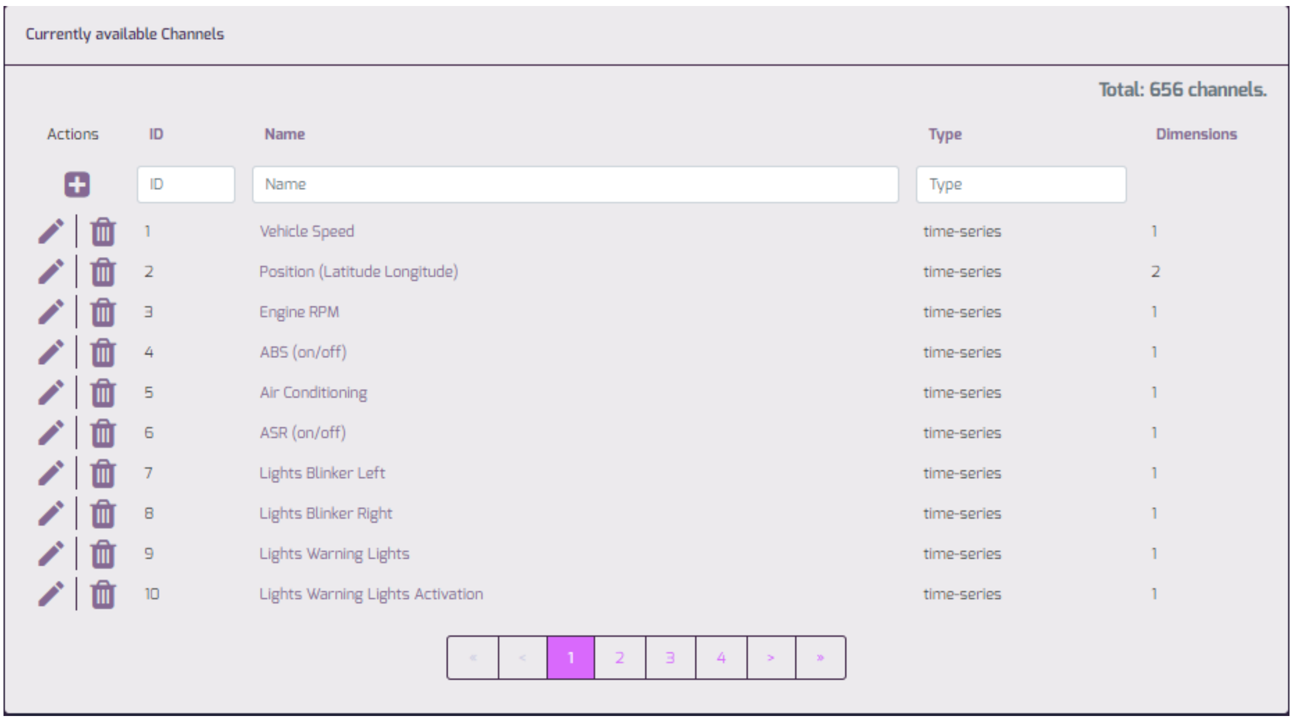
Table 4. Data Package definition.

Property	Occurrence	Type	Format	Description
Timestamp	Required	String	date-time	Timestamp of the event
value	Required	String		e.g. "*Ignition On*", "*Wipers Off*"
datapackages	Optional	Array of data-package		

Table 5. Event Sample Object

2.5. Measurement Channel Catalogue

The complete list of available Measurement Channels in the Cross-CPP Marketplace can be found at https://ng8.datagora.eu/pages/management/channels



Currently available Channels

Total: 656 channels.

Actions	ID	Name	Type	Dimensions
	1	Vehicle Speed	time-series	1
	2	Position (Latitude Longitude)	time-series	2
	3	Engine RPM	time-series	1
	4	ABS (on/off)	time-series	1
	5	Air Conditioning	time-series	1
	6	ASR (on/off)	time-series	1
	7	Lights Blinker Left	time-series	1
	8	Lights Blinker Right	time-series	1
	9	Lights Warning Lights	time-series	1
	10	Lights Warning Lights Activation	time-series	1

Figure 7. Measurement Channel Catalogue

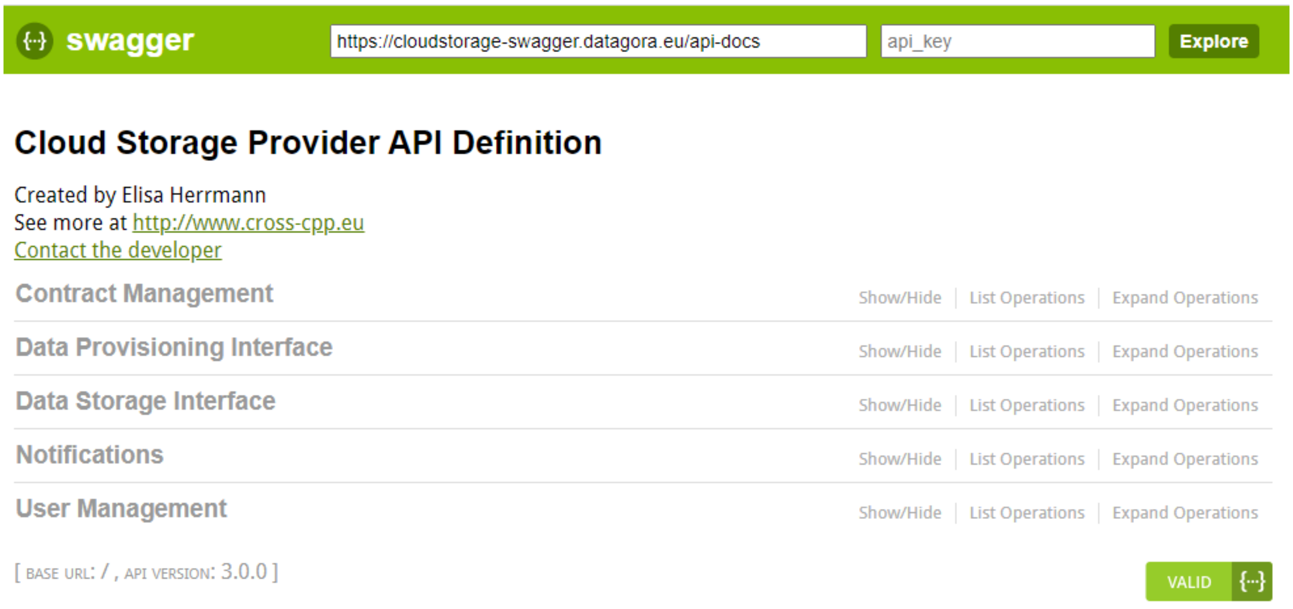
3. Cloud Storage API Specification

Data sent by Data Providers is stored in the Cloud Storage (CS) to make it available for the Cross-CPP Marketplace functionalities such as the data discovery. When Data is received in the CS, the MP will be notified so that it can be collected and sent to those Data Consumers subscribed for that kind of data.

Data received in the Cloud Storage **must follow the Common Industrial Data Model (CIDM).**

The API url (from now on *api_url*) is: **https://cloudstorage-api.datagora.eu/**

The specification of the CS API is provided in the OpenApi specification that allows to describe and visualize RESTful web services. An online reference of version 3 of the API can be found in: https://cloudstorage-swagger.datagora.eu/docs/ under sections:



swagger https://cloudstorage-swagger.datagora.eu/api-docs api_key Explore

Cloud Storage Provider API Definition

Created by Elisa Herrmann
See more at http://www.cross-cpp.eu
Contact the developer

Contract Management	Show/Hide	List Operations	Expand Operations
Data Provisioning Interface	Show/Hide	List Operations	Expand Operations
Data Storage Interface	Show/Hide	List Operations	Expand Operations
Notifications	Show/Hide	List Operations	Expand Operations
User Management	Show/Hide	List Operations	Expand Operations

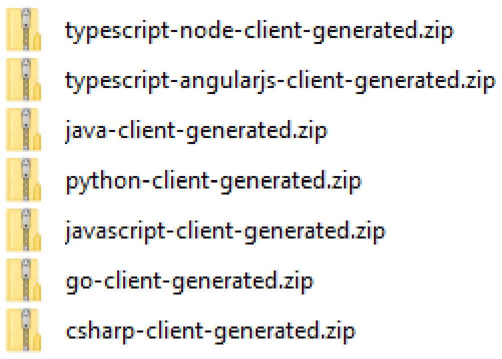
[BASE URL: / , API VERSION: 3.0.0]

Figure 8. Cloud Storage API Specification.

See the complete OpenAPI specification in the Annex OpenAPI Specification of the Company Backend REST API (yaml).

4. SDK

In order to facilitate the integration of the Company Backends with the Cloud Storage there are several SDK provided. The list of the languages supported can be increased by means of tools that automatize the creation from the OpenAPI specification.


typescript-node-client-generated.zip
typescript-angularjs-client-generated.zip
java-client-generated.zip
python-client-generated.zip
javascript-client-generated.zip
go-client-generated.zip
csharp-client-generated.zip

5. Context Monitoring and Extraction (CME)

The CME module provides 2 main customisation endpoints:

- Context models (section 5.1)
- Customisation of existing or creation of new Reasoning Rules (section 5.2)

The customisation explained here and any further customisation of the CME components, either for adding further monitors or reasoning rules for extraction, can be made by downloading and changing the CME module as provided in the open source code project on GitHub[3] under the EPL 2.0 license or by placing a customisation request to the CME team at context-support@cross-cpp.eu.

5.1. Context Models

The Context Models in Cross-CPP are not a software component to be implemented, but models that describes the static and dynamic aspects of use of CPPs.

The Context Models are modelled in the form of ontologies and for this, the context modelling software component is used to create and update the needed models.

As context models describe the situations under which signal value is measured, or a CPP is used, they may vary from CPP to CPP. This means that in the setup phase of the Cross CPP ecosystem, the context model(s) for the specific CPP have to be defined and implemented. The Cross-CPP context model bundle includes one Generic Context Model, which is the basis for the following extensions:

- CPP specific vehicle context model: a specific context model that extends the Generic Context Model with vehicle specific measured and basic signals.
- CPP specific smart infrastructure context model: a specific context model that extends the Generic Context Model with smart infrastructure specific measured and basic signals.
- Vehicle and building discovery extended relational context model: a context model

The chosen context modelling tool was Protégé[4], a free, open-source platform that provides the needed functionality for the Cross-CPP context modelling tool, but any tool of your choice that provides the same functionality to create and update ontologies (*.owl files) could be used. The following txt uses Protégé though when an ontology edit is needed.

5.1.1. Basic principles for context modelling

Some basic principles for context modelling were identified, followed within Cross-CPP and should be taken into account when further developing the provided context models:

[3] Link to be added
[4] https://protege.stanford.edu/products.php

1. Support description of main context:

   In practices, we cannot model all context information, and it is also not realistic. The context model should consider those most related factors according to the requirement of context sensitive adoption.

2. Model the context that is easy acquirable:

   Those context factors considered should be identifiable and acquirable, whether provided through context monitoring services automatically, or by user input explicitly.

3. Trade-off between investment of context modelling/extracting and effects of context sensitive adoption:

   Intuitively, if we could model as much context factors in as much details, the accuracy of context will be higher. However, this does not come for free. On the one hand, more time and efforts are need on context modelling; on the other hand, more computing resources are needed to handle the context, which will bring deficiency to the adoption process.

For each CPP, one has to define which concepts are relevant for the description of the situations (context), under which the CPP signals are generated and measured. Once of the concepts relevant to the description of context of CPP data streams generation are defined, the next step is to define the concepts which are relevant for the application (Cross-CPP module, filtering to be applied, discovery process) where it will be used. As a first approach, and because there are so many CPP measured signals, some general situations are considered (situations that could be interesting for a wide range of cross-sectorial services) and that could bring the most benefit in terms of filtering capabilities that could be offered on the extracted context basis.

The process for defining the Context Model(s) for a CPP is as follows:

- Define CPP specific Context Model (starting from the Generic context model)
- Select functionality of the application for which the context model has to be specified.
- From the existing ontologies and the CPP specific Context Model select a concept relevant for adaptation of the functionality.
- Check whether or not exist CPP basic or measured data in the CPP Cloud Storage
  - If it exists, the concept may be adopted.
  - If it does not exist, check if there are sensor signals from the OEM Backends that are still not yet configured but could be also collected.
  - If this also does not exist, check if is it reasonable to introduce new CPP data concept in the CIDM to provide sensor measured data for it.
- If the concept is selected – define the relative weighing of this concept in the extraction of the current context.
- Repeat the process for each functionality where the extracted context is to be applied

The process is iterative, i.e. based on analysis of the ‘needs’ of each functionality and service, the initial model can be updated, and the process repeated.

5.1.2. Generic context model

The main entities in the generic context model are:

- CPP: comprising all CPPs that are possible
- Activity: the type of activities that can be identified
- Information:
    - Basic CPP Information, comprising all CPP information that are intrinsic for the CPP and do not change (at least not often) such as vehicle colour, sensor height, etc.
    - Sensor Measurement Data, comprising of all sensor signals that can possibly be collected in a certain CPP
- Stakeholder, the actors that are involved in the CPP information value chain

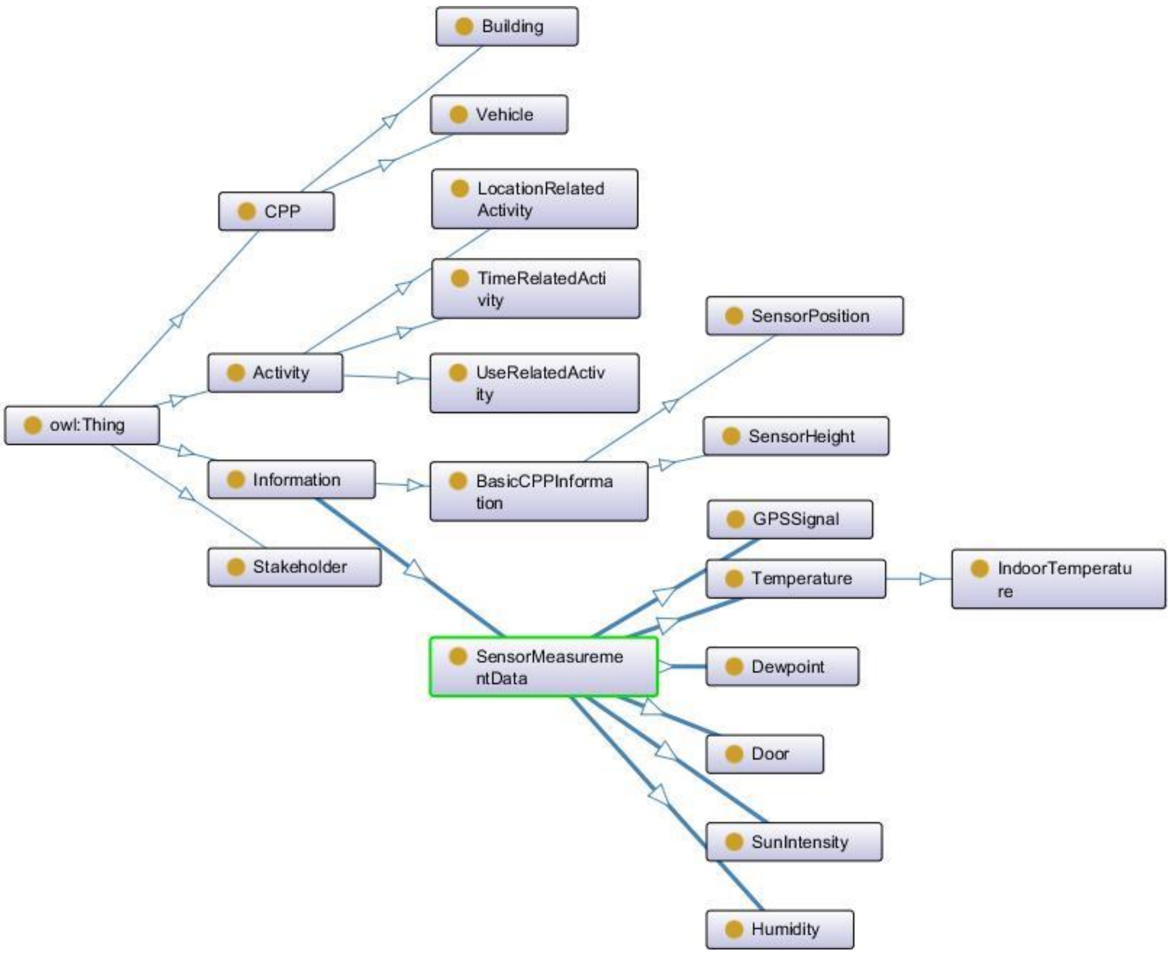


Figure 9: Generic context model

This context model is the basis for the CPP specific context models that exist or will be modelled when that Cross-CPP Marketplace is opened for more OEM data providers.

5.1.3. Creating/Editing of context models

In order to edit any of the currently available CPP specific context models, you need to open the respective ontology in Protégé (*Cross-CPP_Context_Model_v042_vehicle.owl* or *Cross-CPP_Context_Model_v042_building.owl*) and make the necessary edit such as:

1. Add new Sensor Measurement Data
2. Add new Basic CPP Information
3. Add new Stakeholders
4. Save, close and upload in the Context Model repository of the CME module (see more details in section 5.3).

To create a context model for a new CPP you need to:

1. create a new ontology file in Protégé
2. import the generic context model ontology (copy the *genericOntology_v42.owl* file to your working folder and make a direct import in the ontology project that you have created).
3. Follow the same steps as described above for edition of existing ontologies.

The above mentioned models are at the moment available on request (please direct your emails to context-support@cross-cpp.eu) but they will be available in a public GitHub repository free for download under a license to be defined.

The addition of new context models needs to be customised on the Context Monitor component. The CME module is provided as an open source code project on GitHub[5] under the EPL 2.0 license and can be customised. Customisation can also be request via the context-support@cross-cpp.eu.

5.2. Reasoning Rules Configuration

A part of the main adaption work to be done when customising the Context monitoring & extraction module is the introduction of new rules and changing the existing ones. For this purpose, the CME framework provides interfaces both within the code as well as in the form of a freely adaptable configuration file. Both will be described within this section.

5.2.1. Extraction rules configuration file

The configuration file can be found within the main folder of the CME module and is named *extraction_configuration.xml.* Within this file, the following sections can be found which can be adapted by the administrator:

- setup of the reasoning rules for the context-sensitive data discovery

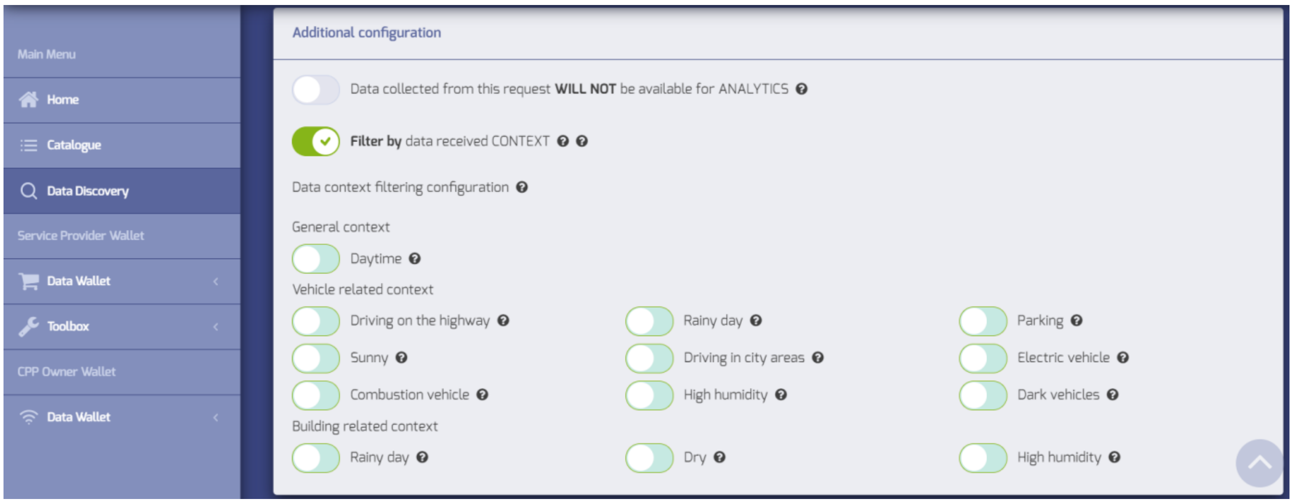
[5] Link to be added

- setup of the reasoning rules for the context-sensitive data access control for the data owner (security feature)
- configuration of the reasoning rules

Setup of the reasoning rules for context-sensitive data discovery

The following

Figure 10 shows the reasoning rules that can be applied by the context-sensitive filtering of data packages during the data discovery as they are listed on the UI of the Marketplace within the *Additional configuration* tab.



Additional configuration

- [ ] Data collected from this request **WILL NOT** be available for ANALYTICS
- [✓] **Filter by** data received CONTEXT

Data context filtering configuration

General context

- [ ] Daytime

Vehicle related context

[ ] Driving on the highway	[ ] Rainy day	[ ] Parking
[ ] Sunny	[ ] Driving in city areas	[ ] Electric vehicle
[ ] Combustion vehicle	[ ] High humidity	[ ] Dark vehicles

Building related context

[ ] Rainy day	[ ] Dry	[ ] High humidity

Figure 10. Context-sensitive data filtering in the Cross-CPP Marketplace UI

In the configuration file, the administrator will find the following structure for building this list of filtering options:

```
<rules>
   <rule id={unique_id} name={rule_name} cppType={cpp_type}
        tooltip={tooltip} />
</rules>
```

Each rule for the context-sensitive data filtering has the following attributes:

- *unique_id:* a unique identification number
- *rule_name:* a name indicating the meaning or scope of the rule
- *cpp_type:* the cpp type to which this rule should be applied in the extraction service (by default *vehicle* and/or *building*)
- *tooltip:* a tooltip which gives more information about the semantic of the rule, which will be shown also in the Marketplace UI

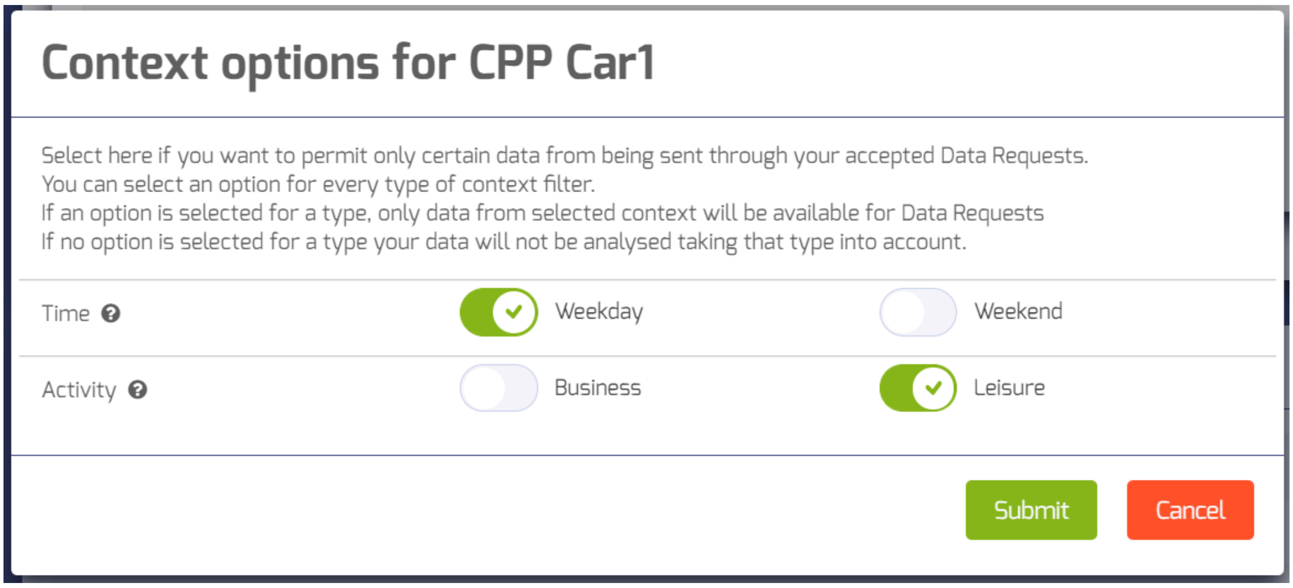
A rule can simply be registered by adding a new *<rule>* element to the *<rules>* section.

Hint: Pay special attention to use user-friendly names, descriptions and tooltips within your rule configuration in order to make them easier accessible for the end users. Make sure to define only rules for which exists a valid configuration within a *<ruleConfiguration>* section.

Setup of the reasoning rules for the context-sensitive data access control for data owners

Find in

Figure 11 a snippet of the context-sensitive data access control options for the data owner in the Marketplace UI, which allows him to control his data access according to specific context parameters.



Context options for CPP Car1

Select here if you want to permit only certain data from being sent through your accepted Data Requests.
You can select an option for every type of context filter.
If an option is selected for a type, only data from selected context will be available for Data Requests
If no option is selected for a type your data will not be analysed taking that type into account.

Time	[✓ Weekday]	Weekend
Activity	Business	[✓ Leisure]

Figure 11. context-sensitive data access control options in the Cross-CPP Marketplace UI

In the configuration file, these options are being defined within the *<dataOwnerContextOptions>* element. Each Option consists of a set of different alternatives for this option and has the following attributes:

- *unique_name:* a unique name for this option
- *tooltip:* a description for this option to be shown as tooltip in the Marketplace UI

Enclosed to an option are different alternatives, specified within the *<alternative>* element as shown below.

```
<dataOwnerContextOptions>
   <option name={unique_name} tooltip={tooltip}>
      <alternative>{alternative1}</alternative>
      <alternative>{alternative2}</alternative>
   </option>
</dataOwnerContextOptions>
```

Hint: in order to function as desired, the choice of options and alternatives has to be aligned with the set of context variables used by the Cross-CPP security module's access control policy. Also make sure to introduce only options and alternatives which correspond to a valid rule configuration (see later in this section).

Configuration of the reasoning rules

Each rule specified within *<rules>* can be configured within a corresponding *<ruleConfiguration>* element. Find the template for the configuration below

```
<ruleConfiguration id={unique_id} internalName={rule_name} cppType={cpp_type}>
   <signalConfiguration id={measurement_channel_id}>
      <value>{value}</value>
   </signalConfiguration>
   <signalConfiguration id={measurement_channel_id}>
      <max>{maximum_value}</max>
      <min>{minimum_value}</min>
   </signalConfiguration>
</ruleConfiguration>
```

The attributes for a rule configuration are:

- *unique_id:* a unique identifier of the rule corresponding to those specified within *<rules>* element
- *rule_name:* the (module internal) name of the rule
- *cpp_type:* the CPP type this rule applies to (by default *vehicle* or *building*)

Each rule configuration is being accompanied by a set of signal configurations, indicating the measurement channels used within the rule. For each measurement channel involved, a *<signalConfiguration>* element will be introduced. The signal configuration specifies the parameter value operators, to which the rule should be applied to. Examples are *maximum (signal < maximum)*, *minimum (signal > minimum)* and *value (signal = value)*. Each of the value operators can be defined within a corresponding *<max>*, *<min>* or *<value>* element, as can be seen in the template above.

5.2.2. Source code customisation

In order to make additional defined reasoning rules function correctly or to edit the already existing rules, the administrator has to adapt the existing CME source code at some specific spots, which will be shown and explained in the following section. Changes will have to be made within the following java classes[6]:

- *ExtractionRule:* In order to make the CME recognise the rule this interface has to be implemented
- *ExtractionOptions:* recently created reasoning rules have to be registered here

For each reasoning rule defined in *<rules>* a corresponding class implementing the *ExtractionRule* has to be implemented. The abstract class *ExtractionRule* provides the following methods to be extended:

- *void readConfiguration(string url):* method to parse the *extraction_configuration.xml* and provide information about the attributes and parameters

[6] Link to GitHub repository to be added here

- *boolean applyRule():* defines the core reasoning logic, which evaluates to *true* if the rule applies to the given set of data

The following class attributes are available and have to be set correctly:

- *id:* the unique identifier of the rule
- *keyword:* a name/description of the rule, which can match with the *internalName* specified within the configuration file
- *CPPType:* the CPP type to which the rule should be applied

A template for reading the extraction configuration is available in the already existing rule implementations and can be used for creating new rules.

In order to make CME recognise and use the newly defined rules they first have to registered in *ExtractionOptions*, as seen below. Once the rules are registered here the extraction service will use them at the next module start up.

```
registerRule(new RuleIsDrivingOnHighway(CONFIG_PATH));
registerRule(new RuleIsWeekday(CONFIG_PATH));
registerRule(new RuleIsWeekend(CONFIG_PATH));
registerRule(new RuleIsBusiness(CONFIG_PATH));
registerRule(new RuleIsLeisure(CONFIG_PATH));

... register here the additional rules
```

5.3. Updating the context model and measurement channel list

Before uploading a new context model into CME it has to be ensured that the signal names match those provided within the Cross-CPP Marketplace, otherwise the respective signal may not be recognized by the monitoring service. If the preconditions are met, just copy the new context model (see section 5.1.3 on how to edit and create a context model) into folder *context-monitoring-extraction/resources* and overwrite the existing one (make a backup before). The CME service has to be restarted to recognize the changed context model.

The measurement channel list with the mapping of the measurement channel IDs to the measurement channel names can be updated in the class *MeasurementChannelIdsEnum*. This mapping is being loaded by the context monitoring service during runtime in order to recognize the valid measurement channels to work with. Here also has to be ensured that the names of the measurement channels match with those provided in the context model, otherwise the respective measurement channel may not be recognized by the monitoring and extraction service.

Further customisation is possible by developers as the CME module is provided as an open source code project on GitHub[7] under the EPL 2.0 license. Furthermore, customisation can also be request via [context-support@cross-cpp.eu.](mailto:context-support@cross-cpp.eu)

[7] Link to be added

F.A.Q.

Cross-CPP data-marketplace

Q: What is Cross-CPP data-marketplace?

A: Cross-CPP data-marketplace connects Data Providers and Data Consumers for selling and acquiring Connected Vehicle and Home Building data under the Common Industrial Data model (CIDM). It offers a secure and privacy preserving experience when selling or buying sharing big data, by having the full control over your data shared, to whom and for what purposes.

Cross-CPP offers to cross-sectorial Data Consumers, the possibility to search for more than 200 sensor signals, display advance visualization representations (such as Histograms, Geo-Histograms or Time Series) and retrieve those datasets in a seamless experience thanks to the open SDK-API created.

Q: How do I, as data provider, register into Cross-CPP data-marketplace?

A: You can find the registration form by clicking the "Sign on!" button in the landing page. Select "Original Equipment Manufacturer" role and fill the fields to request your registration. Once your registration is validated by a system administrator an email will be sent to you to confirm your access,

Q: What do I have to do in order to start working with CROSS-CPP data-marketplace?

A: Once registered you must be familiar with the CIDM, as it is the format in which you will receive the data you request.

Cross-CPP data model

Q: What is the Common Industrial Data Model (CIDM)?

A: The CIDM is a standardized data model for industrial data-driven services.

Q: Which are the benefits and advantages of using the CIDM model for data -driven services:

A: -The CIDM constitute a major business and technical advantage for Data Consumers:

- The CIDM provides a brand-independent and transparent data model, which harmonizes proprietary data into generic datasets independently of any cross-sectorial Industry
- It is built on an open and highly scalable automotive big data format (JSON Schema).
- Active community of service providers increasing the number of signals available from vehicles and Smart Buildings to be recorded as well as the type of measurement channels can be modified or extended
- The Data Provider also provides an origin certification as a CIDM feature to support the validation and verification of origin, integrity and completeness of data. The intention is to protect the data inside the Data Package against manipulation.

Q: What is a signal?

A: A signal is the information provider of each CPP. They are the perception organs of CPPs and it is their main duty to detect physical phenomenon and chemical quantities. They observe the environment and generate data in the CIDM format. An example could be "speed" or "latitude"

Q: What is a channel?

A: A channel is the way the physical signals and their sampled measurements are implemented and represented in the CIDM format. Some examples could be "Vehicle Speed" using the signal "Speed" in a time-series or in a histogram format, or "Position" using both "Latitude" and "Longitude" signals.

Q: Can I request a new signal or channel?

A: Cross-CPP data-marketplace offers a wide variety of signals provided by the manufacturers. The catalogue is really extensive and can be filtered in many ways. If even then you can't find the signal that you need and/or think can be provided by any of our data providers, please contact us in: cross-cpp-support@lists.atosresearch.eu.

Cross-CPP marketplace components

Q: What is the Context Monitoring and Extraction module and how can it help me?

A: The Context Monitoring and Extraction module allows Cross-CPP to suggest signals to add to your current Data Discovery filters, based on the context model of the signals already selected. This might help you find data of interest that you would miss otherwise.

Q: Can I add a new context sensitive filter to the discovery process?

A: This is possible by following the steps described in section 5 as the CME module is provided as an open source code project on GitHub[8] under the EPL 2.0 license. Furthermore, customisation can also be request via context-support@cross-cpp.eu.

Q: Can I get more information on how the context is being calculated?

A: In the context sensitive filter within the discovery process each context option is accompanied by a tooltip in the UI. Hovering over it will show the logic behind each context filtering option.

[8] Link to be added

Glossary

Administrator: Cross-CPP marketplace system administrator

AEON: AEON application

AEON application: publication/subscription based communication application

AEON channel: set configuration for communication between two actors through AEON application

CB: Company Backend

CIDM: Common Industrial Data Model

CIDM model: standardized data model for industrial data-driven services

Cloud Storage: Storage system deployed by Cross-CPP solution to store CPP owners data coming from Data Providers

CME: Context Monitoring and Extraction

Company Backend: Data Provider backend system to connect to the Cloud Storage

CPP: cyber-physical product

CPP Data: data created by a CPP and sent to the system by the Data Provider

CPP Owner: Data Owner which CPP is registered in the Cross-CPP data-marketplace

Cross-CPP: System

CS: Cloud Storage

Data Consumer: actor who receives the data created by owners to use it on the creation or improvement of services

Data Owner: owner of the CPP that sends data to the system

Data Provider: OEM that provides its users data to the Cross-CPP marketplace

Data Request: set of configurations that define a scope for CPP Data to be received by a Data Consumer

Id: generic document id string (example: `5cd96b65ff89151c002d16b3`)

Marketplace: Marketplace Web Application

Measurement Channel: sampler of the data the signals process

MP: Marketplace

OEM: Original Equipment Manufacturer

Provider: Data Provider

Service Provider: Data Consumer

Signal: information provider of the data the CPP sensors generate

System: the whole lot of applications that conforms CROSS-CPP, including Marketplace Web Application and Marketplace Server.

UUID: universally unique identifier. Standardized 16 bytes Id signature formed by 32 hexadecimal digits (example: `90eb04b2-a07c-4835-8618-9c0140f8391a`)

Figures

Tables

Listings

Annex

OpenAPI Specification of the Company Backend REST API (yaml)

```
---
swagger: "2.0"
info:
  version: 3.0.0
  title: Cloud Storage Provider API Definition
  contact:
    name: Elisa Herrmann
    url: http://www.cross-cpp.eu
    email: elisa.herrmann@atos.net
host: cloudstorage-api.datagora.eu
basePath: /
schemes:
- https
consumes:
- application/json
produces:
- application/json
security:
- api_key: []
paths:
  /users:
    get:
      tags:
      - User Management
      summary: Retrieve all users
      description: This functionalities provides a list of all users within this Cloud Storage Provider. This API call is not specified and may only be used by the Cloud Storage Provider for internal user management. Returns an array of `User` objects.
      operationId: usersGET
      parameters: []
      responses:
        "200":
          description: OK
          schema:
            type: array
            items:
              $ref: '#/definitions/FullUser'
        "401":
          description: Unauthorized
        "403":
```

```
        description: Forbidden
      "404":
        description: Not found
      default:
        description: Unexpected error
    x-swagger-router-controller: Default
  post:
    tags:
    - User Management
    summary: Create new user
    description: API call registers a new user and creates new Cloud Storage Vault at the Provider. This API call is not specified and is intended to be used for internal user management.
    operationId: usersPOST
    parameters:
    - in: body
      name: User
      description: Defines full name, login and password of the user.
      required: true
      schema:
        $ref: '#/definitions/User'
    responses:
      "200":
        description: OK
        schema:
          $ref: '#/definitions/FullUser'
      "400":
        description: Bad request
      "401":
        description: Unauthorized
      "403":
        description: Forbidden
      "409":
        description: Conflict - User already exists
      default:
        description: Unexpected error
    x-swagger-router-controller: Default
/users/access:
  post:
    tags:
    - Contract Management
    summary: Grant access permission
    description: Acquires access permissions to the `api_key` that is used in this request. If `key`is a `vault-write-key` `write permission`is granted, if `key`is `vault-read-key` `read permission`is granted.
    operationId: usersAccessPOST
    parameters:
```

```
    - in: body
      name: key
      description: (Read or write) Access key to one Cloud Storage Vault.
      required: true
      schema:
        $ref: '#/definitions/key'
    responses:
      "200":
        description: OK
        schema:
          $ref: '#/definitions/inline_response_200'
      "400":
        description: Bad request
      "401":
        description: Unauthorized
      "403":
        description: Forbidden
      "404":
        description: Not found
      default:
        description: Unexpected error
    x-swagger-router-controller: Default
/users/access/{key}:
  get:
    tags:
    - Contract Management
    summary: Validate access token
    description: tbd
    operationId: usersAccessGET
    parameters:
    - name: key
      in: path
      description: Access key for Cloud Storage Vault
      required: true
      type: string
      format: uuid
    responses:
      "200":
        description: OK
        schema:
          $ref: '#/definitions/inline_response_200_1'
    x-swagger-router-controller: Default
  delete:
    tags:
    - Contract Management
    summary: Release access permissions
```

```
      description: Releases access permissions of the `api_key` that is used in this request. If
`key`is a `vault-write-key` `write permission`is released, if `key`is `vault-read-key` `read
permission`is released.
      operationId: usersAccessKeyDELETE
      parameters:
      - name: key
        in: path
        description: Access key for Cloud Storage Vault
        required: true
        type: string
        format: uuid
      responses:
        "200":
          description: OK
        "400":
          description: Bad request
        "401":
          description: Unauthorized
        "403":
          description: Forbidden
        "404":
          description: Not found
        default:
          description: Unexpected error
      x-swagger-router-controller: Default
  /datapackages:
    post:
      tags:
      - Data Storage Interface
      summary: Push data packages into Cloud Storage
      description: This API call enables OEMs to write data packages into the cloud. Data
packages are stored within a CIDM container structure. Every data package needs to contain
the correct vault-id. The OEM needs write permission to the user's Cloud Storage Vault. The
Cloud Storage Provider assignes an unique datapackage-id to every delivered data package.
      operationId: datapackagesPOST
      parameters:
      - in: body
        name: Data Package Container
        description: CIDM container structure containing valid and json encoded CIDM data
packages
        required: true
        schema:
          type: array
          items:
            $ref: '#/definitions/DataPackage'
      responses:
```

```
      "200":
        description: OK
      "400":
        description: Bad request
      "401":
        description: Unauthorized
      "403":
        description: Forbidden
      default:
        description: Unexpected error
    x-swagger-router-controller: Default
/datapackages/{datapackageid}:
  get:
    tags:
    - Data Provisioning Interface
    summary: Get data package
    description: Retrieve one data package with `datapackage-id`.
    operationId: datapackagesIdGET
    parameters:
    - name: datapackageid
      in: path
      description: Unique data package identifier
      required: true
      type: string
    - name: metadata
      in: query
      description: Retrieve *only* metadata, default=false
      required: false
      type: boolean
      default: false
    responses:
      "200":
        description: OK
        schema:
          $ref: '#/definitions/DataPackage'
      "400":
        description: Bad request
      "401":
        description: Unauthorized
      "403":
        description: Forbidden
      "404":
        description: Not found
      default:
        description: Unexpected error
    x-swagger-router-controller: Default
```

```
 /datapackages/query:
   post:
     tags:
     - Data Provisioning Interface
     summary: Query data packages
     description: Allows searching for data packages. Cloud Storage Provider will only include
Cloud Storage Vaults into search where `api_key` has read access.
     operationId: datapackagesQueryPOST
     parameters:
     - in: body
       name: query
       description: Query for data packages
       required: true
       schema:
         $ref: '#/definitions/Query'
     responses:
       "200":
         description: OK
         schema:
           type: array
           items:
             $ref: '#/definitions/DataPackage'
       "400":
         description: Bad request
       "401":
         description: Unauthorized
       "403":
         description: Forbidden
       "404":
         description: Not found
       default:
         description: Unexpected error
     x-swagger-router-controller: Default
 /datapackages/query_stream:
   post:
     tags:
     - Data Provisioning Interface
     summary: Query data packages
     description: Allows searching for data packages. Cloud Storage Provider will only include
Cloud Storage Vaults into search where `api_key` has read access.
     operationId: datapackagesQueryStreamPOST
     parameters:
     - in: body
       name: query
       description: Query for data packages
       required: true
```

```
        schema:
          $ref: '#/definitions/Query'
      responses:
        "200":
          description: OK
          schema:
            type: array
            items:
              $ref: '#/definitions/DataPackage'
        "400":
          description: Bad request
        "401":
          description: Unauthorized
        "403":
          description: Forbidden
        "404":
          description: Not found
        default:
          description: Unexpected error
      x-swagger-router-controller: Default
  /datapackages/basicCppQuery:
    post:
      tags:
      - Data Provisioning Interface
      summary: Query basic CPP information data packages
      description: Allows searching for data packages. Cloud Storage Provider will only include
Cloud Storage Vaults into search where `api_key` has read access.
      operationId: basicCppQueryPOST
      parameters:
      - in: body
        name: query
        description: Query for data packages
        required: true
        schema:
          $ref: '#/definitions/Query'
      responses:
        "200":
          description: OK
          schema:
            type: array
            items:
              $ref: '#/definitions/DataPackage'
        "400":
          description: Bad request
        "401":
          description: Unauthorized
        "403":
          description: Forbidden
        "404":
          description: Not found
        default:
          description: Unexpected error
      x-swagger-router-controller: Default
  /datapackages/basicCppQuery_stream:
    post:
      tags:
      - Data Provisioning Interface
      summary: Query basic CPP information data packages
      description: Allows searching for data packages. Cloud Storage Provider will only include
Cloud Storage Vaults into search where `api_key` has read access.
      operationId: basicCppQueryStreamPOST
      parameters:
      - in: body
        name: query
        description: Query for data packages
        required: true
        schema:
          $ref: '#/definitions/Query'
      responses:
        "200":
          description: OK
          schema:
            type: array
            items:
              $ref: '#/definitions/DataPackage'
        "400":
          description: Bad request
        "401":
          description: Unauthorized
        "403":
          description: Forbidden
        "404":
          description: Not found
        default:
          description: Unexpected error
      x-swagger-router-controller: Default
  /notifications:
    post:
      tags:
      - Notifications
      summary: Subscribe Push Notification
      description: Functionality of the marketplace to subscribe to push notification events. Push notifications are sent, whenever users put data into their Cloud Storage Vaults.
      operationId: notificationsPOST
      parameters:
      - in: body
        name: config
        description: Push Notification URL
        required: true
        schema:
          $ref: '#/definitions/config'
      responses:
        "200":
          description: OK
        "400":
          description: Bad request
        "401":
          description: Unauthorized
        "403":
          description: Forbidden
        default:
          description: Unexpected error
      x-swagger-router-controller: Default
    delete:
      tags:
      - Notifications
      summary: Unsubscribe Push Notification
      description: No more push notifications will be sent from the Cloud Storage Provider to the marketplace.
      operationId: notificationsDELETE
      parameters: []
      responses:
        "200":
          description: OK
        "401":
          description: Unauthorized
        "403":
          description: Forbidden
        "404":
          description: Not found
        default:
          description: Unexpected error
      x-swagger-router-controller: Default
  /access:
    get:
      tags:
      - Contract Management
      summary: Validate authentication token
      description: t.b.d.
      operationId: accessGET
      parameters: []
      responses:
        "200":
          description: OK
          schema:
            $ref: '#/definitions/inline_response_200_2'
        "403":
          description: Forbidden / No authorization token in header
        "404":
          description: Authorization token not found!
        "500":
          description: Internal Error
      x-swagger-router-controller: Default
securityDefinitions:
  api_key:
    description: Provides authentification for OEM and Marketplace. Must be sent in HTTP header.
    type: apiKey
    name: Authentication
    in: header
definitions:
  FullUser:
    allOf:
    - $ref: '#/definitions/User'
    - {}
  User:
    type: object
    required:
    - full-name
    - login-name
    - password
    properties:
      full-name:
        type: string
        description: Full name of the user
      login-name:
        type: string
        format: email
        description: Login name of the user (e-mail)
      password:
        type: string
        description: Password of the user
    example:
      password: password
      full-name: full-name
      login-name: login-name
  DataPackage:
    type: object
    required:
    - cpp-type
    - cvim-version
    - data
    - measurement-channel-id
    - type
    - vault-id
    properties:
      vault-id:
        type: string
        description: Cloud Storage Vault ID, where the data packages are pushed into.
      datapackage-id:
        type: string
        description: Unique identifier of the data package. Property is set by Cloud Storage
Provider.
        readOnly: true
      data:
        type: object
        description: CIDM data
        properties: {}
      cvim-version:
        type: string
        description: Version of the CIDM protocol >=1.1.2
      type:
        type: string
        description: Type of the Data Package
      cpp-type:
        type: string
        description: Type of the CPP device, "vehicle" or "builging"
      cpp-id:
        type: string
        description: CPP ID for identifying vehicles or bulding of the same owner and vault-id
      measurement-channel-id:
        type: string
        description: Identifier of the Measurement Channel whose data is inside this data package
      timestamp:
        type: string
        description: Measurement timestamp (basic-cpp-information and event-based)
      timestamp-start:
        type: string
        description: Measurement start time
      timestamp-stop:
        type: string
        description: Measurement stop time
      mileage-start:
        type: number
        description: Mileage at the start of measurement in kilometres (km)
      mileage-stop:
        type: number
        description: Mileage at the end of measurement in kilometres (km)
      geo-bounding-box:
        type: object
        description: geographic bounding box (see reference manual section 6.5.1.1)
        properties: {}
      room-id:
        type: string
        description: Identifier of the room in a building cpp-type
      oem-certification:
        type: object
        description: OEM certification (see reference manual section 6.5.1.2)
        properties: {}
      ownership-information:
        type: object
        description: Data Ownership Information (see reference manual section 6.5.1.3)
        properties: {}
      expiration-date:
        type: string
        description: Data expiration date
      data-masking-active:
        type: boolean
        description: Indicates status of data-masking (true = active)
    example:
      datapackage-id: datapackage-id
      vault-id: vault-id
      data: '{}'
      data-masking-active: true
      cvim-version: cvim-version
      type: type
      mileage-stop: 6.027456183070403
      measurement-channel-id: measurement-channel-id
      timestamp-start: timestamp-start
      geo-bounding-box: '{}'
      mileage-start: 0.8008281904610115
      oem-certification: '{}'
      ownership-information: '{}'
      timestamp-stop: timestamp-stop
      expiration-date: expiration-date
```

```
Query:
  type: object
  properties:
    datapackage-id:
      type: array
      description: Array of Data Package IDs
      items:
        type: string
        description: Data Package ID
    measurement-channel-id:
      type: array
      description: Array of Measurement Channel IDs
      items:
        type: string
        description: Measurement Channel ID
    vault-id:
      type: array
      description: Array of Cloud Storage Vault IDs
      items:
        type: string
    submit-time:
      $ref: '#/definitions/Query_submittime'
    metadata:
      type: boolean
      description: Request only metadata, default=off
      default: false
  example:
    datapackage-id:
    - datapackage-id
    - datapackage-id
    submit-time:
      min: 2000-01-23T04:56:07.000+00:00
      max: 2000-01-23T04:56:07.000+00:00
    metadata: false
    vault-id:
    - vault-id
    - vault-id
    measurement-channel-id:
    - measurement-channel-id
    - measurement-channel-id
key:
  type: object
  properties:
    vault-access-key:
      type: string
      format: uuid
      description: Access key for Cloud Storage Vault
config:
  type: object
  properties:
    handler-url:
      type: string
      format: uuid
      description: URL for push notifications
    level:
      type: string
      description: Defines the level of the notification ('id-only', 'metadata' or 'full')
Query_submittime:
  properties:
    min:
      type: string
      format: date-time
      description: Earliest Data Package submission time
    max:
      type: string
      format: date-time
      description: Latest Data Package submission time
  description: Data Package submission time
  example:
    min: 2000-01-23T04:56:07.000+00:00
    max: 2000-01-23T04:56:07.000+00:00
inline_response_200:
  type: object
  properties:
    full-name:
      type: string
    vault-id:
      type: string
      description: ID of the user's Cloud Storage Vault.
inline_response_200_1:
  type: object
  properties:
    full-name:
      type: string
    vault-id:
      type: string
      format: uuid
      description: ID of the user's Cloud Storage Vault.
    type:
      type: string
      description: '`read` or `write` access key'
    in-use:
      type: boolean
      description: '`true` when key is already in use, otherwise `false`'
  inline_response_200_2:
    type: object
    properties:
      name:
        type: string
      type:
        type: string
```

CIDM v1.2.1 jsonSchema

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "description": "Common Industrial Data Model",
    "type": "object",
    "properties": {
        "Signal": {
            "$ref": "#/definitions/Signal"
        },
        "MeasurementChannel": {
            "$ref": "#/definitions/MeasurementChannel"
        },
        "DataPackage": {
            "$ref": "#/definitions/DataPackage"
        }
    },
    "additionalProperties": false,
    "definitions": {
        "TimeSeriesMeasurementChannel": {
            "title": "TimeSeriesChannel",
            "type": "object",
            "properties": {
                "type": {
                    "type": "string",
                    "enum": [
                        "time-series"
                    ]
                },
                "capture-interval": {
                    "type": "number"
                },
                "on-change": {
                    "type": "boolean"
                },
                "sample-strategy": {
                    "type": "string",
                    "enum": [
                        "min",
                        "max",
                        "average",
                        "last-known-value"
                    ]
                },
                "signal": {
                    "$ref": "#/definitions/Signal"
                },
                "format": {
                    "type": "string"
                },
                "dimension": {
                    "type": "number"
                }
            },
            "required": [
                "type",
                "capture-interval",
                "on-change",
                "sample-strategy",
                "signal"
            ]
        },
        "MeasurementChannel": {
            "type": "object",
            "properties": {
                "id": {
                    "type": "string"
                },
                "name": {
                    "type": "string"
                },
                "type": {
                    "type": "string",
                    "enum": [
                        "time-series",
                        "histogram",
                        "geo-histogram",
                        "general-purpose",
                        "event-based",
                        "basic-cpp-information"
                    ]
                },
                "comment": {
                    "type": "string"
                }
            },
            "required": [
                "id",
                "name",
                "type"
            ],
            "oneOf": [
                {
                    "$ref": "#/definitions/TimeSeriesMeasurementChannel"
                },
                {
                    "$ref": "#/definitions/HistogramMeasurementChannel"
                },
                {
                    "$ref": "#/definitions/GeoBasedHistogramMeasurementChannel"
                },
                {
                    "$ref": "#/definitions/GeneralPurposeMeasurementChannel"
                },
                {
                    "$ref": "#/definitions/BasicCppInformationMeasurementChannel"
                },
                {
                    "$ref": "#/definitions/EventBasedMeasurementChannel"
                }
            ]
        },
        "HistogramMeasurementChannel": {
            "type": "object",
            "properties": {
                "type": {
                    "type": "string",
                    "enum": [
                        "histogram",
                        "geo-histogram"
                    ]
                },
                "aggregation-strategy": {
                    "type": "string",
                    "enum": [
                        "time",
                        "count",
                        "min",
                        "max"
                    ]
                },
                "capture-interval": {
                    "type": "number"
                },
                "dimensions": {
                    "type": "integer",
                    "minimum": 1
                },
                "bins": {
                    "type": "array",
                    "minItems": 1,
                    "items": {
                        "type": "object",
                        "properties": {
                            "type": {
                                "type": "string",
                                "enum": [
                                    "linear",
                                    "logarithmic",
                                    "custom"
                                ]
                            },
                            "lower-bound": {
                                "type": "number"
                            },
                            "upper-bound": {
                                "type": "number"
                            },
                            "signal": {
                                "$ref": "#/definitions/Signal"
                            },
                            "number-of-bins": {
                                "type": "integer",
                                "minimum": 0
                            },
                            "alternative-bin-labels": {
                                "type": "array",
                                "items": {
                                    "type": "string"
                                }
                            }
                        },
                        "required": [
                            "type",
                            "lower-bound",
                            "upper-bound",
                            "signal",
                            "number-of-bins"
                        ],
                        "oneOf": [
                            {
                                "type": "object",
                                "properties": {
                                    "type": {
                                        "type": "string",
                                        "enum": [
                                            "linear",
                                            "logarithmic"
                                        ]
                                    }
                                },
                                "required": [
                                    "type"
                                ]
                            },
                            {
                                "type": "object",
                                "properties": {
                                    "type": {
                                        "type": "string",
                                        "enum": [
                                            "custom"
                                        ]
                                    },
                                    "custom-bounds": {
                                        "type": "array",
                                        "items": {
                                            "type": "number"
                                        }
                                    }
                                },
                                "required": [
                                    "type",
                                    "custom-bounds"
                                ]
```

```
                    }
                ]
            }
        }
    },
    "required": [
        "type",
        "aggregation-strategy",
        "capture-interval",
        "dimensions",
        "bins"
    ],
    "additionalProperties": false
},
"GeoBasedHistogramMeasurementChannel": {
    "type": "object",
    "properties": {
        "type": {
            "type": "string",
            "enum": [
                "geo-histogram"
            ]
        },
        "geo-resolution": {
            "type": "number"
        }
    },
    "required": [
        "type",
        "geo-resolution"
    ],
    "additionalProperties": false,
    "allOf": [
        {
            "$ref": "#/definitions/HistogramMeasurementChannel"
        }
    ]
},
"GeneralPurposeMeasurementChannel": {
    "type": "object",
    "properties": {
        "type": {
            "type": "string",
            "enum": [
                "general-purpose"
            ]
        },
        "signal": {
            "$ref": "#/definitions/Signal"
        }
    },
    "required": [
        "type",
        "signal"
    ],
    "additionalProperties": false
},
"BasicCppInformationMeasurementChannel": {
    "type": "object",
    "properties": {
        "type": {
            "type": "string",
            "enum": [
                "basic-cpp-information"
            ]
        },
        "signal": {
            "$ref": "#/definitions/Signal"
        }
    },
    "required": [
        "type",
        "signal"
    ],
    "additionalProperties": false
},
"EventBasedMeasurementChannel": {
    "type": "object",
    "properties": {
        "type": {
            "type": "string",
            "enum": [
                "event-based"
            ]
        },
        "format": {
            "type": "string"
        },
        "event-sample-strategy": {
            "type": "string",
            "enum": [
                "real-time-event",
                "trigger-event",
                "threshold-event"
            ]
        },
        "comment": {
            "type": "string"
        }
    },
    "required": [
        "type",
        "signal",
        "event-sample-strategy"
    ],
    "additionalProperties": false
},
"DataPackage": {
    "type": "object",
    "properties": {
        "datapackage-id": {
            "type": "string"
        },
        "vault-id": {
```

```
        "type": "string"
    },
    "trip-id": {
        "type": "string"
    },
    "cpp-id": {
        "type": "string"
    },
    "cvim-version": {
        "type": "string",
        "enum": [
            "1.0.0",
            "1.0.1",
            "1.2.0",
            "1.2.1"
        ]
    },
    "cpp-type": {
        "type": "string",
        "enum": [
            "vehicle",
            "building"
        ]
    },
    "type": {
        "type": "string",
        "enum": [
            "time-series",
            "histogram",
            "geo-histogram",
            "general-purpose",
            "event-based",
            "basic-cpp-information"
        ]
    },
    "measurement-channel-id": {
        "type": "string"
    },
    "submit-time": {
        "type": "string",
        "format": "date-time"
    },
    "mileage-start": {
        "type": "number"
    },
    "room-id": {
        "type": "string"
    },
    "mileage-stop": {
        "type": "number"
    },
    "geo-bounding-box": {
        "type": "object",
        "properties": {
            "latitude-min": {
                "type": "number"
            },
                "latitude-max": {
                    "type": "number"
                },
                "longitude-min": {
                    "type": "number"
                },
                "longitude-max": {
                    "type": "number"
                },
                "altitude-min": {
                    "type": "number"
                },
                "altitude-max": {
                    "type": "number"
                }
            },
            "additionalProperties": false
        },
        "location": {
            "type": "object",
            "properties": {
                "latitude": {
                    "type": "number"
                },
                "longitude": {
                    "type": "number"
                }
            },
            "additionalProperties": false
        },
        "oem-certification": {
            "type": "object",
            "properties": {
                "signature": {},
                "checksum": {},
                "sequence-number": {}
            },
            "additionalProperties": false
        },
        "expiration-date": {
            "type": "string",
            "format": "date-time"
        },
        "data-ownership-information": {
            "type": "object",
            "properties": {
                "privacy-veto-rights": {
                    "type": "object",
                    "properties": {
                        "consent-level": {
                            "type": "string",
                            "enum": [
                                "public",
                                "shared",
                                "private"
                            ]
                        },
                    "data-format": {
                        "type": "string",
                        "enum": [
                            "time-series",
                            "histogram"
                        ]
                    },
                    "jurisdiction": {
                        "type": "string",
                        "enum": [
                            "Europe",
                            "any"
                        ]
                    },
                    "storage-constraint": {
                        "type": "string",
                        "enum": [
                            "OEM storage",
                            "Personal storage"
                        ]
                    }
                },
                "required": [
                    "consent-level"
                ]
            },
            "copyright-stakeholders": {
                "type": "array",
                "items": [
                    {
                        "type": "object",
                        "properties": {
                            "name": {
                                "type": "string"
                            },
                            "status": {
                                "type": "string"
                            }
                        },
                        "required": [
                            "name",
                            "status"
                        ],
                        "additionalProperties": false
                    }
                ]
            },
            "data-stakeholders": {
                "type": "array",
                "items": [
                    {
                        "type": "object",
                        "properties": {
                            "name": {
                                "type": "string"
                            },
                            "status": {
                                "type": "string"
                            }
                        },
                        "required": [
                            "name",
                            "status"
                        ],
                        "additionalProperties": false
                    }
                ]
            },
            "data-privacy-level": {
                "type": "string",
                "enum": [
                    "public",
                    "shared",
                    "private"
                ]
            }
        },
        "additionalProperties": false
    },
    "data-masking-active": {
        "type": "boolean"
    },
    "signatures": {
        "type": "array",
        "items": [
            {
                "type": "object",
                "properties": {
                    "signatory": {
                        "type": "string"
                    },
                    "checksum": {
                        "type": "string"
                    },
                    "signature": {
                        "type": "string"
                    }
                },
                "required": [
                    "signatory",
                    "checksum",
                    "signature"
                ],
                "additionalProperties": false
            }
        ]
    }
},
"required": [
    "cvim-version",
    "type",
    "measurement-channel-id",
    "vault-id",
    "cpp-type"
```

```
],
"oneOf": [
    {
        "type": "object",
        "properties": {
            "type": {
                "type": "string",
                "enum": [
                    "time-series"
                ]
            },
            "number-of-samples": {
                "type": "integer",
                "minimum": 1
            },
            "statistic-properties": {
                "type": "object",
                "properties": {
                    "min": {
                        "type": "number"
                    },
                    "max": {
                        "type": "number"
                    },
                    "average": {
                        "type": "number"
                    },
                    "histogram": {
                        "type": "object",
                        "properties": {
                            "measurement-channel-id": {
                                "type": "string"
                            },
                            "data": {
                                "type": "array",
                                "minItems": 0,
                                "maxItems": 10,
                                "items": {
                                    "type": "number"
                                }
                            }
                        },
                        "required": [
                            "measurement-channel-id",
                            "data"
                        ]
                    }
                }
            },
            "data": {
                "type": "array",
                "minItems": 1,
                "items": {
                    "type": "object",
                    "properties": {
                        "timestamp": {
                            "type": "string",
                            "format": "date-time"
                        },
                        "value": {
                            "type": [
                                "array",
                                "string",
                                "number",
                                "boolean"
                            ],
                            "items": {
                                "type": [
                                    "string",
                                    "number",
                                    "boolean"
                                ],
                                "minLength": 1
                            }
                        }
                    }
                }
            },
            "timestamp-start": {
                "type": "string",
                "format": "date-time"
            },
            "timestamp-stop": {
                "type": "string",
                "format": "date-time"
            }
        },
        "required": [
            "type",
            "number-of-samples",
            "data",
            "timestamp-start",
            "timestamp-stop"
        ]
    },
    {
        "type": "object",
        "properties": {
            "type": {
                "type": "string",
                "enum": [
                    "histogram",
                    "geo-histogram"
                ]
            },
            "data": {
                "type": "array",
                "items": {
                    "type": [
                        "number",
                        "array"
                    ],
                    "minItems": 1,
                    "items": {
                        "type": "number"
                    }
                }
            },
            "timestamp-start": {
                "type": "string",
                "format": "date-time"
            },
            "timestamp-stop": {
                "type": "string",
                "format": "date-time"
            }
        },
        "required": [
            "type",
            "data",
            "timestamp-start",
            "timestamp-stop"
        ]
    },
    {
        "type": "object",
        "properties": {
            "type": {
                "type": "string",
                "enum": [
                    "general-purpose",
                    "basic-cpp-information"
                ]
            },
            "data": {},
            "timestamp": {
                "type": "string",
                "format": "date-time"
            }
        },
        "required": [
            "type",
            "data",
            "timestamp"
        ]
    },
    {
        "type": "object",
        "properties": {
            "type": {
                "type": "string",
                "enum": [
                    "event-based"
                ]
            },
            "data": {
                "type": "object",
                "properties": {
                    "event-type": {
                        "type": "string",
                        "enum": [
                                        "real-time-event",
                                        "trigger-event",
                                        "threshold-event"
                                    ]
                                },
                                "event-data": {
                                    "type": "object",
                                    "properties": {
                                        "value": {
                                            "type": "string"
                                        },
                                        "event-datapackages": {
                                            "type": "array",
                                            "minItems": 1,
                                            "items": {
                                                "$ref": "#/definitions/DataPackage"
                                            }
                                        }
                                    },
                                    "required": [
                                        "value"
                                    ]
                                }
                            },
                            "required": [
                                "event-type",
                                "event-data"
                            ]
                        },
                        "timestamp": {
                            "type": "string",
                            "format": "date-time"
                        }
                    },
                    "required": [
                        "type",
                        "data",
                        "timestamp"
                    ]
                }
            ]
        },
        "Signal": {
            "title": "CIDM Signal",
            "description": "Signals are the perception organs of vehicles. It is their ma
in duty to detect physical phenomenons and chemical quantities by transferring them into
electrical signals. They observe the environment and gener-
ate the data that is exchangeable at AutoMat’s marketplace. They are one of the core comp
onents of the AutoMat project. Figure 11 shows the UML modelling of the signals. Within A
utoMat all information pro-
viders are modelled as Signal. They can be classified as static signals or changing/non-
static signals, having a sample rate larger than zero. ",
            "type": "object",
            "properties": {
                "id": {
                    "type": "string"
                },
                "name": {
                    "type": "string"
                },
                "cpp-type": {
                    "type": "string",
                    "enum": [
                        "vehicle",
                        "building"
                    ]
                },
                "type": {
                    "type": "string",
                    "enum": [
                        "numeric",
                        "information",
                        "enumeration",
                        "general-purpose"
                    ]
                },
                "format": {
                    "type": "string"
                },
                "sample-rate": {
                    "type": "number",
                    "minimum": 0
                },
                "comment": {
                    "type": "string"
                }
            },
            "required": [
                "id",
                "name",
                "type",
                "sample-rate",
                "cpp-type"
            ],
            "oneOf": [
                {
                    "$ref": "#/definitions/NumericSignal"
                },
                {
                    "$ref": "#/definitions/EnumerationSignal"
                },
                {
                    "$ref": "#/definitions/InformationSignal"
                },
                {
                    "$ref": "#/definitions/GeneralPurposeSignal"
                }
            ]
        },
        "GeneralPurposeSignal": {
            "type": "object",
            "properties": {
                "type": {
                    "type": "string",
                "enum": [
                    "general-purpose"
                ]
            }
        },
        "required": [
            "type"
        ]
    },
    "InformationSignal": {
        "type": "object",
        "properties": {
            "type": {
                "type": "string",
                "enum": [
                    "information"
                ]
            },
            "format": {
                "type": "string"
            }
        },
        "required": [
            "type",
            "format"
        ]
    },
    "EnumerationSignal": {
        "type": "object",
        "properties": {
            "type": {
                "type": "string",
                "enum": [
                    "enumeration"
                ]
            },
            "items": {
                "type": "array",
                "items": {
                    "type": "string"
                }
            }
        },
        "required": [
            "type",
            "items"
        ]
    },
    "NumericSignal": {
        "type": "object",
        "properties": {
            "type": {
                "type": "string",
                "enum": [
                    "numeric"
                ]
            },
                "format": {
                    "type": "string",
                    "enum": [
                        "int",
                        "uint",
                        "float",
                        "double"
                    ]
                },
                "min": {
                    "type": "number"
                },
                "max": {
                    "type": "number"
                },
                "resolution": {
                    "type": "number"
                },
                "unit": {
                    "type": "string"
                }
            },
            "required": [
                "type",
                "format",
                "min",
                "max",
                "resolution",
                "unit"
            ]
        }
    }
}
```

About Cross-CPP

The objective is to establish an IT environment for the integration and analytics of data streams coming from high volume (mass) products with cyber physical features, as well from Open Data Sources, aiming to offer new cross sectorial services and focusing on the commercial confidentiality, privacy and IPR and ethical issues using a context sensitive approach. The project addresses cross-stream analysis of large data volumes from mass cyber physical products (CPP) from various industrial sectors such as automotive, and home automation. The business objective of the research is to allow for analyses of such data streams in combination to other (non-industrial, open) data streams and for the establishment of diverse enhanced sectorial and cross-sectorial services. The project will develop: (i) New models for integration and analytics of data streams coming from multi-sectorial CPP, including shared systems of entity identifiers applicable to multi-sectorial CPP (as well as the definition of agreed data models for data streams from multiple CPP aiming at defacto standard; (ii) Ecosystem, including a common Marketplace, and methodology to use such models to build multi-sectorial cloud based services, (iii) Toolbox for real-time and predictive cross-stream analytics, context modelling and extraction, and dynamically changing security policy, privacy and IPR conditions/rules and (iv) set of services such as services based on a combination of data streams from home automation and (electrical) vehicles to pro-vide enhanced local weather forecast and predict and optimise energy consumptions in households. The project will build upon the results from past and current projects, where results from the project AutoMat, addressing services developed based on data streams from vehicles, will be used as a basis for further development aiming to extend it to integrated, cross-sectorial data streams analytics. More information is available at https://cross-cpp.eu

Funded by the Horizon 2020 Framework Programme of the European Union