



CROSS-**CPP**

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



CROSS-**CPP**

Cross-CPP MARKETPLACE DEVELOPER GUIDE

CPP DATA CONSUMER GUIDE
(CROSS-SECTORIAL SERVICE PROVIDER)



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

Introduction

Service Provider Developer Guide offers data consumers the information needed to develop a backend system to start working with Cross-CPP Marketplace data.

Purpose

This guide aims to help developers from Data Consumers companies (from now on called Service Providers) on how to develop a backend system capable of working with Cross-CPP marketplace data and give knowledge about the different functionalities available.

Audience

This guide is meant for and solely for developers Service Provider companies that wants to work with Cross-CPP marketplace.

Scope

The contents of this guide are meant to be taken into consideration only when developing a backend system 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.

If you find there is no content in this guide for some functionality you can request it through: cross-cpp-support@lists.atosresearch.eu.

Troubleshooting

For any questions or inquiries about the use of the Cross-CPP Marketplace API or AEON, 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, 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

Introduction.....	1
Purpose.....	1
Audience.....	1
Scope.....	1
Troubleshooting.....	1
Contact	1
Contents	2
Guide 7	
1. General Workflow of Marketplace Usage	7
1.1. Registration at Marketplace.....	8
1.2. Search for Shared Data (Data Discovery).....	10
1.2.1. Discovery Definition	10
1.2.2. Discovery Results	13
1.3. Data Request Creation.....	14
1.4. Data Retrieval.....	17
1.4.1. Pull Approach Data Retrieval.....	17
1.4.1.1. Metadata.....	18
1.4.1.2. Data Packages.....	18
1.4.1.3. Query Parameters.....	20
1.4.1.4. Last value query.....	21
1.4.2. Push Approach Data Retrieval.....	21
2. Toolbox Guide.....	24
2.1. General Workflow of Toolbox Usage	24
2.2. Data Views.....	24
2.2.1. Categories	25
2.3. Analytics	25
2.3.1. Time Series Analytics.....	25
2.3.2. Trajectory Analysis.....	27
2.3.3. Networks.....	29
2.3.4. Machine Learning.....	31
2.3.4.1. Initial machine-learning processing function.....	32

2.3.4.1.1.	Create (build) machine-learning model.....	32
2.3.4.2.	Common control functions.....	32
2.3.4.2.1.1.	Status of process.....	32
2.3.4.2.2.	Kill a process.....	33
2.3.4.2.3.	Drop a process.....	33
2.3.4.2.4.	Export model.....	34
2.3.4.2.5.	Import model	35
2.3.4.3.	Main machine-learning processing functions.....	36
2.3.4.3.1.	Update machine-learning model	36
2.3.4.3.2.	Apply machine-learning model to testing data	37
2.3.4.3.3.	Evaluate machine-learning model to evaluation data.....	39
3.	API.....	41
3.1.	First Steps.....	41
3.1.1.	Authentication.....	41
3.1.2.	Get Access Token.....	42
3.1.3.	Service Provider Profile	43
3.2.	General Purpose API.....	44
3.2.1.	Catalogue.....	44
3.2.1.1.	Signal Catalogue	44
3.2.1.2.	Signal	45
3.2.1.3.	Signals.....	46
3.2.1.4.	Channel.....	47
3.2.1.5.	Channels.....	48
3.2.2.	Data Discovery.....	49
3.2.3.	Channel Suggestions	53
3.3.	Service Provider API.....	54
3.3.1.	Data Requests	54
3.3.1.1.	Create Data Request.....	54
3.3.1.2.	Delete Data Request.....	56
3.3.1.3.	Get my Data Requests.....	57
3.3.1.4.	Get a Data Request.....	58
3.3.1.5.	Get Data Request Contracts	59
3.3.2.	Data	60

3.3.2.1.	Get Data Transactions	60
3.3.2.2.	Get Data Request Data Transactions	61
3.3.2.3.	Get Data Package Details	62
3.3.2.4.	Get Data Request received Data Packages.....	64
3.3.2.5.	Get Data Request received Metadata Packages	65
3.4.	Toolbox API	66
3.4.1.	Data Views.....	66
3.4.1.1.	Create Data View	67
3.4.1.2.	Get Data Views list.....	68
3.4.1.3.	Get Data View details.....	68
3.4.1.4.	Retrieve Data View	69
3.4.1.5.	Data View category	70
3.4.1.5.1.	Create new category	70
3.4.1.5.2.	Get categories list.....	71
3.4.1.5.3.	Get Category details	72
3.4.1.5.4.	Assign Data View category.....	73
3.4.1.5.5.	Assign category values	73
3.4.2.	Analytics.....	74
3.4.2.1.	Time Series Analytics	75
3.4.2.1.1.	Create new Time Series Analytics.....	75
3.4.2.1.1.1.	SampleEntropy.....	76
3.4.2.1.1.2.	PermutationEntropy.....	77
3.4.2.1.1.3.	Irreversibility	77
3.4.2.1.1.4.	PearsonCorrelation.....	77
3.4.2.1.1.5.	SpearmanCorrelation	78
3.4.2.1.1.6.	RegressionTree	78
3.4.2.1.1.7.	NeuralNetwork.....	78
3.4.2.1.1.8.	Arima	79
3.4.2.1.2.	Get Time Series Analytics list.....	79
3.4.2.1.3.	Get Time Series Analytics details	80
3.4.2.1.4.	Delete Time Series Analytics details	81
3.4.2.2.	Trajectory Analysis.....	82
3.4.2.2.1.	Create new Trajectory Analytics	82

3.4.2.2.1.1.	Statistics.....	83
3.4.2.2.1.2.	Interpolation.....	84
3.4.2.2.1.3.	Clustering.....	84
3.4.2.2.1.4.	Anomaly Detection (anomaly_detection)	84
3.4.2.2.2.	Get Trajectory Analysis list.....	85
3.4.2.2.3.	Get Trajectory Analysis details.....	85
3.4.2.2.4.	Get Trajectory Analysis results.....	86
3.4.2.2.4.1.	Statistics	87
3.4.2.2.4.2.	Interpolation	87
3.4.2.2.4.3.	Clustering.....	88
3.4.2.2.4.4.	Anomaly Detection.....	88
3.4.2.3.	Networks	88
3.4.2.3.1.	Create new Network	88
3.4.2.3.2.	Get Network list	89
3.4.2.3.3.	Get Network.....	90
3.4.2.3.4.	Get Network status.....	91
3.4.2.4.	Machine Learning.....	92
3.4.2.4.1.	Create ML model.....	92
3.4.2.4.2.	Get ML model list.....	93
3.4.2.4.3.	Get ML model details	94
3.4.2.4.4.	Export ML model.....	95
3.4.2.4.5.	Get ML model status.....	96
3.4.2.4.6.	Evaluate ML model	97
3.4.2.4.7.	Apply ML model.....	97
3.4.2.4.8.	Kill ML model.....	98
3.4.2.4.9.	Delete ML model.....	99
4.	AEON.....	100
4.1.	What is AEON	100
4.2.	Configuring AEON channels.....	101
4.3.	Subscribing to AEON.....	101
5.	Context Monitoring and Extraction (CME) guide.....	102
5.1.	Extraction rules configuration file.....	102
5.2.	Source code customisation.....	105

FAQ	107
Cross-CPP data-marketplace	107
Cross-CPP data model	107
Cross-CPP marketplace components.....	108
Data Requests	108
Toolbox	109
AEON.....	111
Glossary.....	112
Figures.....	114
Tables	114
Lists	116

Guide

The Cross-CPP Marketplace offers five distinct sections for data consumers:

- Cross-CPP Marketplace workflow: this section describes the main steps a Service provider needs to perform in order to consume shared data.
- Cross-CPP Marketplace Analytics Toolbox guide: complete guide of all available analytics functions to analyse and work with collected data.
- Cross-CPP Marketplace API: is considered as the customer central point where they can query the available data and the company current requests.
- AEON: bus driven data communication application through which the Cross-CPP Marketplace data is sent for currently active data requests and analytics.
- Context Monitoring and Extraction (CME): for data consumers that want to customise the CME Reasoning Rules to provide needed filters.

1. General Workflow of Marketplace Usage

The main purpose of the Marketplace is to share smart buildings and connected vehicle data and to provide these shared data for creating new added value services. This section objective is to provide an insightful understanding, of how data can be retrieved by Service Providers.

The following figure (Figure 1) shows the Cross-CPP Marketplace generic usage flow¹. First, a registration at the Marketplace has to be performed in order to identify the Service Provider user. Without registration no access to the functionalities is granted. After the registration, the first step is the discovery of shared data. When enough data is available or the search matches all needs, an offer can be created. Different types of services require different types of data. Therefore, multiple offers can be created. Cross-CPP empowers the data owner users to have control over their data. They need to agree that a Service Provider may have access to their data. In the next step, the Service Provider needs to wait for data owner user's agreements. After users commit to share the data, it can be retrieved by the service provider using the Marketplace API or the SDK for data subscription.

¹ This workflow does not include the use of the toolbox or CME (see sections **Fehler! Verweisquelle konnte nicht gefunden werden.** and **Fehler! Verweisquelle konnte nicht gefunden werden.** for more information)

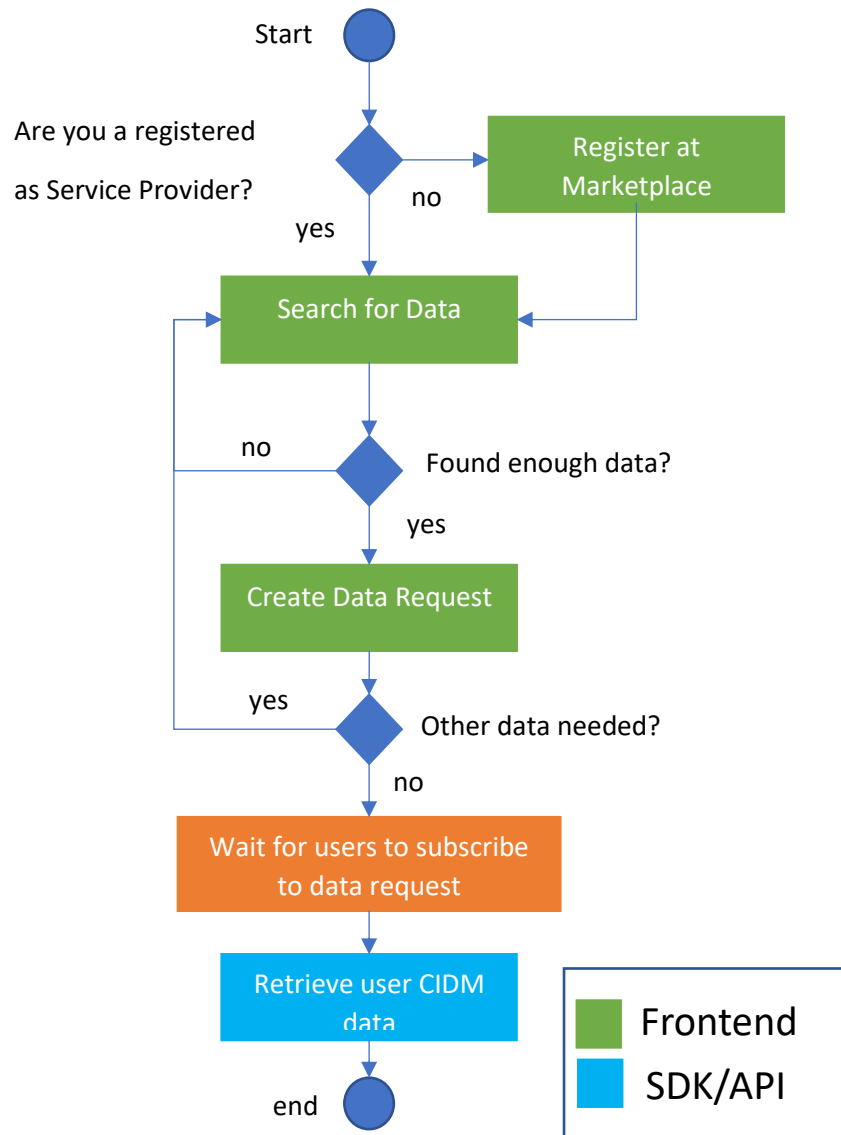




Figure 1. Cross-CPP Marketplace general workflow

1.1. Registration at Marketplace

In order to identify towards the Marketplace, Service Providers need to create an account. In the  button to create a service provider account request. After filling the sign-up form, the marketplace administrator will receive a notification to validate the service provider. Once the service provider user is validated, the user receives the credentials for  into the Marketplace. See form for options in Figure 2.

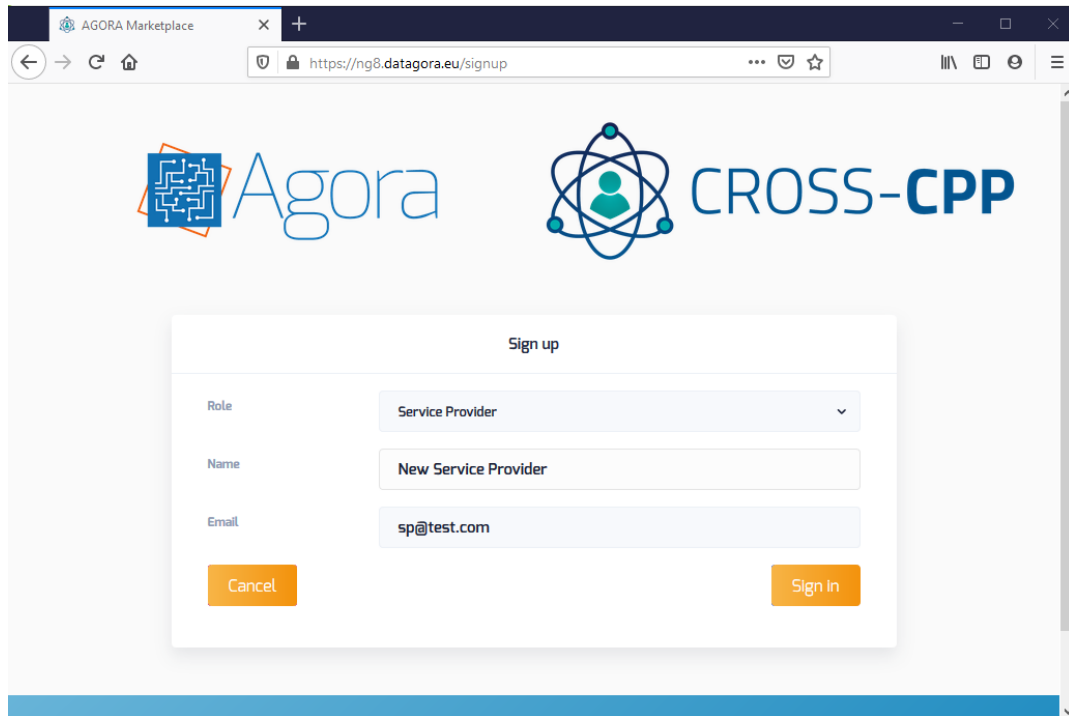
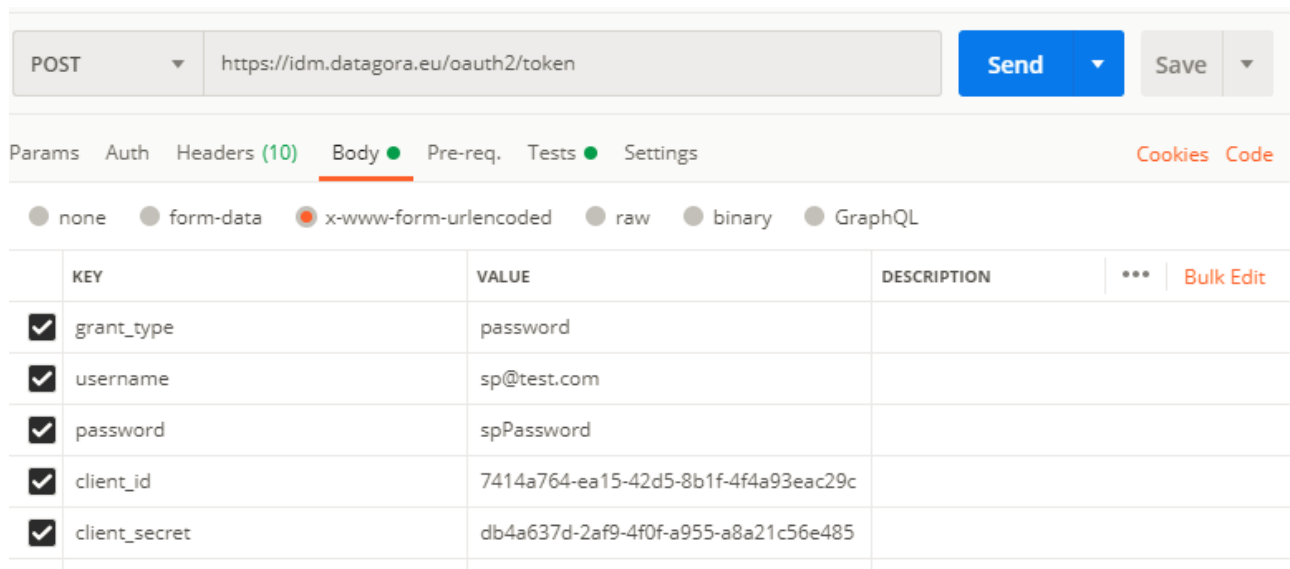


Figure 2. Cross-CPP Marketplace sign up form

In order to interact with the Marketplace API a service provider application needs to include the HTTP header, **x-Auth-Token: <Token>**, in every HTTP request (See more information in [3.1.1 Authentication below](#)). To get the Oauth token the application has to request to <https://idm.datagora.eu/oauth2/token> providing the credentials like in the following example.



	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	grant_type	password			
<input checked="" type="checkbox"/>	username	sp@test.com			
<input checked="" type="checkbox"/>	password	spPassword			
<input checked="" type="checkbox"/>	client_id	7414a764-ea15-42d5-8b1f-4f4a93eac29c			
<input checked="" type="checkbox"/>	client_secret	db4a637d-2af9-4f0f-a955-a8a21c56e485			

Figure 3. Postman example - Marketplace user authentication request

Response

```

{
  "access_token": "5d0c5c550153fe6d30f8864bd869002adb90c17e",
  "token_type": "Bearer",
  "expires_in": 3599,
  "refresh_token": "836a80b89e74ba2f3399d949ea3405f611bc737e",
  "scope": [
    "bearer"
  ]
}

```

Figure 4. Postman example - Marketplace user authentication response

1.2. Search for Shared Data (Data Discovery)

The second step is the discovery of relevant data. Different queries can be performed selecting different type of filtering like filters based on signals, time, region, etc. Even though the shared data search can be accessed both via the frontend as well as the API, the frontend is the recommended solution for users.

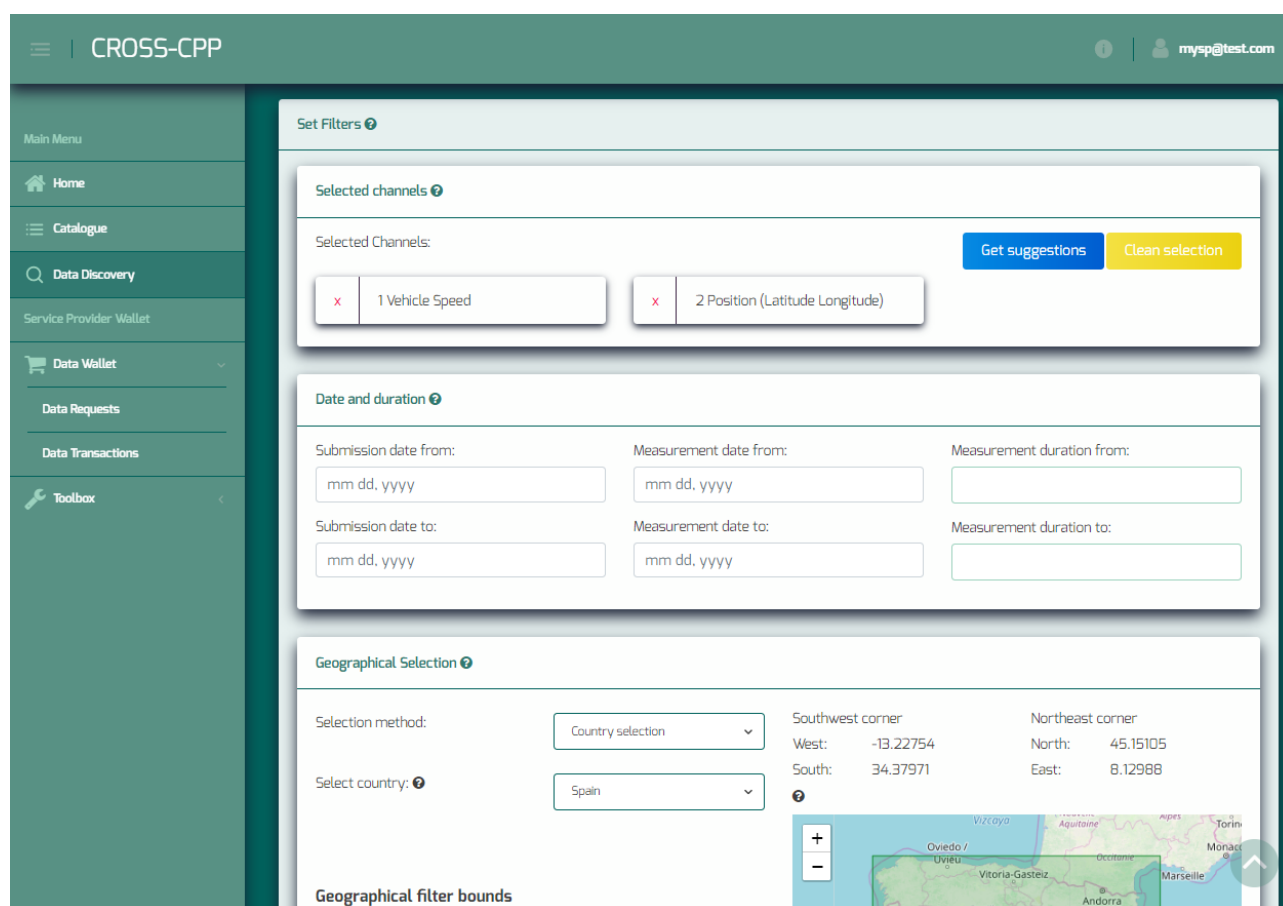


Figure 5. Cross-CPP Data Discovery view

1.2.1. Discovery Definition

The following figure (Figure 6) shows some screen capture of the discovery GUI. On the top, a signal filter can be applied. Signals can be searched for using the input fields in the topmost line. On the lower part of the figure one can see the button to "Get suggestions". After the first set of channels has been selected this option will invoke the Discovery functionality provided by the

Context Monitoring and Extraction module that makes semantically valid channels suggestion (suggestions are based on a specially created ontology where the signals are in relation to each other). The service provider may add any of the suggestions to the currently selected channels.

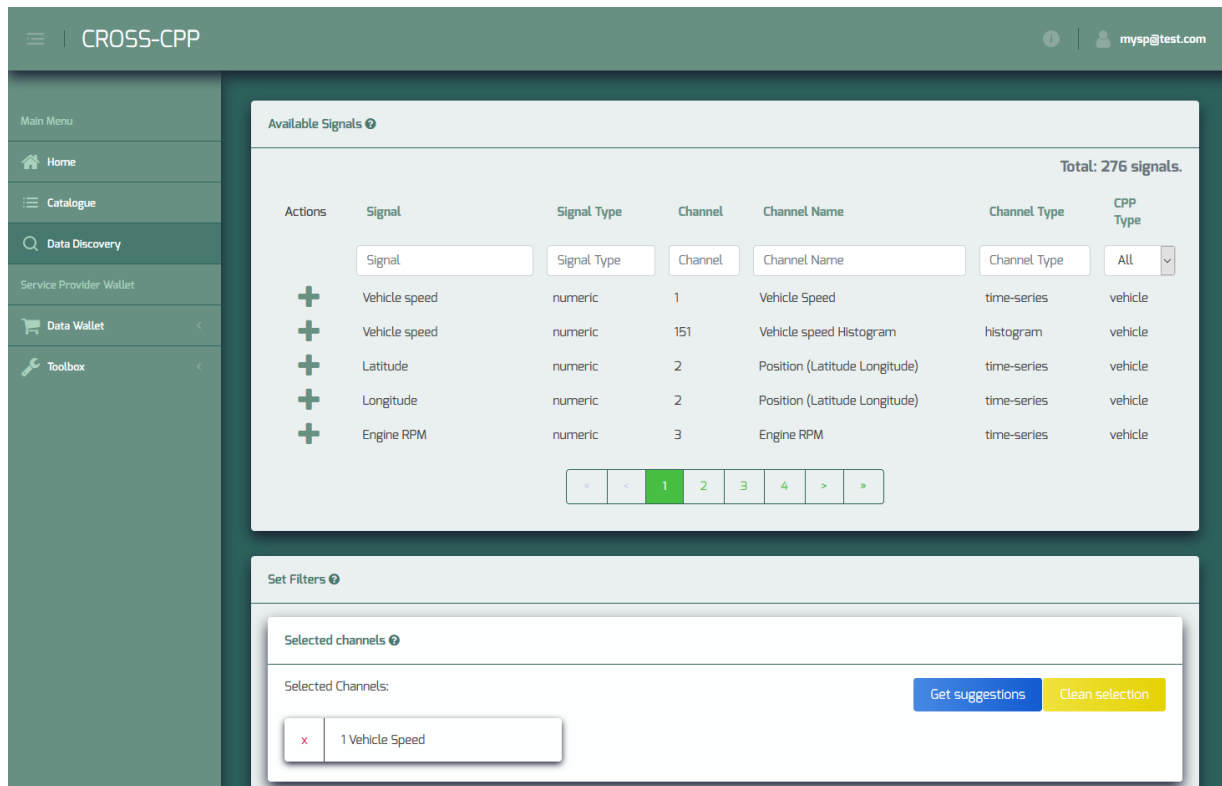


Figure 6. Data Discovery step 1

Below, time-filters can be applied. They cover the submission date of the data as well as the data occurrence time. The duration of data packages is also a filter to ensure a minimum amount of data length.

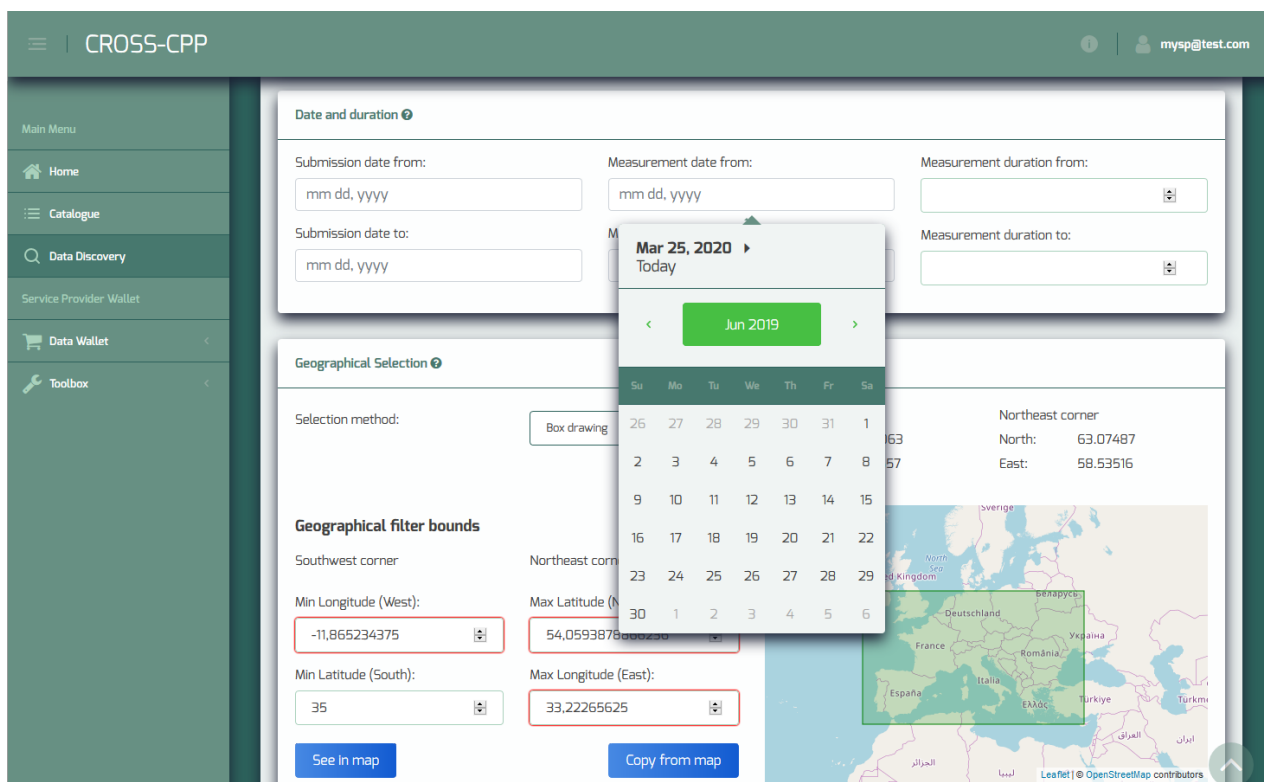


Figure 7. Data Discovery step 2

On the bottom right a geographical filter can be applied. By selecting the map on the screen, inserting the latitude and longitude or selecting countries from the drop-down list. By clicking on “Query” the database is searched for available data.

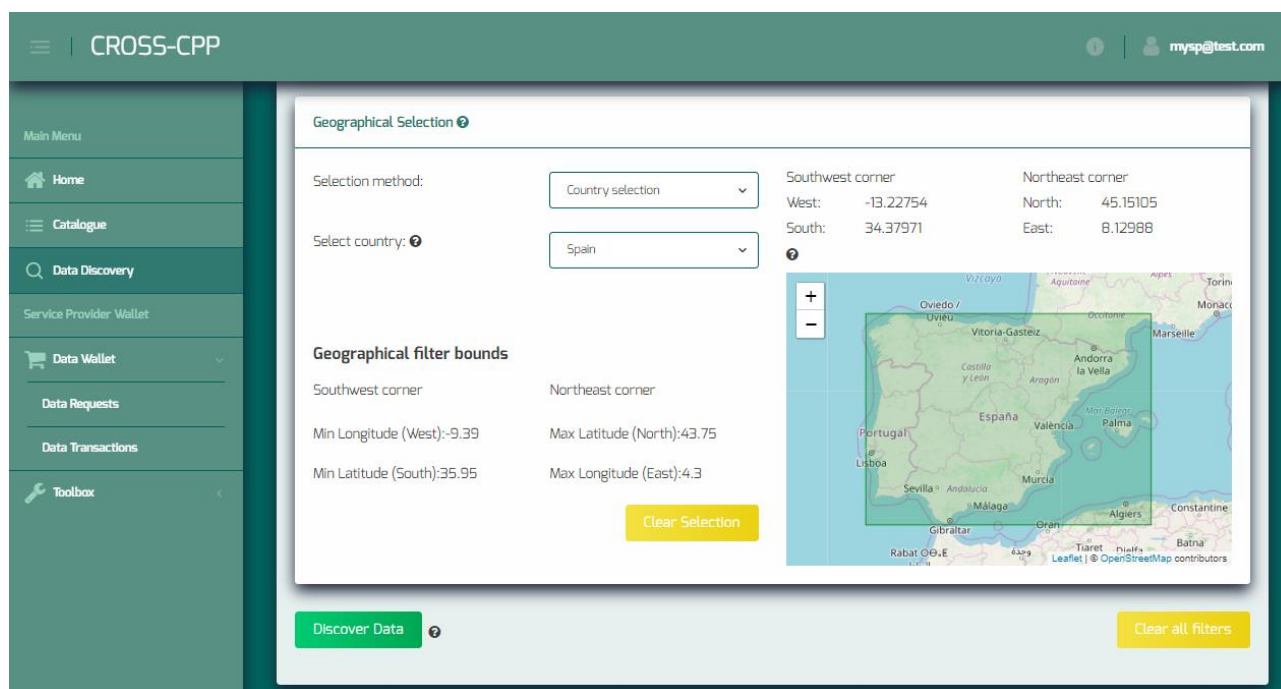


Figure 8. Data Discovery step 3

1.2.2. Discovery Results

After a query has been performed, the result is displayed below. Depending of the amount of found data, this may take some seconds. Figure 9 shows a screenshot of a query result. The result consists of a statistical overview, providing the number of possible data packages and the number of users, who collected those packages. Also, a timeline, when those data packages were generated is provided. A heat map provides a rough estimate of the area, which is covered by the vehicle data discovery (see Figure 10). Afterwards, if the expected availability does not match the needs, the query can be refined. Otherwise, if the availability is satisfactory, an offer needs to be created.

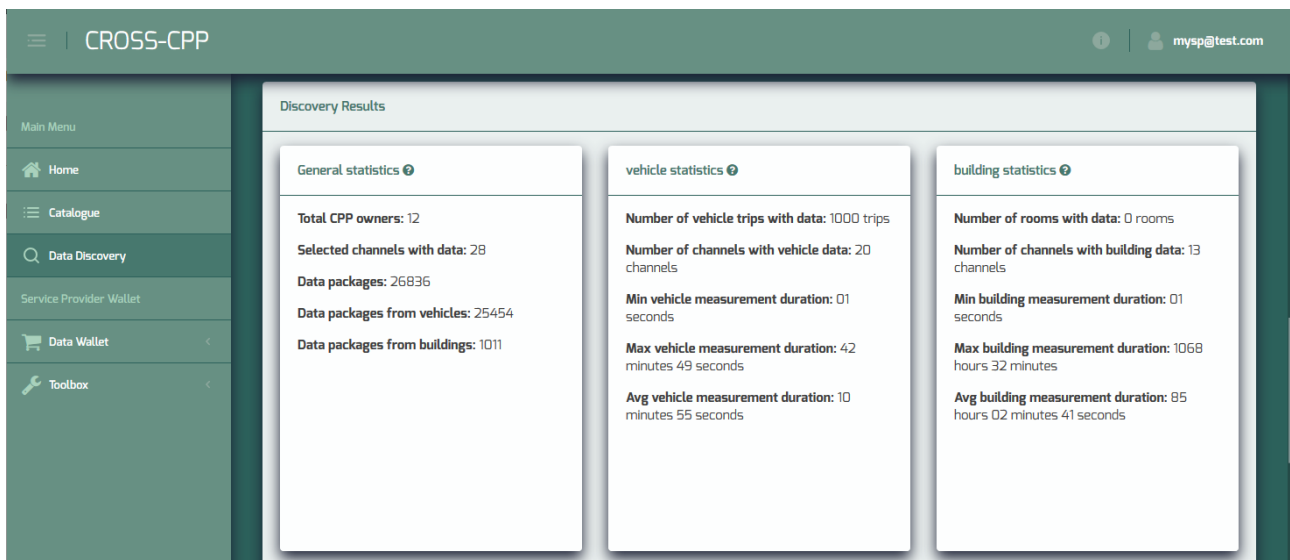


Figure 9. Discovery results - general statistics

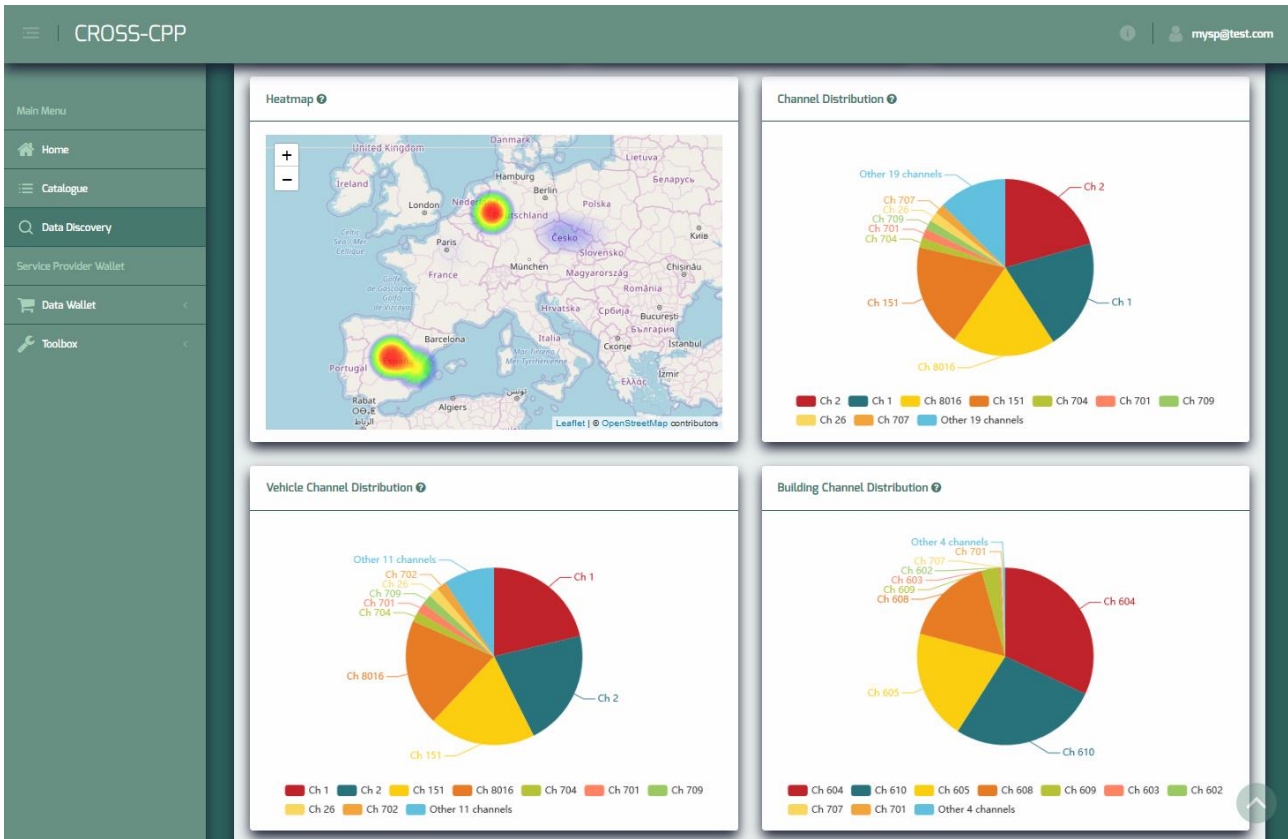


Figure 10. Discovery results - heatmap and charts

1.3. Data Request Creation

When the service provider is satisfied with the availability of data discovered, he can create a Data Request directly below the results of the data discovery. Only a short description must be provided to derive a Data Request from the previous search.

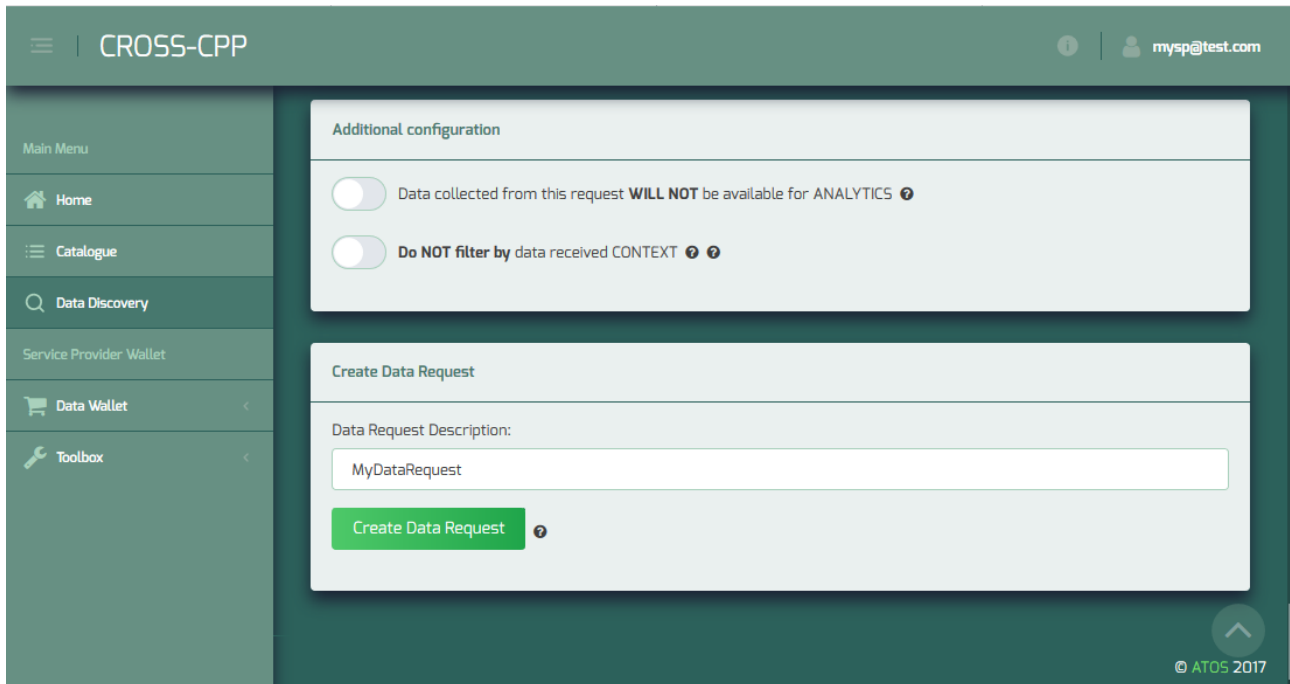


Figure 11. Create data Request from Data Discovery

Data owner users must accept a data request to give their consent to the service provider to access the data. The Marketplace does not store the shared data, but only caches metadata to locate the data in the Cloud Storages.

All created data request can be displayed under the menu

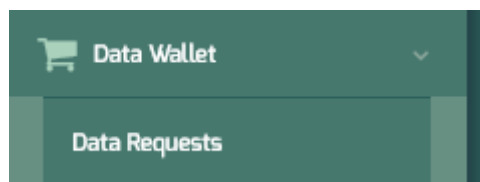


Figure 12 shows the detailed data request view. It displays the id as well as the subscription URL, which are needed to retrieve CIDM data from the offer. Single data packages can be seen by clicking on the "eye" icon in the data packages section (see Figure 13).

CROSS-CPP

mysp@test.com

Main Menu

Home

Catalogue

Data Discovery

Service Provider Wallet

Data Wallet

Data Requests

Data Transactions

Toolbox

Go Back

Data Request

ID

5e7b6bb278f6c92100f0b861

DESCRIPTION

MyDataRequest

AEON SUBSCRIPTION URL

https://aeon.atosresearch.eu:3000/subscribe/99cee559-2233-40bc-84e8-c8b78682e31d

CREATED

Mar 25, 2020

ANALYTICS

Data collected from this data request CANNOT be used for the creation of analytics

CONTEXT

Data collected from this data request WILL NOT be analysed and filtered by its context

Vehicle Data Request

CHANNELS

All channels

GEO BOUNDING BOX

All countries

SUBMISSION DATE

All submission dates

MEASUREMENT DATE

All measurement dates

TRAVEL DISTANCE

All distances

MEASUREMENT DURATION

All durations

Data Discovery

Data Packages

Figure 12. Data Request details

CROSS-CPP

myasp@test.com

Main Menu

Home

Catalogue

Data Discovery

Service Provider Wallet

Data Wallet

Data Requests

Data Transactions

Toolbox

Data Packages

Total: 3210 data packages.

Actions	Data Package ID	Data Type	CPP Type	Channel	Submitted	Start	Duration
	5ed73df8-5c4a-422a-9f04-9e9f94891355	geo-histogram	vehicle	8016	Nov 18, 2019	Nov 18, 2019 11:04:08	1361
	6c637f9c-49f2-42b1-aed9-92751ef471ff	histogram	vehicle	151	Nov 18, 2019	Nov 18, 2019 11:04:08	1361
	a9e62405-55a7-4670-b9a9-fa5c9a2127cc	time-series	vehicle	2	Nov 18, 2019	Nov 18, 2019 11:04:08	1361
	6359f778-cfd7-499b-bc5b-1b36a7c50520	time-series	vehicle	1	Nov 18, 2019	Nov 18, 2019 11:04:08	1361
	8152913b-2c4a-4068-b3ba-5d9df351bf8	geo-histogram	vehicle	8016	Nov 18, 2019	Nov 18, 2019 11:04:06	697
	ccfe26e3-389a-4972-ba7c-7f3e25451277	histogram	vehicle	151	Nov 18, 2019	Nov 18, 2019 11:04:06	697
	390b9571-5894-4007-9168-d2207d967c0b	time-series	vehicle	2	Nov 18, 2019	Nov 18, 2019 11:04:06	697
	a1462cdd-48fb-4867-892f-6bc9c11a0bbd	time-series	vehicle	1	Nov 18, 2019	Nov 18, 2019 11:04:06	697
	cea1fee3-a758-4511-8e88-16de86d1c4d3	geo-histogram	vehicle	8016	Nov 18, 2019	Nov 18, 2019 11:04:04	647
	dbfa1263-c9d1-45a1-8d14-6d850afefb85	histogram	vehicle	151	Nov 18, 2019	Nov 18, 2019 11:04:04	647

Figure 13. Data Request data packages received list

1.4. Data Retrieval

Data retrieval is the most crucial part of the whole Cross-CPP project. There are two possible methods to retrieve data:

- **Pull approach:** Data must be pulled by the Service Provider application from the Marketplace API.
- **Push approach:** Data is automatically pushed from the MP to the Service Provider application by means of a message queueing systems (AEON).

1.4.1. Pull Approach Data Retrieval

The pull approach uses the Marketplace REST API endpoints to retrieve data or metadata of a specific data request. The basic request consists of a HTTP-GET request providing the ID of the data request (offer) and additional query parameters to filter, sort, and pagination. This section provides some basic information on the procedures. For more detailed information see section 3 API below.

1.4.1.1. Metadata

The screenshot below shows the metadata CIDM pull request performed using Postman² tool.

<https://api.datagora.eu/api/ServiceProviderOffer/{OfferId}/CvimMetadata>

Metadata default page size is 1000 (limit=1000) and default page number is 0.

The response header "x-total-count" provides the total number of packages that match the filter. Also, total and page size is provided in the response of the body to facilitate the page iteration.

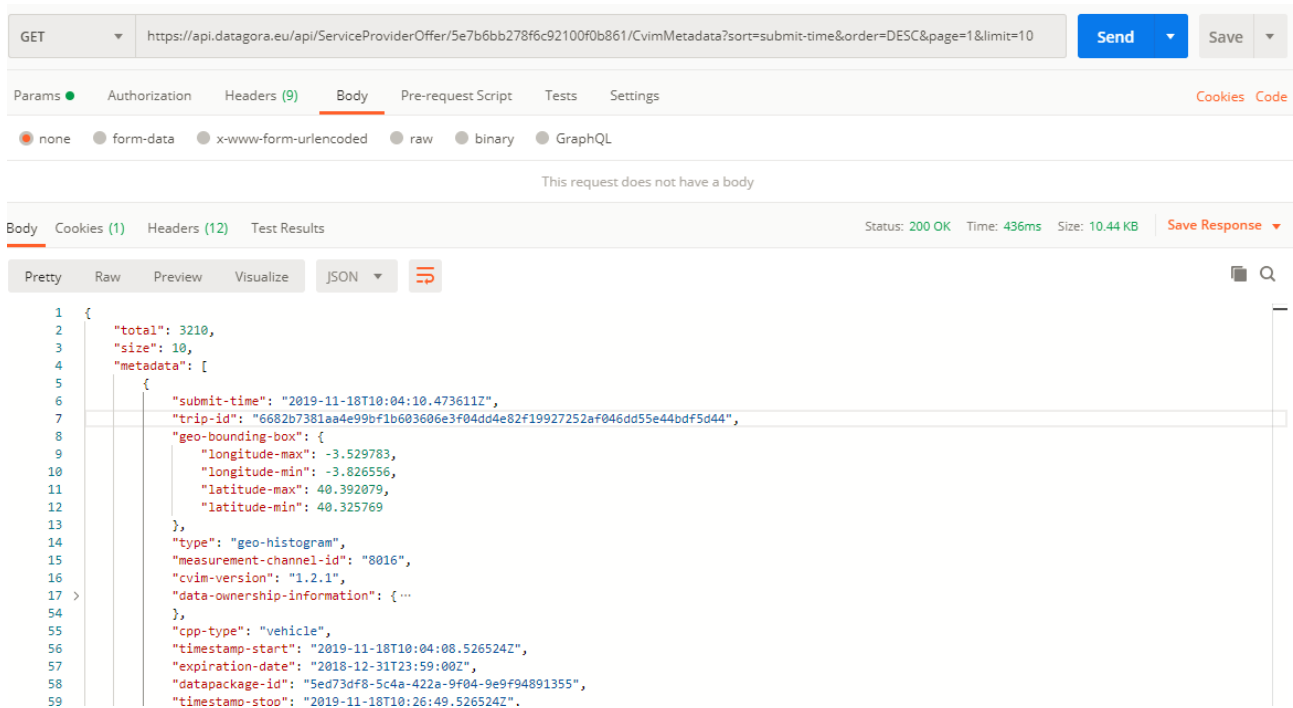


Figure 14. Data Request metadata pull example

1.4.1.2. Data Packages

The screenshot below shows the data CIDM pull request.

<https://api.datagora.eu/api/ServiceProviderOffer/{OfferId}/cvimDataPackages>

Data packages default page size is 10 (limit=10) and default page number is 0.

The response header "x-total-count" provides the total number of packages that match the filter.

² <https://www.postman.com>

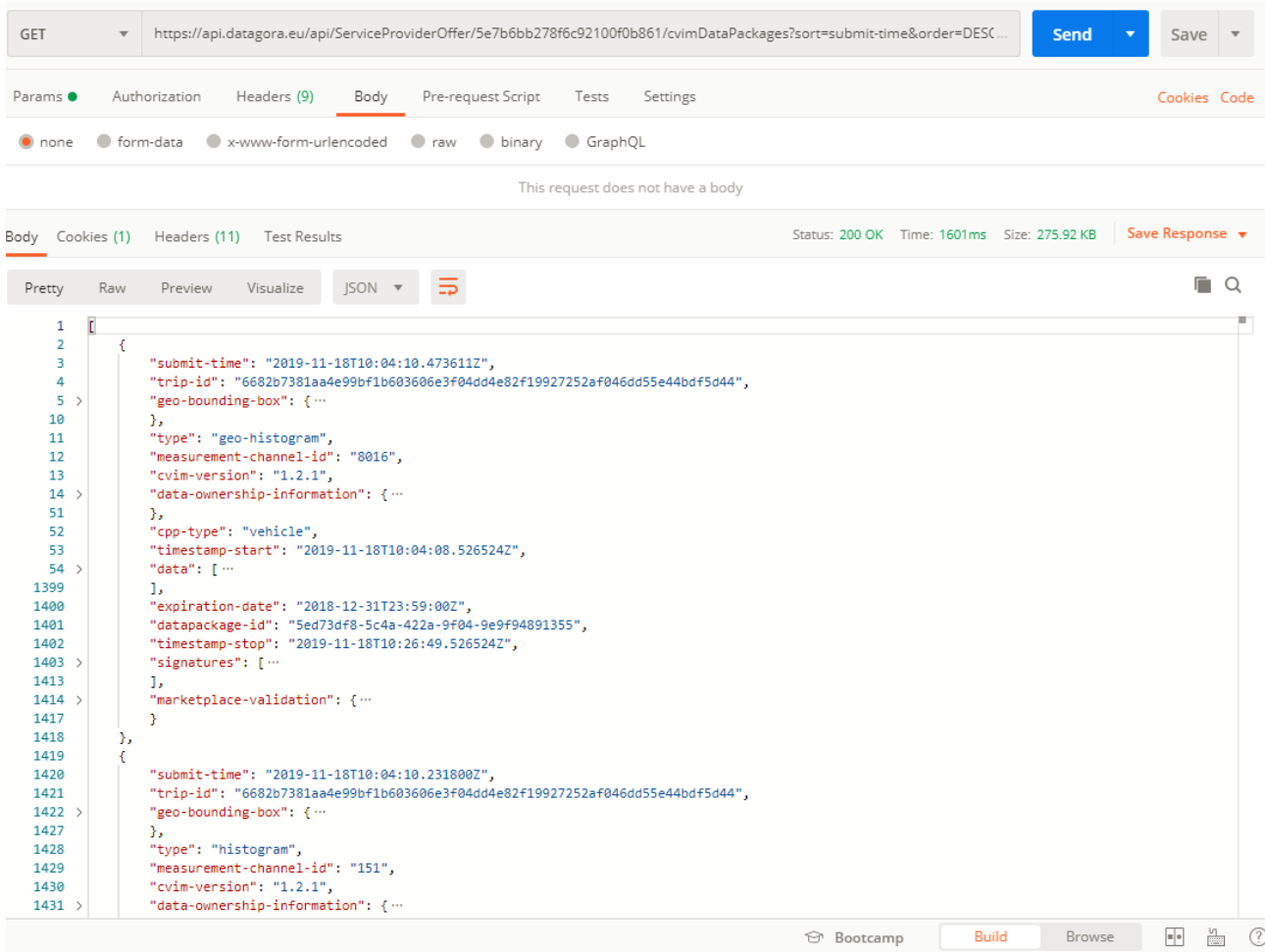


Figure 15. Data Request data pull example

The screenshot below shows a data CIDM pull request to get one data package.

<https://api.datagora.eu/api/ServiceProviderOffer/{OfferId}/cvimDataPackages/{datapackageId}>

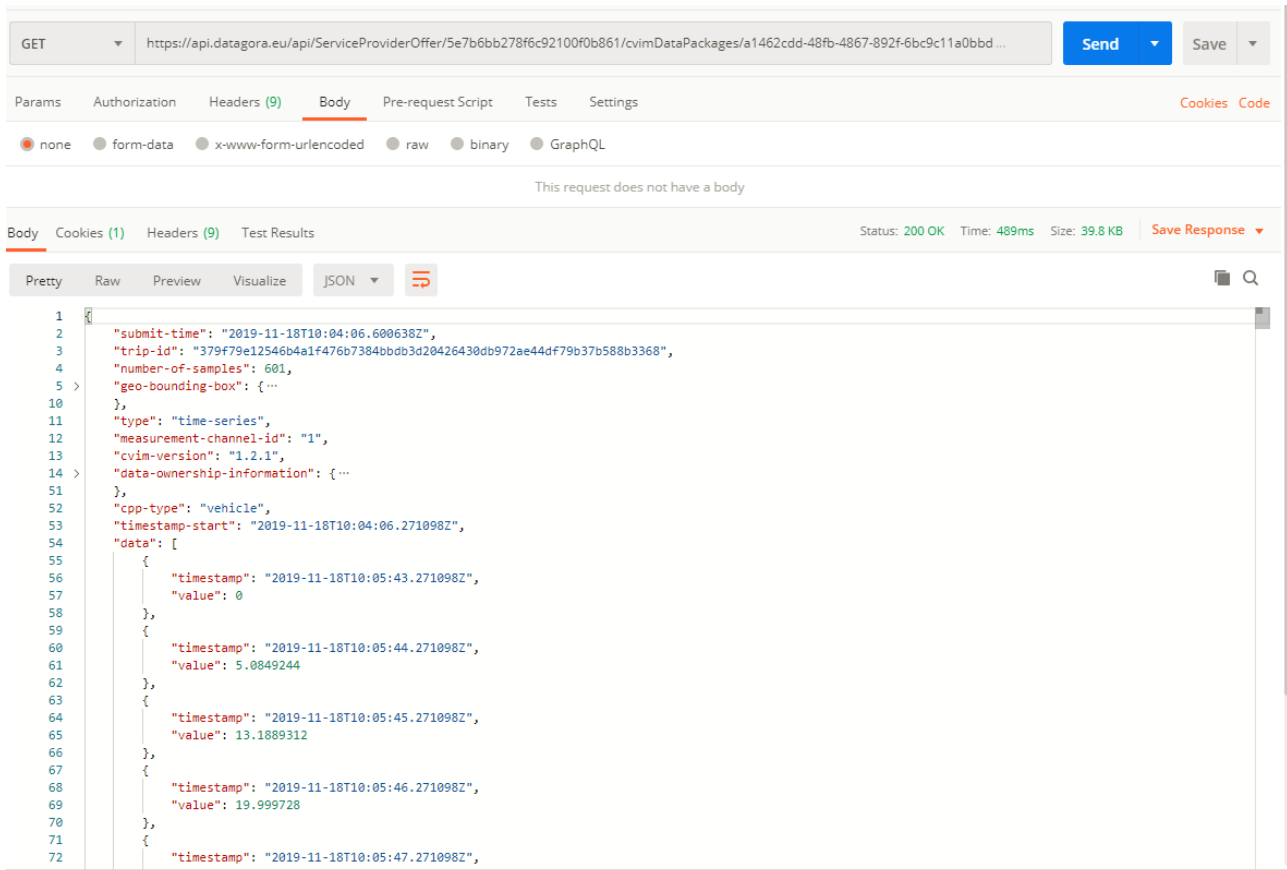


Figure 16. Data package details pull request

1.4.1.3. Query Parameters

Previous queries can filter packets providing query parameters as follow.

page: Number of page [1.x-total-count].

limit: Number of datapackages or metadata to receive in 1 page. Default value is 100 for datapackage and 1000 for metadata.

from: Offset from the first result you want to fetch. Default value is 0.

sort: Field to sort results.

order: Order to sort results (asc or desc).

where: Allow to filter data or metadata. It is JSON object which contains a channels, trip-id, submit-time.min and submit-time.max, timestamp.min and timestamp.max, latitude.min and latitude.max, and longitude.min and longitude.max.

Note: Result window is too large, from must be less than or equal to: [10000].

Where example: where = { "channels" : ["1","2"], "submit-time.min": "2019-07-01T12:39:30Z", "submit-time.max": "2020-05-09T11:38:00Z", "timestamp.min": "2019-07-01T12:39:30Z", "latitude.min" : 49.836, "latitude.max" : 49.840 }

1.4.1.4. Last value query

These queries are similar to the previous one but they provide the last datapackage or metadata of every device and every measurement-channel of the data request. They are useful for basic-cpp-information and event-values where the most relevant values are the last ones.

</ServiceProviderOffer/{OfferID}/lastcidmDataPackages>

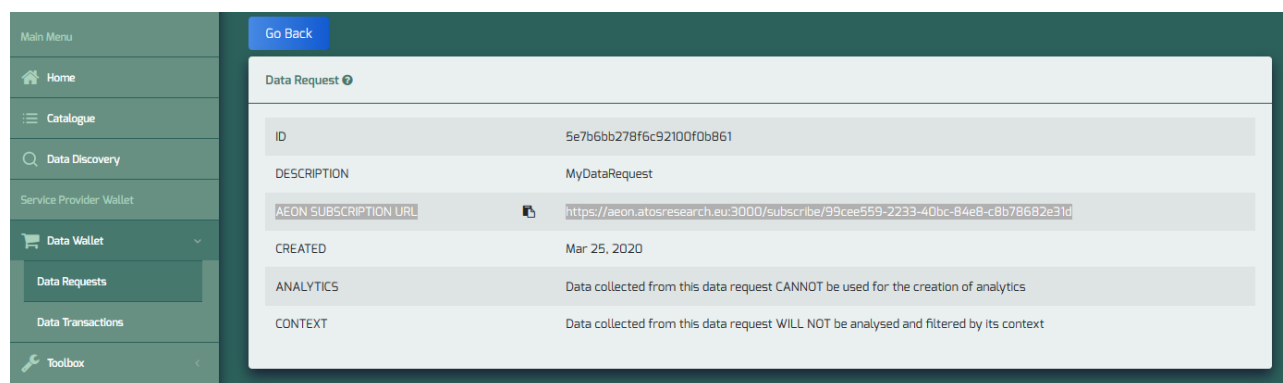
</ServiceProviderOffer/{OfferID}/lastcidmMetadata>

1.4.2. Push Approach Data Retrieval

The push approach data retrieval allows the processing of CIDM data packages as soon as they are available in the Marketplace. The push approach uses the AEON platform³, that is a cloud platform to create applications with real time communications channels. This section provides some basic information on the procedures. For more detailed information on AEON see section 4 AEON below.

AEON platform offers a cloud-based message queuing framework enabling messaging between various entities that wish to communicate with each other seamlessly and reliably using standard protocols.

When a new data request is created, the marketplace creates a new AEON Channel to provide data streaming between the service provider application and the marketplace. The service provider application needs to subscribe to the AEON Channel to receive the shared data. The information to subscribe is provided in the Data Request details (AEON Subscription URL).



Data Request	
ID	5e7b6bb278f6c92100f0b861
DESCRIPTION	MyDataRequest
AEON SUBSCRIPTION URL	https://aeon.atosresearch.eu:3000/subscribe/99cee559-2233-40bc-84e8-c8b78682e31d
CREATED	Mar 25, 2020
ANALYTICS	Data collected from this data request CANNOT be used for the creation of analytics
CONTEXT	Data collected from this data request WILL NOT be analysed and filtered by its context

Figure 17. Data Request AEON subscription url

The AEON SDK is provided for Java, Node.js and JavaScript programming languages (<https://scm.atosresearch.eu/ari/aeon-sdk/tree/master/SDK/releases>).

SDK instance

³ <https://aeon.atosresearch.eu/app/index.html>

To start publishing/subscribing with the SDK, it is necessary to instantiate a new SDK object.

```
sdk = new AeonSDK(url, [subscriptionData])
```

* url: AEON Subscription URL provided for the data request created.

* subscriptionData: It is a JSON containing information about the service provider subscription in order to make a unique client subscription.

```
{
  "id": " 7735901e3943424791f19f0dad5e428e",
  "desc": "MyDataRequest unique connection"
}
```

The combination of "id" and "desc" (strings) makes your subscription unique in the AEON network, the values are provided by de service provider application.

Subscribe

This functionality allows to subscribe to a specific channel and receive all the published information by the data request.

```
sdk.subscribe(received, [control])
```

* received: callback function to receive published data messages coming from the Marketplace.

* control: callback function to receive AEON control messages.

Each time that a message arrives over the subscription, the callback "received" will be executed in order to process the incoming message, the example below shows the messages on the console but you probably would put the data into a database or make a complex process of it .

Control messages are received over the subscription like messages but includes information regarding the status of the connection and the subscription. You can be subscribed (not recommended) without a "control" callback function. If control callback is passed, control messages will be received: see errors table.

Code	Message
1	Bad URL
3	Communication Infrastructure Down
50	Communication Infrastructure up
100	SDK operating in Publication Mode
101	SDK operating in Subscription Mode
201	Subscription in use
202	You are not subscribed
203	Subscription incorrect
250	You have been subscribed

251	Your subscription has been deleted
252	You have been unsubscribed

The example below shows the specification of the *control* and the *received* callbacks:

```
var messagesReceived = 0;
// Print control messages on the console
var control = function control(msg) {
  console.log("Control: ", msg);
  // Control: {"code":200,"desc":"ok","result":[{"subkey":"channelEdited-95504801-queue"}]}
}
// Print data messages and the number of message on the console
var received = function received(msg) {
  console.log("Received: ", msg)
  // Received: {"Message":"<first message from marketplace regarding a data request>"}
  messagesReceived++;
  console.log("Received messages: "+ messagesReceived);
  // Received messages: 1
}
```


2. Toolbox Guide

2.1. General Workflow of Toolbox Usage

The main purpose of the Toolbox is to provide Data Analytics of the shared data. The following Figure 18 shows the usage flow of the Marketplace Toolbox. First, a data request is needed for creating an analysis. Different types of analysis require different types of data. After creating an analysis, analysis data can be retrieved by the service provider using the Marketplace API or the SDK regarding the type of analysis created.

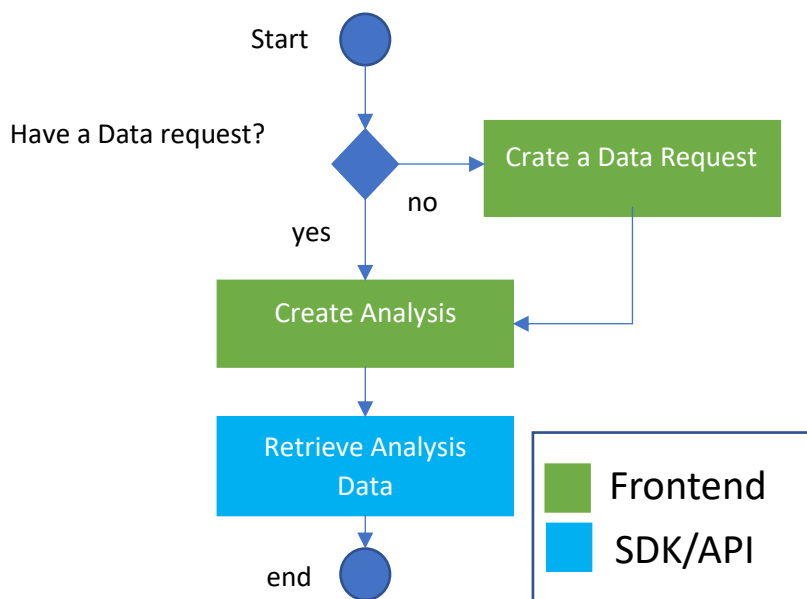


Figure 18. Analytics Toolbox workflow

There are two distinct sections inside the Toolbox: Analytics, which provides the analytical tools to analyse your data, and Data Views, which provides a different way of receiving the data and are used for the Machine Learning analytics.

2.2. Data Views

Data Views enable combining and joining data from measurement channels in a given Data Request. Service providers can use them to receive data filtered by defined criteria or to provide input to machine learning services. The filtered output can be directly sent in a specific AEON channel, or it can be retrieved by invoking the Retrieve function.

Data Views are created from Data Requests. It means that service providers have to choose relevant channels and create a Data Request first. Data owners then found the request and can decide to accept it. The Data View is then formed from channel values in data packages of the data owners who decided to provide the data.

2.2.1. Categories

A Category is a way to label and use the Data View to analyse the collected data for a certain purpose (i.e.: "charger availability" or "dangerous driving"). Categories consists on an enumeration of values provided by the Service Provider to categorize the collected data (i.e.: "high", "regular", "low").

Service Providers can assign a category value to each of the rows generated by the Data View.

Service Providers can create any number of categories, but a Data View can only have one assigned category, and once assigned it cannot be changed.

2.3. Analytics

2.3.1. Time Series Analytics

The Time Series Analytics are a set of analysis that explodes time series data packages. The analysis works in a streaming mode, receiving input data form the marketplace, calculating the results and sending the results to de service provider application subscribed to the analysis.

In order to create the analytics, the first step is to select different types of analytics functions like, sample entropy, permutation entropy, irreversibility, neural networks, regression tree, Arima, Pearson and Spearman.



Figure 19: Create Time Series analytics request – select analysis type

The second step is to select the data request that is going to provide the input data of the analysis. The third step is to select the target datapackages of the analytics that has to be time-series. The fourth and fifth steps are to define time filters and location filters. The sixth step is to provide additional parameter regarding the function type selected in the first step. Finally, the last step is to specify the name or description of the analysis. See user guide for further information about the types of analytics and the configuration.

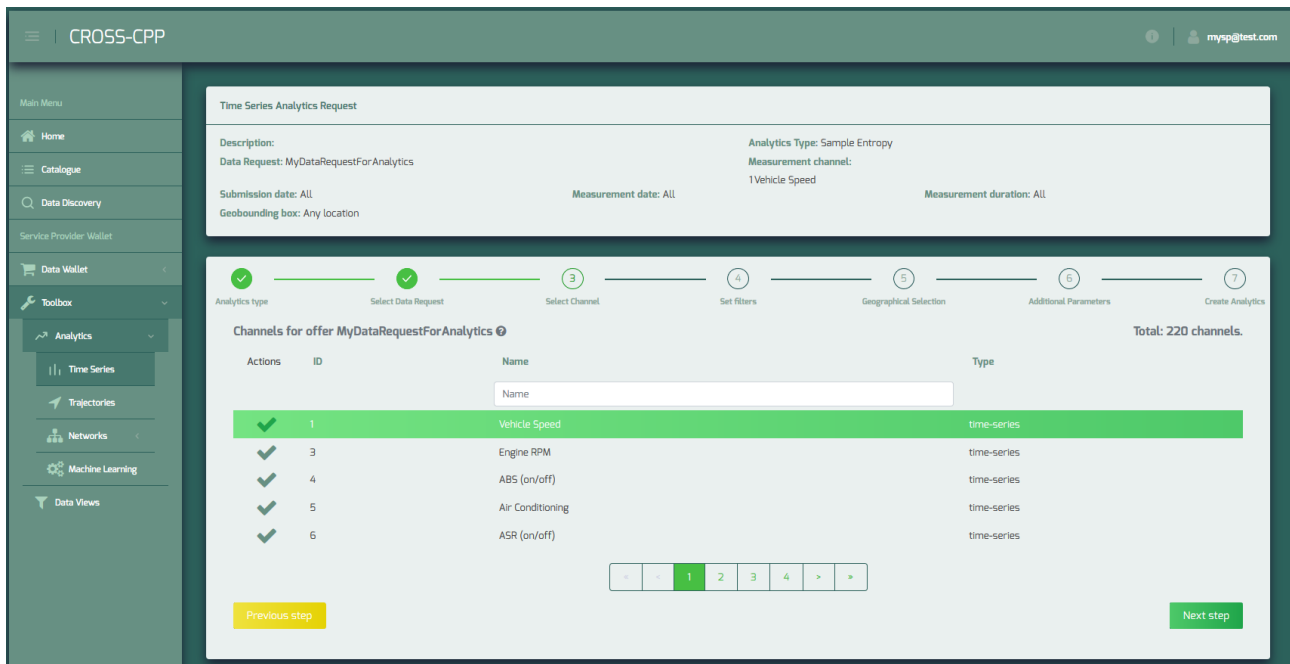


Figure 20: Create Time Series analytics request – select Data Request

Once the analysis is created, the marketplace provides the AEON subscription URL for the service provider application to receive the analytics results.

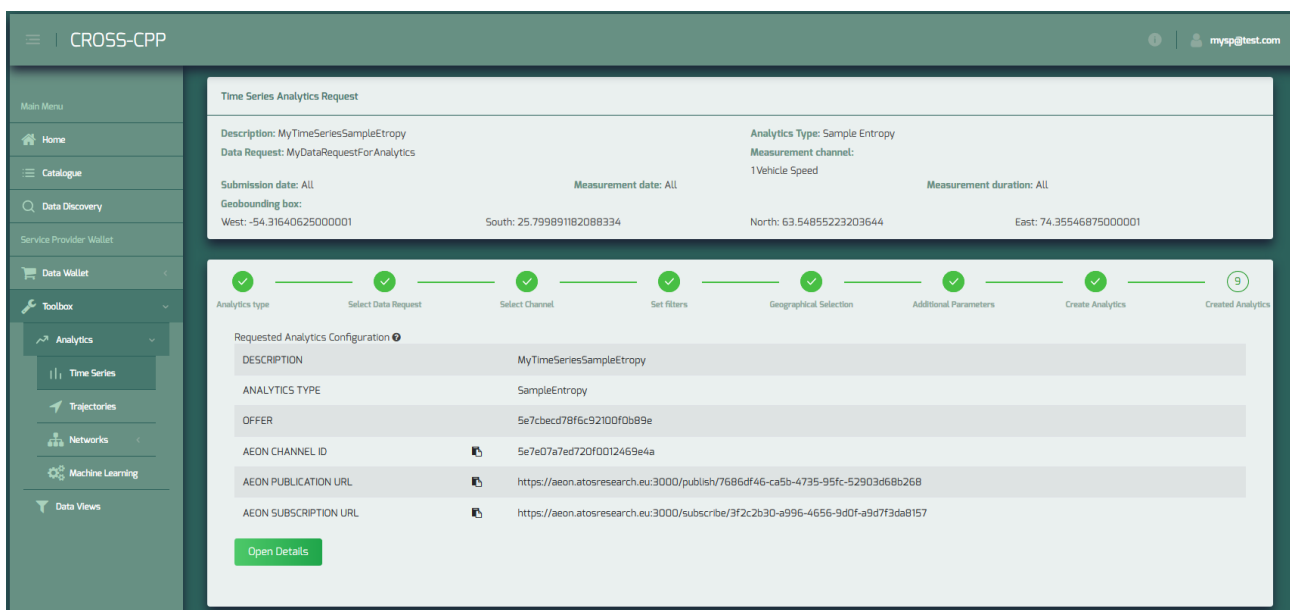


Figure 21: Create Time Series analytics request – requested analytics information

To subscribe and receive the results you have follow the instruction provided for the Push Approach Data Retrieval

2.3.2. Trajectory Analysis

The Trajectory Analytics are a set of analysis that target vehicle information that includes position timeseries datapackages. In order to create a trajectory analysis a data request that includes location is needed. The data request should be configured to collect data for analytics.

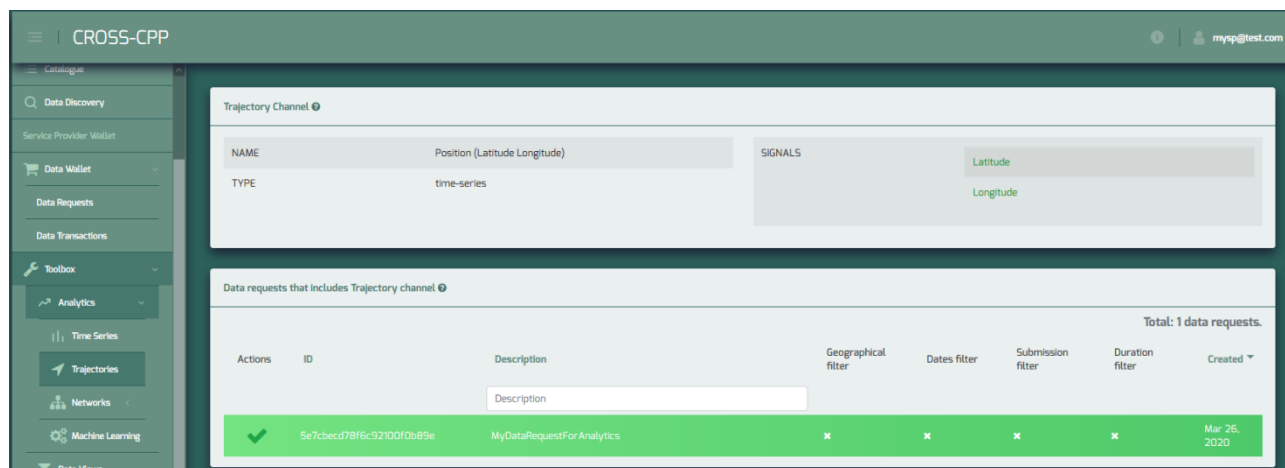


Figure 22: Request trajectory analytics – select Data Request

The filters can be configured for the analysis, like travel dates, submission dates, duration of the trip and location.

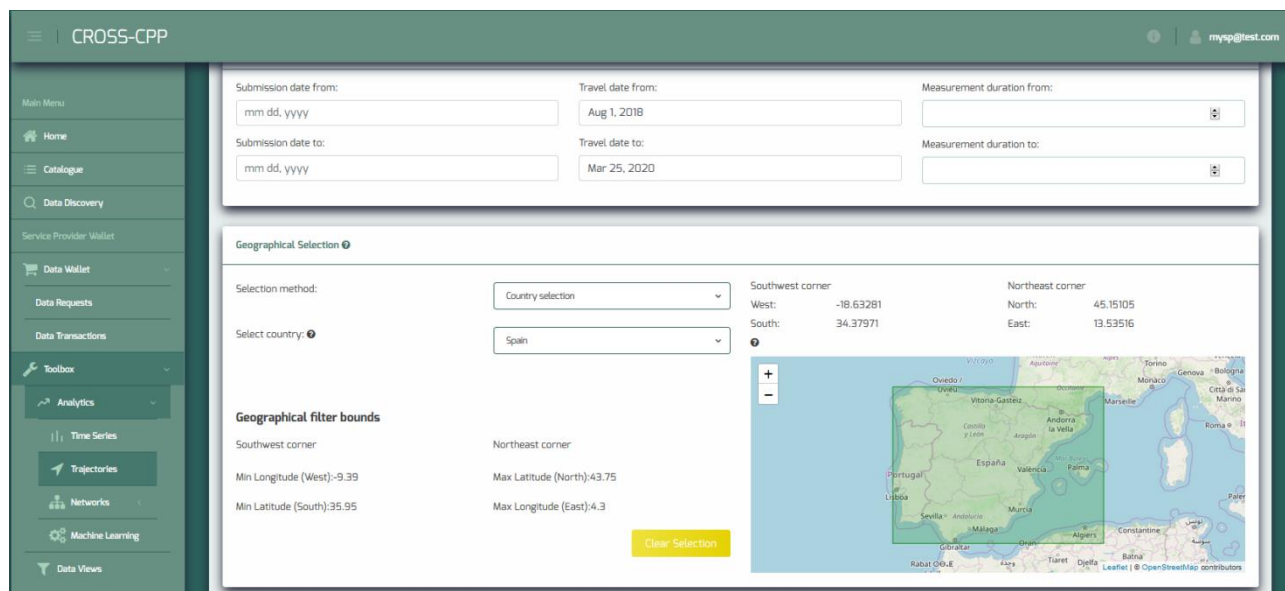


Figure 23: Request trajectory analytics – configure filters

There are many Analytics functions that can be performance selecting the analysis type and providing the parameters values when needed. To learn more about the analysis the user guide.

Finally, to create the request you have to provide the analysis description.

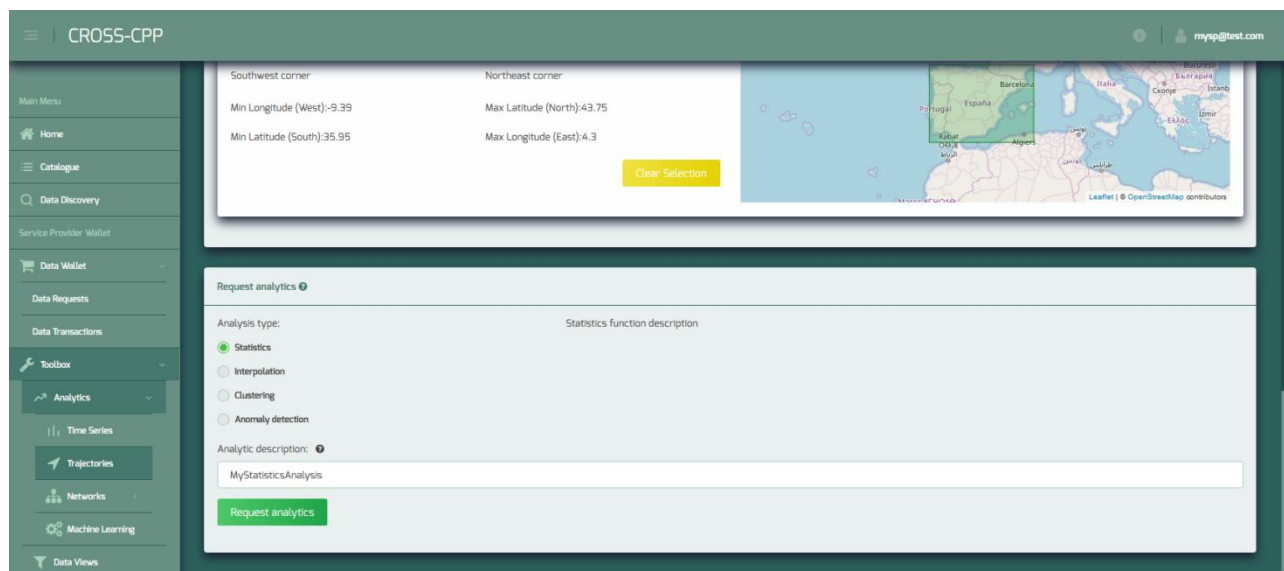
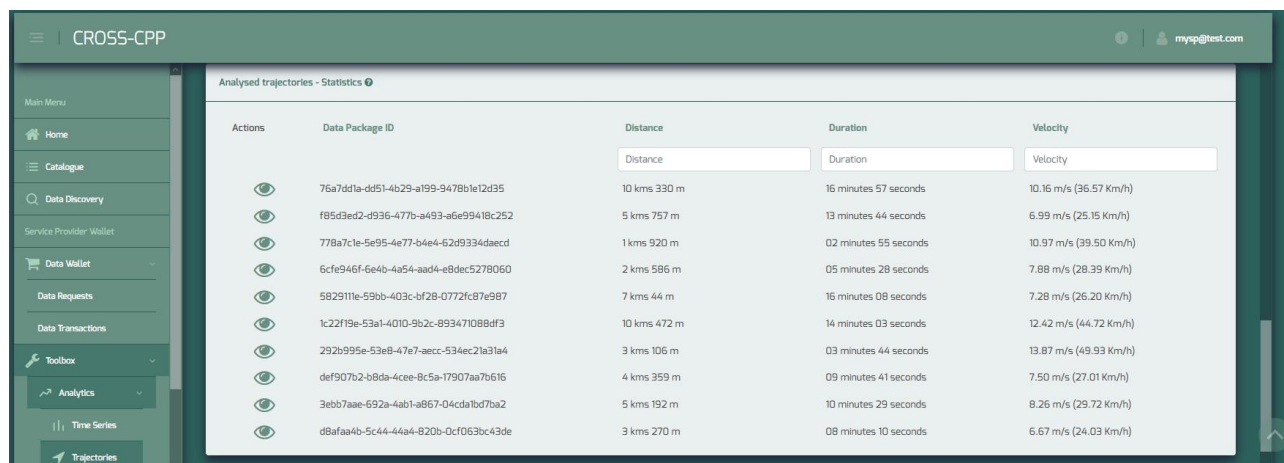


Figure 24: Request trajectory analytics – select analysis type

The process collects all the input data of the data request and apply the filters selected to get the input data of the analysis. The process has stages from the creation of the analysis: processing, OK, failed or no data.

If the status of the process is OK, the results are going to be available to see them in the marketplace application and to be recovered by service provider application via the marketplace API.



Actions	Data Package ID	Distance	Duration	Velocity
	76a7dd1a-dd51-4b29-a199-9478b1e12d35	10 kms 330 m	16 minutes 57 seconds	10.16 m/s (36.57 Km/h)
	f85d3ed2-d936-477b-a493-a6e99418c252	5 kms 757 m	13 minutes 44 seconds	6.99 m/s (25.15 Km/h)
	778a7c1e-5e95-4e77-b4e4-62d9334daecd	1 kms 920 m	02 minutes 55 seconds	10.97 m/s (39.50 Km/h)
	6cfe946f-6e4b-4a54-aad4-e8dec5278060	2 kms 586 m	05 minutes 28 seconds	7.88 m/s (28.39 Km/h)
	5829111e-59bb-403c-bf28-0772fc87e987	7 kms 44 m	16 minutes 08 seconds	7.28 m/s (26.20 Km/h)
	1c22f19e-53a1-4010-9b2c-893471088df3	10 kms 472 m	14 minutes 03 seconds	12.42 m/s (44.72 Km/h)
	292b995e-53e8-47e7-aec2-534ec21a31a4	3 kms 106 m	03 minutes 44 seconds	13.87 m/s (49.93 Km/h)
	def907b2-b8da-4cee-8c5a-17907aa7b616	4 kms 359 m	09 minutes 41 seconds	7.50 m/s (27.01 Km/h)
	3ebb7aae-692a-4ab1-a867-04cda1bd7ba2	5 kms 192 m	10 minutes 29 seconds	8.26 m/s (29.72 Km/h)
	d8afaa4b-5c44-44a4-820b-0cf063bc43de	3 kms 270 m	08 minutes 10 seconds	6.67 m/s (24.03 Km/h)

Figure 25: Trajectory analysis results

To retrieve data from the API you need to provide the ID of the analytics to this endpoint <https://api.datagora.eu/api/analytics/trajectoryAnalysis/{id}>

GET
https://api.datagora.eu/api/analytics/trajectoryAnalysis/5e7cc02178f6c92100f0b8a4
Send
Save

Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
Code

Headers
7 hidden

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Content-Type	application/json				
<input checked="" type="checkbox"/>	X-Auth-Token	093ba1dc21a2b2918999daa8a03d8064560b4178				
	Key	Value	Description			

Body
Cookies (1)
Headers (9)
Test Results
Status: 200 OK
Time: 398ms
Size: 4.31 KB
Save Response

Pretty
Raw
Preview
Visualize
JSON

```

1 {
2   > "statistics-results": [ ...
103   ],
104   "interpolation-results": [],
105   "clustering-results": [],
106   "travelDates": {
107     "min": "2018-07-31T22:00:00.000Z",
108     "max": "2020-03-24T23:00:00.000Z",
109     "createdAt": "2020-03-26T14:45:53.956Z",
110     "updatedAt": "2020-03-26T14:45:53.956Z",
111     "id": "5e7cc02178f6c92100f0b8a3"
112   },
113   > "offer": { ...
126   },
127   "description": "MyStatisticsAnalysis",
128   "model": "statistics",
129   "geoBoundingBoxConstraint": {
130     "latitude-min": 35.95,
131     "longitude-min": -9.39,
132     "latitude-max": 43.75,
133     "longitude-max": 4.3
134   },
135   "submissionDate": {},
136   "travelledDuration": {},
137   "variables": {},
138   "status": "OK",
139   "createdAt": "2020-03-26T14:45:53.962Z",
140   "updatedAt": "2020-03-26T14:45:56.032Z",
141   "id": "5e7cc02178f6c92100f0b8a4"
142 }

```

Figure 26: Trajectory analysis API – get analytics information

2.3.3. Networks

The Networks module enables a direct interpretation of the relationship between the signals collected from a specific channel (vehicle or building related). First, analogous to any of the previously detailed functions, the generation of a data request is required to be able to generate a network.

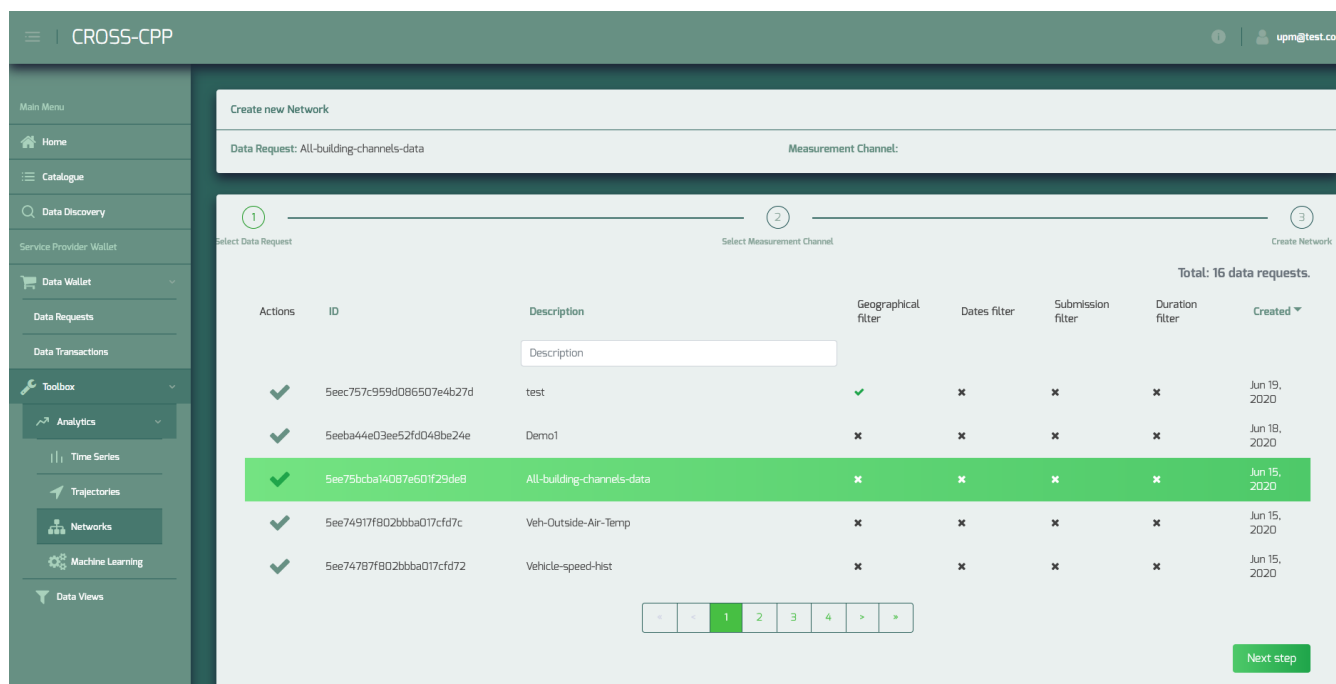


Figure 277: Request Networks module – Data Request Generation

The selected Data Request must have been generated in previous steps (analogous to the procedure indicated in the Trajectories Module).

After selecting the desired Data Request, the user must indicate which of the channels comprised in it is going to be analysed.

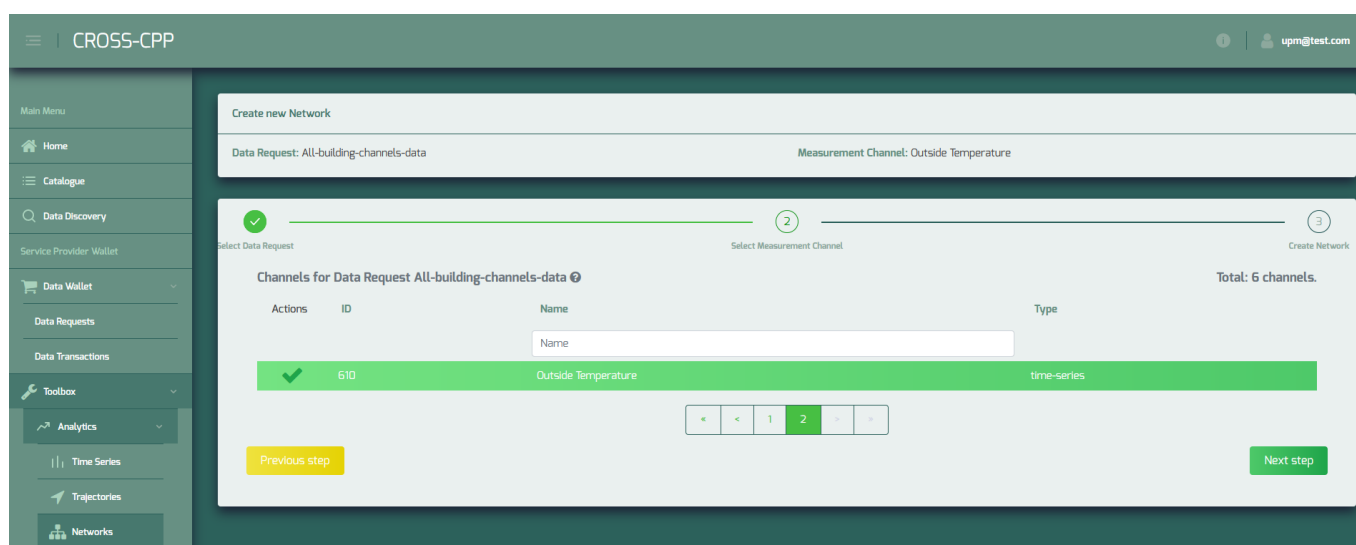


Figure 288: Request Networks module – select signal

Note that, as this module revolves around the idea of relationship between signals from a specific channel, only channels comprising more than one signal will be accepted as input to the network-generation process. Whether or not the selected channel fulfils this requirement is indicated in the interface.

As a final step, just click the "Create" button to proceed with the creation of the network.

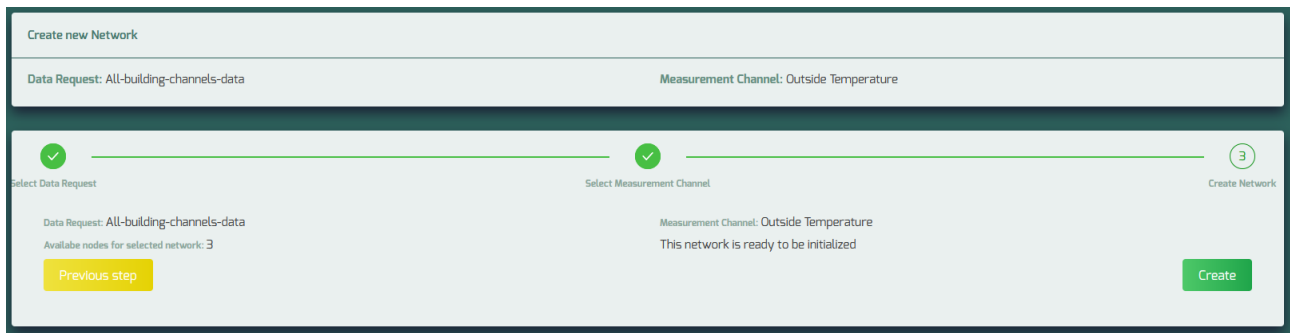


Figure 299: Request trajectory analytics – select analysis type

After the process is completed, the user will be presented an interactive visualization of the network as connected nodes (signals). A list of metrics regarding the structure of the network and the distribution of its nodes is also provided.

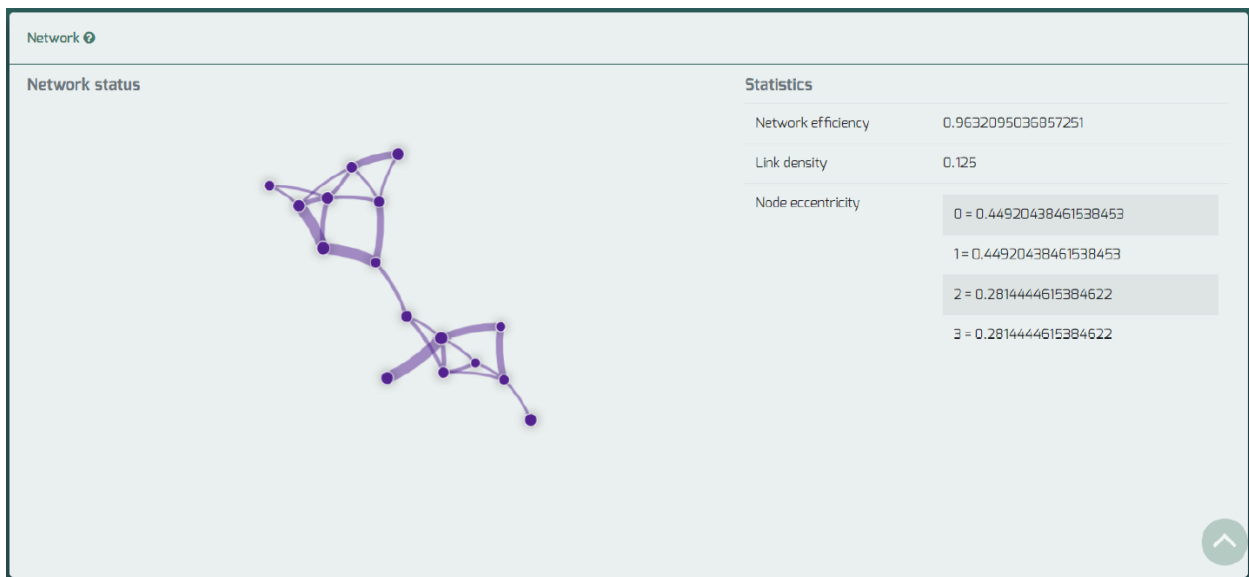


Figure 30: Request trajectory analytics – select analysis type

2.3.4. Machine Learning

Machine learning functions are based on API, which is specified in [OpenApi Specification version 3 format](#). Functions are divided into following categories:

- Initial machine-learning processing function (build model)
- Common control functions (status of process, kill process, drop model, export or import model)
- Main machine-learning processing functions (update model, apply model and evaluate model)

All of main machine-learning processing functions need channel data in context for their processing. This is done by obtaining data from data views. Thus, first of all, it is needed to create particular data views for these functions.

In case of classification, there is needed to create category column by endpoint /sei/category/create and its filling by endpoint /sei/category/assign.

2.3.4.1. Initial machine-learning processing function

2.3.4.1.1. Create (build) machine-learning model

The basis of remaining main machine-learning processing operations is to create (build) machine-learning model. Model is created by endpoint ml_model/batch/build, which returns unique identifier of this model, which is used in other endpoints as an input argument.

Example – Request for creation of machine learning model for channels 1 (speed), 2 (location) and 150 (rain intensity):

```
{
  "alname": "sklearn.neureal_network.MLPClassifier",
  "config": {},
  "data": {
    "view_id": "demo_driver_behavior_train_dataset",
    "limit": 100000,
    "page": 0
  }
}
```

Example – Response for creation of machine learning model for channels 1 (speed), 2 (location) and 150 (rain intensity):

```
{
  "status": "success",
  "data": {
    "alguid": "e85986bc-e80f-4ba8-b9c3-d2e1010b6755"
  }
}
```

2.3.4.2. Common control functions

2.3.4.2.1. Status of process

When this request was sent, it can be found out the state of model creation process by endpoint /ml_model/status, where we can find, if the creation process already running, was successfully finished or was finished with some error.

Example – Request for status of machine learning model creation process for channels 1 (speed), 2 (location) and 150 (rain intensity):

```
{
  "alguid": "e85986bc-e80f-4ba8-b9c3-d2e1010b6755"
```

```
}
```

Example – Response for status of machine learning model creation process for channels 1 (speed), 2 (location) and 150 (rain intensity):

```
{
  "status": "success",
  "data": {
    "status": "finished"
  }
}
```

2.3.4.2.2. Kill a process

In case that creation process still running and we need to stop this process, we can use endpoint /ml_model/kill – this endpoint only stops the processing of model creation, an entry of model creation is still preserved.

Example – Request of kill some machine learning model creation process:

```
{
  "alguuid": "01720cf6-d4d4-4e4f-93bc-76de3e08f8c3"
}
```

Example – Response of kill some machine learning model creation process:

```
{
  "status": "success",
  "data": []
}
```

2.3.4.2.3. Drop a process

If the creation process is no longer running, it is possible to drop it /ml_model/drop. Otherwise it is needed to kill it by previously mentioned endpoint.

Example – Request of drop some machine learning model entry:

```
{
```

```
"alguid": "01720cf6-d4d4-4e4f-93bc-76de3e08f8c3"
}
```

Example – Response of drop some machine learning model entry:

```
{
  "status": "success",
  "data": []
}
```

2.3.4.2.4. Export model

Once a model is created, it may be exported by endpoint /ml_model/export to its internal representation, from which can be imported again. It can be used for backup or experimentation purposes.

Example – Request of export some machine learning model entry:

```
{
  "alguid": "01720cf6-d4d4-4e4f-93bc-76de3e08f8c3"
}
```

Example – Response of export some machine learning model entry is a file with following content:

```
{
  "type": "batch",
  "alname": "sklearn.neural_network.MLPClassifier",
  "config": {},
  "data": {
    "sei_uri_base": "https://vian-dev.fit.vutbr.cz/cross-cpp/",
    "view_id": "demo_driver_behavior_train_dataset",
    "limit": 100000,
    "page": 0
  },
  "model":
  "\\200\\u0003csklearn.neural_network.multilayer_perceptron\\nMLPClassifier\\nq\\000
  )\\201q\\u0001}q\\u0002(X\\n\\000\\000\\000activationq\\u0003X\\u0004\\000\\000\\000r
  eluq\\u0004X\\u0006\\000\\000\\000solverq\\u0005X\\u0004\\000\\000\\000adamq\\u0006X\\
  u0005\\000\\000\\000alphaq\\u0007G?\\u001a6\\342\\353\\u001cC-
  X\\n\\000\\000\\000batch_sizeq\\bX\\u0004\\000\\000\\000autoq\\tX\\r\\000\\000\\000le
  arning_rateq\\nX\\b\\000\\000\\000constantq\\u000bX\\u0012\\000\\000\\000learning_ra
  te_initq\\fG?PbM\\322\\361\\251\\374X\\u0007\\000\\000\\000power_tq\\rG?\\340\\000\\
  000\\000\\000\\000\\000X\\b\\000\\000\\000max_iterq\\u000eK\\310X\\u0004\\000\\000
  \\000lossq\\u000fX\\b\\000\\000\\000log_lossq\\u0010X\\u0012\\000\\000\\000hidden_la
  yer_sizesq\\u0011Kd\\205q\\u0012X\\u0007\\000\\000\\000shuffleq\\u0013\\210X\\f\\000\\
  000\\000random_stateq\\u0014NX\\u0003\\000\\000\\000tolq\\u0015G?\\u001a6\\342\\353
```

```

\u001cC-
X\u0007\000\000\000verboseq\u0016\211X\n\000\000\000warm_startq\u0017\21
1X\b\000\000\000momentumq\u0018G?\354\314\314\314\314\314\315X\u0012\
000\000\000nesterovs_momentumq\u0019\210X\u000e\000\000\000early_stoppingq
\u001a\211X\u0013\000\000\000validation_fractionq\u001bG?\271\231\231\23
1\231\231\232X\u0006\000\000\000beta_1q\u001cG?\354\314\314\314\314\
314\315X\u0006\000\000\000beta_2q\u001dG?\357\367\316\331\u0016\207+X\u
0007\000\000\000epsilonq\u001eG>Ey\216\3420\214:X\u0010\000\000\000n_it
er_no_changeq\u001fK\nX\u0010\000\000\000_label_binarizerq
csklearn.preprocessing.label\LabelBinarizer\nq!)\201q\}q#(X\t\000\000\000n
eg_labelq$K\000X\t\000\000\000pos_labelq%K\u0001X\r\000\000\000sparse_out
putq&\211X\u0007\000\000\000y_type_q'X\n\000\000\000multiclassq(X\r\000\
\000\000sparse_input_q)\211X\b\000\000\000classes_q*cumpy.core.multiarray\
n_reconstruct\nq+cumpy\ndarray\nq,K\000\205q-
C\u0001bq.\207q/Rq0(K\u0001K\u0004\205q1cumpy\ndtype\nq2X\u0002\000\000\0
00U9q3K\000K\u0001\207q4Rq5(K\u0003X\u0001\000\000\000<q6NNNK$K\u0004K\btq7
b\211c\220d\000\000\000a\000\000\000n\000\000\000g\000\000\000e\0
00\000\000r\000\000\000o\000\000\000u\000\000\000s\000\000\000h\0
00\000\000a\000\000\000z\000\000\000a\000\000\000r\000\000\000d\0
00\000\000o\000\000\000u\000\000\000s\000\000\000n\000\000\000o\0
00\000\000r\000\000\000m\000\000\000a\000\000\0001\000\000\000\00
0\000\000\000\000\000\000\000\000\000\000r\000\000\000i\000\
000\000s\000\000\000k\000\000\000y\000\000\000\000\000\000\000\00
0\000\000\000\000\000\000\000\000\000\000q8tq9bX\u0010\000\000\
\000_sklearn_versionq:X\u0006\000\000\0000.21.1q;ubh*h0X\n\000\000\000n_ou
tputs_q<K\u0004X\r\000\000\000_random_stateq=cumpy.random\n_RandomState_cto
r\nq>)Rq?(X\u0007\000\000\000MT19937q@h+h,K\000\205qAh.\207qBRqC(K\u0001Mp
\u0002\205qDh2X\u0002\000\000\000u4qEK\000K\u0001\207qFRqG(K\u0003h6NNNJ\
377\377\377\377J\377\377\377\377K\000tqHb\211B\300\t\000\000\243S\
377\u0011\211\u0010\334\u0018X\327\314\267{\363\303\3701\364\372]\u001
9\224JD\361t\u0007\366\273\271\351\255g\266+\201Q4y^\254\277\247\
276\262\314}c\351)\230\250\342\212\336\3251\273\353\212\316\314]\
212\u0011d\u001f(\227\225\367\203\211:Y\360\340oWSH\317\374L"%Z\373\
333v\214\313\u0010D\257$\234\344\212\277\u001b\316%\365\u0007g[u\245\
2654E9\273\304\332\275%\214\u0006\215\u0003\u0004\332%\264\314\247\t\u
0005\251\367\265\344\273F\320\373\366^]\322\355_\365RL\202Lkv\u001f\
\3251\345\371\235Qa\225\250\353\230\276:\u000e|\000\342x]7{f\356{\27
7q\222~\3204\360x\216-
\u001b\3175\237\u0014\271\250[\235\226_\354\u0007\220\000\301A\u001e
\360 \277\235\u001d@ia\307\u000f\2707M\267\374\u0018nW\
[...]
Eq\321\246\u0004r4?`i5\325\367\277\257>:\345\261\275'\3220?\275\253I
\223a\u001aG?r\u0005\u0002\000\000tr\u0006\u0002\000\000beubX\u0005\000\0
00\000loss_r\u0007\u0002\000\000j\306\u0001\000\000h:h;ub."
}

```

2.3.4.2.5. Import model

Instead of creating a model by build endpoint you can also use import endpoint (/ml_model/import) to import model from internal representation created from the previous use of export endpoint

Example – Request of import some machine learning model entry:

```

{
  "type": "batch",

```

```

    "alname": "sklearn.neural_network.MLPClassifier",
    "config": {},
    "data": {
        "sei_uri_base": "https://vian-dev.fit.vutbr.cz/cross-cpp/",
        "view_id": "demo_driver_behavior_train_dataset",
        "page": 100000,
        "limit": 0
    },
    "model": "\\200\\u0003csklearn.neural_network.multilayer_perceptron\\n
MLPClassifier\\nq\\000)\\201[...]"
}

```

Example – Response of import some machine learning model entry:

```

{
    "status": "success",
    "data": {
        "alguuid": "91b33944-c0d3-4115-89e3-d3b148465805"
    }
}

```

2.3.4.3. Main machine-learning processing functions

2.3.4.3.1. Update machine-learning model

Already created model can be improved by extra data (for example other page from training data or another data view) by `/ml_model/batch/update` endpoint. This endpoint creates new model and assign to it new `alguuid` identifier, which is returned.

Example – Request of update some machine learning model:

```

{
    "alguuid": "e85986bc-e80f-4ba8-b9c3-d2e1010b6755",
    "config": {},
    "data": {

```

```

    "view_id": "demo_driver_behavior_train_dataset",
    "limit": 100000,
    "page": 1
  }
}

```

Example – Request of update some machine learning model:

```

{
  "status": "success",
  "data": {
    "alguuid": "a94ee08e-90cb-4c9e-b043-0ba70d8d584f"
  }
}

```

2.3.4.3.2. Apply machine-learning model to testing data

Once a model is trained, we can apply it to testing data to predict some value or classify to some class by /ml_model/batch/apply endpoint.

Example – Request of apply some machine learning model to testing data:

```

{
  "alguuid": "e85986bc-e80f-4ba8-b9c3-d2e1010b6755",
  "data": {
    "view_id": "demo_driver_behavior_test_dataset",
    "limit": 1000,
    "page": 0
  }
}

```

Example – Request of apply some machine learning model to testing data:

```

{
  "status": "success",
  "data": [
    ...
    {

```

```
"1": "69.79254800283908",
"150": "0.8944514838653745",
"2_Latitude": "49.23487606034672",
"2_Longitude": "16.591834275131614",
"ml_prediction_output": "normal"
},
{
  "1": "75.11103007668163",
  "150": 0,
  "2_Latitude": "49.23502669317959",
  "2_Longitude": "16.591701072103056",
  "ml_prediction_output": "normal"
},
{
  "1": "69.7968282329311",
  "150": "0.7510775046095213",
  "2_Latitude": "49.235188804860044",
  "2_Longitude": "16.591557718450403",
  "ml_prediction_output": "dangerous"
},
{
  "1": "70.80922943008888",
  "150": 0,
  "2_Latitude": "49.23533944693091",
  "2_Longitude": "16.591424507252782",
  "ml_prediction_output": "normal"
},
{
```

```

    "1": "72.56495706486243",
    "150": "0.5417261819678416",
    "2_Latitude": "49.23549227406114",
    "2_Longitude": "16.59128936383017",
    "ml_prediction_output": "risky"
  },
  {
    "1": "70.2379894739633",
    "150": "0.4606970283059384",
    "2_Latitude": "49.23564889056766",
    "2_Longitude": "16.59115086950198",
    "ml_prediction_output": "normal"
  },
  ...
]
}

```

2.3.4.3.3. Evaluate machine-learning model to evaluation data

To obtain a quality of trained machine-learning model we can evaluate it by /ml_model/batch/evaluate endpoint.

Example – Request of evaluate some machine learning model according to evaluation data:

```

{
  "alguuid": "01720cf6-d4d4-4e4f-93bc-76de3e08f8c3",
  "config": {},
  "data": {
    "view_id": "demo_driver_behavior_eval_dataset",
    "limit": 5000,
    "page": 0
  }
}

```

Example – Request of evaluate some machine learning model according to evaluation data:


```
{  
  "status": "success",  
  "data": {  
    "score": 0.8614691276516347  
  }  
}
```

3. API

The API url (from now on *api_url*) is always: <https://api.datagora.eu/>

An online reference can be found in <https://swagger.datagora.eu/> under sections:

- Catalogue: signals and measurement channels available
- CIDM data: request of active data requests data received
- Service Providers: requests service provider profile, data requests and data transactions
- Discovery: perform a filtered search of data available in the marketplace
- Analytics: different requests regarding Analytics Toolbox

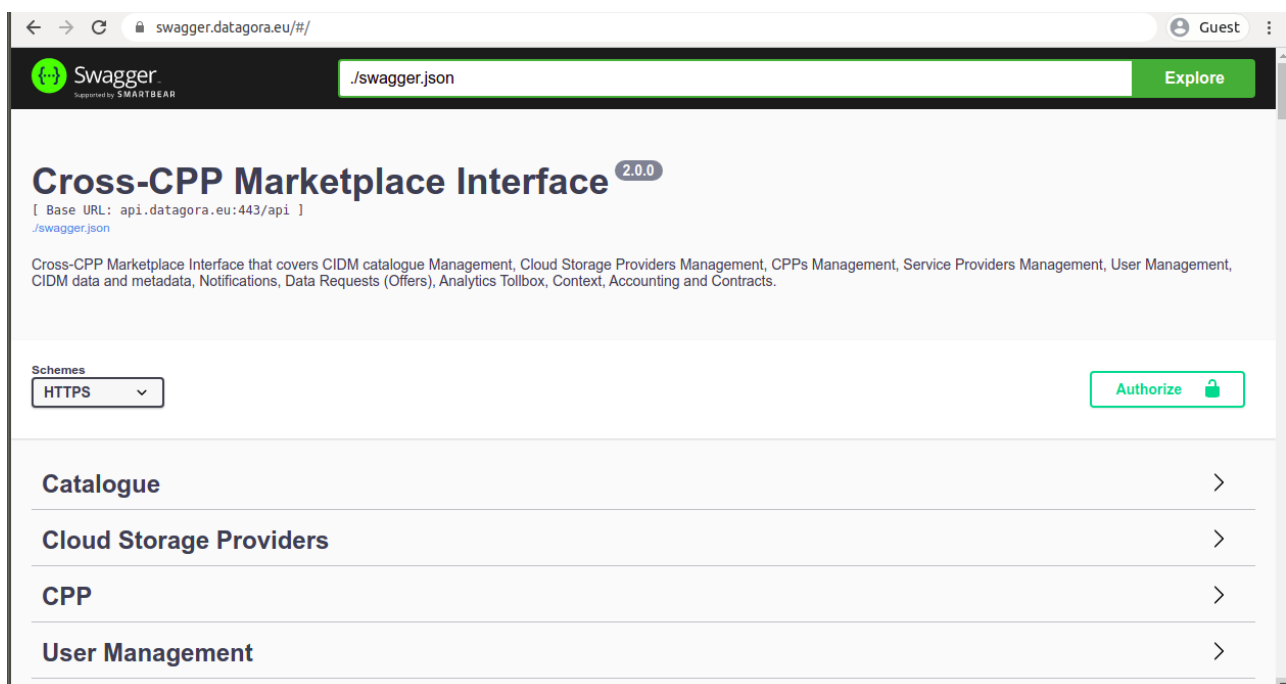


Figure 30. Cross-CPP OpenAPI Specification

3.1. First Steps

3.1.1. Authentication

The Cross-CPP Marketplace API is secured by an Identity Manager (IdM) and OAuth v2 protocols, meaning the endpoints cannot be accessed without first identifying and retrieving an access token.

The IdM has its own server, to secure resource allocation, and therefore has a different url.

Once the token is retrieved it can be used to access the Marketplace resources. All Marketplace API requests must include the following headers:

Parameter	Value	Required	Type	Format	Description
X-Auth-Token	{{access_token}}	Always	string	RFC 6750	OAuth2 token provided by IDM

					component
Content-type	application/json	POST requests	-	-	-

Table 1. Request headers

3.1.2. Get Access Token

This request to the IdM allows the service provider to generate a token to access the Marketplace resources.

To access the IdM resources the request needs the client id and secret that will be provided by Cross-CPP administrators. The Authorization header is a Basic auth generated using the client id and client secret provided.

REQUEST GET ACCESS TOKEN		
Method	POST	
Url	https://idm.datagora.eu	
Endpoint	/oauth2/token	
Headers	Content-Type	application/x-www-form-urlencoded
	Authorization	Basic ...DE4Yg==
Body	grant_type	"password"
	username	string
	password	string

Table 2. Get Access Token

Responses:

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

Request ok response body example:

```

{
  "access_token": "5FOW3KmS7Jv1y3j5ARnabzNryn7kV3",           // string
  "token_type": "Bearer",                                     // string
  "expires_in": 3600,                                         // integer
  "refresh_token": "6pRcNbpaHkMuH2rFRF2feuY1lsCX1L"          // string
}
```

Listing 1. Get Access Token response ok

The "access_token" parameter of the response needs to be stored to be used as an authenticator in every Marketplace request header with key ***X-Auth-Token***.

3.1.3. Service Provider Profile

This request allows the service provider to retrieve its profile, containing the Service provider ID (serviceProviderId from now on), which is needed for most of the service provider requests.

REQUEST GET USER PROFILE		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/UserProfile	
Headers	X-Auth-Token	{{access_token}}

Table 3. Get user profile request

Responses:

Code	Description	
200	OK	Request succesful
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

Request ok fresponse body schema:

```
{
  "serviceProviders": [
    {
      "name": string,
      "email": string,
      "createdAt": date,
      "updatedAt": date,
      "id": id
    }
  ],
  "roles": [
    "serviceProvider"
  ],
  "email": string
}
```

```
}
```

Listing 2. Get user profile response ok

The "serviceProviders.id" parameter of the response needs to be stored to be used as a parameter in most of the Service Provider requests in the Marketplace.

3.2. General Purpose API

This API can be accessed by any authenticated user in the Marketplace. It includes general information functions such as listing available signals and perform a search through the Data Discovery Component.

3.2.1. Catalogue

This section includes functions regarding the retrieval of available signals information.

3.2.1.1. Signal Catalogue

This request allows the user to retrieve the list of currently available signals.

REQUEST		GET CATALOGUE
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/catalogue/signals	
Headers	X-Auth-Token	{{access_token}}

Table 4. Get Catalogue request

Responses:

Code		Description
200	OK	Request succesful
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

Request ok response body example:

```
[
  {
    "id": "----",
    "name": "Vehicle speed",
    // id
    // string
  }
]
```

```

        "type": "numeric", // enumeration string
        "cpp-type": "vehicle", // enumeration string
        "unit": "km/h", // string
        "measurement-channel-id": "1", // string
        "measurement-channel-name": "Vehicle Speed", // string
        "measurement-channel-type": "time-series" // enumeration string
    },
    {
        "id": "---",
        "name": "Latitude",
        "type": "numeric",
        "cpp-type": "vehicle",
        "unit": "o",
        "measurement-channel-id": "2",
        "measurement-channel-name": "Position (Latitude Longitude)",
        "measurement-channel-type": "time-series"
    },
    ...
]

```

Listing 3. Get Catalogue response ok

Each entity in the response array represents a relationship between a signal being sampled by a measurement channel. Both signal and channel ids can be found inside each entity.

3.2.1.2. Signal

This request allows the user to retrieve a Signal details.

REQUEST		GET SIGNAL
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/signal/{{signalId}}	
Headers	X-Auth-Token	{{access_token}}

Table 5. Get single Signal request

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

Request ok response body example for "signalId = 'Vehicle speed' uuid":

```
{
  "channels": [
    {
      "measurement-channel-id": "1",           // string
      "name": "Vehicle Speed",                 // string
      "type": "time-series",                   // enumeration string
      "on-change": false,                      // boolean
      "dimensions": 1,                         // integer
      "format": "Double",                      // string
      "sample-strategy": "last-known-value",   // enumeration string
      "capture-interval": 0,                   // number
      "comment": "...",                       // string
      "createdAt": "2018-03-03T14:50:32.467Z", // date
      "updatedAt": "2018-04-05T15:30:59.811Z", // date
      "id": "---"                             // id
    },
    {
      "measurement-channel-id": "151",
      "name": "Vehicle speed Histogram",
      "type": "histogram",
      "aggregation-strategy": "count",
      "capture-interval": 1,
      "dimensions": 1,
      "comment": "Counter will never be resetted",
      "createdAt": "2018-03-03T14:51:21.943Z",
      "updatedAt": "2018-03-28T09:24:33.203Z",
      "id": "---"
    }
  ],
  "name": "Vehicle speed",                    // string
  "type": "numeric",                          // string
  "unit": "km/h",                             // string
  "sample-rate": "differs",                   // enumeration string
  "createdAt": "2018-03-03T14:50:32.365Z",     // date
  "updatedAt": "2018-04-05T15:30:59.279Z",     // date
  "cpp-type": "vehicle",                      // enumeration string
  "id": "---"                                 // id
}
```

Listing 4. Signal response

A signal entity will be sent with all the channels that sample the data received by the signal.

3.2.1.3. Signals

This request allows the user to retrieve a list of signals.

REQUEST GET SIGNALS	
Method	GET

Url	https://api.datagora.eu/api	
Endpoint	/signal	
Headers	X-Auth-Token	{{access_token}}

Table 6. Get all signals request

Responses:

Code	Description	
200	OK	Request succesful
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

Request ok response body:

Response is an array of signals. See [3.2.1.2 Signal above](#)

3.2.1.4. Channel

This request allows the user to retrieve a Measurement Channel details.

REQUEST GET MEASUREMENT CHANNEL		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/measurementchannel/{{measurement_channel_id}}	
Headers	X-Auth-Token	{{access_token}}

Table 7. Get single measurement channel request

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

Request ok response body example for "measurement-channel-id = 1":

```
{
  "signal": [
    {
      "name": "Vehicle speed",           // string
      "type": "numeric",                 // enumeration string
      "unit": "km/h",                   // string
      "sample-rate": "differs",         // enumeration string
      "createdAt": "2018-03-03T14:50:32.365Z", // date
      "updatedAt": "2018-04-05T15:30:59.279Z", // date
      "cpp-type": "vehicle",            // enumeration string
      "id": "---"                       // id
    }
  ],
  "measurement-channel-id": "1",        // string
  "name": "Vehicle Speed",              // string
  "type": "time-series",                // enumeration string
  "on-change": false,                   // boolean
  "dimensions": 1,                      // integer
  "format": "Double",                   // string
  "sample-strategy": "last-known-value", // enumeration string
  "capture-interval": 0,                 // number
  "comment": "...",                     // string
  "createdAt": "2018-03-03T14:50:32.467Z", // date
  "updatedAt": "2018-04-05T15:30:59.811Z", // date
  "id": "---"                           // id
}
```

Listing 5. Channel response

3.2.1.5. Channels

This request allows the user to retrieve a list of signals.

REQUEST GET MEASUREMENT CHANNELS		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/measurementchannel	
Headers	X-Auth-Token	{{access_token}}

Table 8. Get all channels request

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

Request ok response body:

Response is an array of measurement channels. See [3.2.1.4 Channel](#) above

3.2.2. Data Discovery

This request allows the user to get a resume of available data within a configuration of filters.

REQUEST DATA DISCOVERY		
Method	POST	
Url	https://api.datagora.eu/api	
Endpoint	/VehicleDataDiscovery	
Headers	X-Auth-Token	{{access_token}}
	Content-type	application/json
Body	<pre>{ "channels": [], // string array "submissionDate": {}, // Object "min": "", // Number "max": "", // Number }, "travelDates": {}, // Object "min": "", // Number "max": "", // Number } "travelledDuration": {}, // Object "min": "", // Number "max": "", // Number } "geoBoundingBox": {}, // Object "latitude-min": "", // Date "longitude-min": "", // Date "latitude-max": "", // Date "longitude-max": "", // Date } "includeContextData": false, // boolean "basicCppInformationFilters": [] // Object array }</pre>	

Table 9. Data Discovery request

Body explanation:

** for every filter: if sent empty that filter will be count as NO FILTER (retrieve all)*

- **channels:** Array of "measurement-channel-id". Filters data packages to those from selected channels.
- **submissionDate:** Object representing the dates in which the data has been stored in the marketplace, including max and min values between which data will be searched.
- **travelDates:** Object representing the dates in which the data has been generated by the signal, including max and min values between which data will be searched.
- **travelledDuration:** Object representing the duration in seconds of each data package, including max and min values between which data will be searched.
- **geoBoundingBox:** Object representing the geographical rectangular area within which the data is desired, including longitude and latitude max and min values.
- **includeContextData:** Boolean activating the receiving of additional context data.
- **basicCppInformationFilters:** Array of objects representing additional filters based on basic-cpp-information signals, otherwise known as CPP metadata. These objects include the channel, name and value of the filters, such as "red car" (see example)

Body example:

This example represents a search for data packages belonging to:

- Vehicle speed and location
- Taken and submitted in August 2019
- Not longer than 3 seconds
- Within the geographical area of Austria
- Without context data

```
"channels": [
  "1",
  "2"
],
"submissionDate": {
  "min": "2019-08-01",
  "max": "2019-08-31"
},
"travelDates": {
  "min": "2019-08-01",
  "max": "2019-08-31"
},
"travelledDuration": {
  "min": 0,
  "max": 3
},
"geoBoundingBox": {
  "longitude-min": 9.48,
  "latitude-min": 46.43,
  "longitude-max": 16.98,
  "latitude-max": 49.04
},
"includeContextData": false,
"basicCppInformationFilters": [
  {
    "channel": {
```

```

        "id": "---",
        "name": "Vehicle Colour",
        "type": "information",
        "cpp-type": "vehicle",
        "measurement-channel-id": "702",
        "measurement-channel-name": "Vehicle Colour",
        "measurement-channel-type": "basic-cpp-information"
    },
    "name": "Vehicle Colour",
    "cpp-type": "vehicle",
    "measurement-channel-id": "702",
    "value": "red"
}
]
}

```

Listing 6. Discovery request example

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

Request ok response body with no data:

```

{
  "data-packages": 0
}

```

Request ok response body example with data:

```

{
  "data-packages": 15733, // integer
  "rooms": 0, // integer
  "data-packages-trip": 1000, // integer
  "trips": 1000, // integer
  "cpp-owners": 8, // integer
  "duration-histogram": {
    "min-duration": 0, // integer
    "max-duration": 3844800, // integer
    "bins-size": 3600, // integer
    "bins": [ ... ], // integer array
  },
  "channel-results": [ // Object array
    {
      "measurement-channel-id": "2", // string
      "count": 2930 // integer
    },
    ...
  ],
}

```

```

"cpp-type-results": [                                // Object array
  {
    "minDuration": 1,                                // integer
    "maxDuration": 1527,                             // integer
    "avgDuration": 478.2052477357485,                // double
    "cpp-type": "vehicle",                           // enumeration string
    "count": 15016                                   // integer
  },
  { ... }
],
"room-results": [],                                  // Object array
"cpptype-channel-results": [
  {
    "channel": [                                      // Object array
      {m
        "measurement-channel-id": "1",               // string
        "count": 2813                                // integer
      },
      { ... }
    ],
    "cpp-type": "vehicle",                           // string
    "count": 15016                                   // integer
  },
  { ... },
],
"total-channel-results": 25,                         // integer
"total-cpp-type-results": 2,                         // integer
"total-room-results": 0,                             // integer
"time-distribution": {
  "min-timestamp-start": "2017-10-03T13:03:13.000Z", // date
  "max-timestamp-stop": "2019-07-17T14:30:02.000Z", // date
  "histogram": {
    "timestamp-start": "2017-10-03T00:00:00.000Z",   // date
    "timestamp-stop": "2019-07-17T23:59:59.000Z",   // date
    "bins-size": 86400000,                           // integer
    "bins": [ ... ],                                 // integer array
  }
},
"geo-distribution": {
  "latitude-min": 48.599998474121094,                // double
  "latitude-max": 51.522789001464844,                // double
  "longitude-min": 3,                                // double
  "longitude-max": 18.35022735595703,
  "heatmap": {
    "latitude-min": 48.5,                             // double
    "latitude-max": 51.6,                             // double
    "longitude-min": 3,                                // double
    "longitude-max": 18.400000000000002,               // double
    "bins-size": 0.1,                                 // double
    "bins": [                                          // Object array
      {
        "latitude": 51.48193359375,                   // double
        "longitude": 7.44873046875,                   // double
        "count": 4848                                 // integer
      },
      { ... }
    ]
  }
}

```

```

    }
  }
}

```

Listing 7. Discovery response example

3.2.3. Channel Suggestions

This request allows the user to get a list of suggested channels based on the already selected channels during data discovery process.

REQUEST GET ACCESS TOKEN		
Method	POST	
Url	https://api.datagora.eu/api	
Endpoint	/suggestion/suggestions	
Headers	X-Auth-Token	{{access_token}}
	Content-type	application/json
Body	<pre> { "channels": [], // string array } </pre>	

Table 10. Get channel suggestions

Body explanation:

**if sent empty, no results will be given*

Channels: Array of "measurement-channel-id". Filters data packages to those from selected channels.

Body example:

This example partly represents a request for suggestions related to:

- Vehicle speed

```

{
  "channels": [
    "1"
  ],
}

```

Responses:

Code	Description	
200	OK	Request succesful
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

Request ok response body:

```
[
  {
    "requestedSignal": "VehicleSpeed",           // string
    "relatedSignals": [                         // signal-channel array
      {
        "signal-type": "Sensor Measurement Data", // string
        "cpp-type": "Vehicle",                   // string
        "signal-name": "Wheel RPM - front left",  // string
        "measurement-channel-id": "115",          // string
        "measurement-channel-type": "time-series", // string
        "measurement-channel-name": "Wheel RPM - front left" // string
      },
      {
        "signal-type": "Sensor Measurement Data",
        "cpp-type": "Vehicle",
        "signal-name": "Wheel RPM - front right",
        "measurement-channel-id": "116",
        "measurement-channel-type": "time-series",
        "measurement-channel-name": "Wheel RPM - front right"
      },
      { ... }
    ]
  }
]
```

Listing 8. Channel suggestions response

3.3. Service Provider API

This API can be accessed by Service Providers authenticated in the Marketplace. It includes Data Requests, Data Transactions and Analytics related requests that allow Service Providers to consult, review and manage their data.

3.3.1. Data Requests

Data Requests are the main point for Service Providers. Through them all available data fitting each one filter configuration is sent.

3.3.1.1. Create Data Request

To create a Data Request a Service Provider should perform a Data Discovery first in order to check if results fits with what is desired, although a Data Request can be created even if no data is retrieved with the Data Discovery.

REQUEST CREATE DATA REQUEST		
Method	POST	
Url	https://api.datagora.eu/api	
Endpoint	/ServiceProviderOffer	
Headers	X-Auth-Token	{{access_token}}
	Content-type	application/json
Body	<pre>{ "description": "", // string "serviceProvider": "", // uuid "include-context": true, // boolean "include-analytics": true, // boolean "vehicleDataRequest" {, // data discovery model "channels": [], // string array "submissionDate": {}, // Object "travelDates": {}, // Object "travelledDuration": {}, // Object "geoBoundingBox": {}, // Object "includeContextData": false, // boolean "BasicCppInformationFilters": [] // Object array } }</pre>	

Table 11. Create Data Request request

Body explanation:

- **description:** the name / short description so data owners know in a glance what it might be requesting
- **serviceProvider:** identifier of the Service Provider
- **include-context:** boolean stating whether the data collected from this request includes context data or not
- **include-analytics:** boolean stating whether the data collected from this request will be used in the creation of analytics or not
- **vehicleDataRequest:** use Data Discovery model.

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

Request ok response body:

```
{
  "vehicleDataRequest": "---- ",           // id
  "serviceProvider": "----",               // id
  "description": "Postman test offer",      // string
  "include-context": true,                 // string
  "include-analytics": true,               // string
  "maxNumberOfDataSources": 0,             // integer
  "createdAt": "2019-09-23T14:52:35.439Z", // string
  "updatedAt": "2019-09-23T14:52:35.439Z", // string
  "id": "----",                           // id
  "aeonChannelId": "----",                 // id
  "aeonSubscriptionUrl": "https://{aeon_url}/subscribe/{uuid}", // string
  "aeonPublicationUrl": "https://{aeon_url}/publish/{uuid}"     // string
}
```

Listing 9. Create data request response

3.3.1.2. Delete Data Request

This method can be used to delete an existing Data Request. This action cannot be undone and will prevent the Service Provider to receive any more data from users that had accepted the request before deleting it.

REQUEST		DELETE DATA REQUEST	
Method		DELETE	
Url		https://api.datagora.eu/api	
Endpoint		/ServiceProviderOffer/{ serviceProviderOfferId }	
Headers		X-Auth-Token	{{access_token}}

Table 12. Delete Data Request request

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

3.3.1.3. Get my Data Requests

This request can be used to retrieve all active Data Request belonging to the Service Provider

REQUEST GET SERVICE PROVIDER DATA REQUESTS		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/ServiceProvider/{{serviceProviderId}}/offers	
Headers	X-Auth-Token	{{access_token}}

Table 13. Get Data Requests request

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

Request ok response body:

```
[
  {
    "id": "---",
    "description": "description",
    "vehicleDataRequest": {
      "id": "---"
      "channels": [],
      "submissionDate": {},
      "travelDates": {},
      "travelledDuration": {},
      "geoBoundingBox": {},
    },
    "serviceProvider": {
      "id": "---",
      "name": "",
      "email": "@.",
    },
    "createdAt": "2019-05-09T12:50:00.638Z",
    "updatedAt": "2019-05-09T12:50:00.926Z",
    "maxNumberOfDataSources": 0,
    "include-context": true,
    "include-analytics": true,
    "aeonChannelId": "---",
  }
]
```

```

    "aeonSubscriptionUrl": "https://{aeon_url}/subscribe/{uuid}", // string
    "aeonPublicationUrl": "https://{aeon_url}/publish/{uuid}"      // string
  },
  { ... }
]

```

Listing 10. Get my Data Requests

3.3.1.4. Get a Data Request

This request can be used to retrieve a Data Request details

REQUEST		GET DATA REQUEST DETAILS	
Method	GET		
Url	https://api.datagora.eu/api		
Endpoint	/ServiceProviderOffer/{{spOfferId}}		
Headers	X-Auth-Token	{{access_token}}	

Table 14. Get Data Request details request

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

Request ok response body:

```

{
  "id": "---",
  "description": "test for tables",
  "serviceProvider": {
    "name": "name",
    "email": "@.",
    "id": "---"
  },
  "createdAt": "2019-06-27T11:21:13.903Z",
  "updatedAt": "2019-06-27T11:21:14.273Z",
  "aeonChannelId": "---",
  "aeonSubscriptionUrl": "https://{aeon_url}/subscribe/{uuid}",
  "aeonPublicationUrl": "https://{aeon_url}/publish/{uuid}",
  "include-context": true,
}

```

// id
 // string
 // service provider object

 // date-time
 // date-time
 // id
 // string
 // string
 // boolean

```

"include-analytics": true,                                     // boolean
"vehicleDataRequest": {                                     // filter config object
  "channels": [                                             // string array
    "1",
    "2"
  ],
  "travelledDuration": {
    "min": -,                                             // date *
    "max": -                                             // date *
  },
  "submissionDate": {
    "min": "2019-06-03",                                   // date *
    "max": "2019-06-27"                                   // date *
  },
  "travelDates": {
    "min": -,                                             // date *
    "max": -                                             // date *
  },
  "geoBoundingBox": {
    "latitude-max": 58,                                   // number **
    "latitude-min": 38,                                   // number **
    "longitude-max": 25,                                   // number **
    "longitude-min": -5                                   // number **
  }
},
},
}

```

Listing 11. Get Data Request details

* If this value is null or undefined the object will be sent but will not contain the key

** If any of the values is null or undefined, the object will be sent empty, as if there is no filter

3.3.1.5. Get Data Request Contracts

This request can be used to retrieve currently active contracts for an active data request (acceptance of the data request from data owners)

REQUEST GET DATA REQUEST CONTRACTS		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/ServiceProviderOffer/{spOfferId}/contracts	
Headers	X-Auth-Token	{{access_token}}

Table 15. Get Data Request contracts request

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

Request ok response body:

```
[
  {
    "id": "---",
    "spOffer": "---",
    "effectiveDate": "2019-07-03T09:17:38.492Z",
    "createdAt": "2019-07-03T09:17:37.787Z",
    "updatedAt": "2019-07-03T09:17:37.787Z",
    "vehicleOwner": {
      "id": "---",
      "name": "Vehicle1",
      "email": "@.",
      "cloudVaultId": "---",
      "readAccessKey": "---",
      "discoveryClearance": true,
      "testDataClearance": false,
      "aggregationServiceDataClearance": false,
      "createdAt": "2019-04-05T11:58:13.188Z",
      "updatedAt": "2019-04-05T11:58:13.188Z",
      "cloudStorageProvider": {
        "cloudBaseUrl": "https://url.domain",
        "accessToken": "WT4mRvs5w6a7W3KbZN7VeNap",
        "name": "Company Cloud Storage",
        "contactEmail": "@.",
        "createdAt": "2019-04-04T12:11:29.531Z",
        "updatedAt": "2019-04-04T12:11:29.531Z",
        "id": "5ca5f471c216151d00fe13e3"
      }
    }
  },
  { ... }
]
```

Listing 12. Get Data Request contracts

3.3.2. Data

Data Requests are sets of configured filters defining which data a service provider will receive through them. The Service Provider can query a resume of this data at any time.

3.3.2.1. Get Data Transactions

This request can be used to retrieve a list of data transactions belonging to the service provider data requests

REQUEST GET DATA TRANSACTIONS		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/ServiceProvider/{{serviceProviderId}}/dataTransactions	
Headers	X-Auth-Token	{{access_token}}

Table 16. Get Data Transactions resume request

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

Request ok response body:

```
[
  {
    "serviceProviderOffer": "---",
    "serviceProvider": "---",
    "method": "pull",
    "datapackage-id": "---",
    "accessDatetime": "2019-05-13T13:04:37.473Z",
    "size": 1976,
    "datapackageHash": "----",
    "createdAt": "2019-05-13T13:04:37.478Z",
    "updatedAt": "2019-05-13T13:04:37.478Z",
    "id": "5cd96b65ff89151c002d16b3"
  },
  { ... }
]
```

Listing 13. Get transactions resume

3.3.2.2. Get Data Request Data Transactions

This request can be used to retrieve a list of data transactions belonging to a concrete service provider data request

REQUEST GET DATA REQUEST TRANSACTIONS	
Method	GET

Url	https://api.datagora.eu/api	
Endpoint	/ServiceProviderOffer/{{spOfferId}}/dataTransactions	
Headers	X-Auth-Token	{{access_token}}

Table 17. Get Data Request transactions request

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

Request ok response body:

```
[
  {
    "serviceProviderOffer": "---",
    "serviceProvider": "---",
    "method": "pull",
    "datapackage-id": "---",
    "accessDatetime": "2019-05-13T13:04:37.473Z",
    "size": 1976,
    "datapackageHash": "----=",
    "createdAt": "2019-05-13T13:04:37.478Z",
    "updatedAt": "2019-05-13T13:04:37.478Z",
    "id": "5cd96b65ff89151c002d16b3"
  },
  { ... }
]
```

Listing 14. Get Data Request transactions

3.3.2.3. Get Data Package Details

This request can be used to retrieve the details of a single data package.

REQUEST	GET DATA PACKAGE DETAILS
Method	GET
Url	https://api.datagora.eu/api

Endpoint	/ServiceProviderOffer/{{spOfferId}}/CvimDataPackages/{{dataPackageId}}	
Headers	X-Auth-Token	{{access_token}}

Table 18. Get Data Package details request

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

Request ok response body:

```
{
  "timestamp-start": "2017-10-03T13:03:13Z",           // date-time
  "cvim-version": "1.2.0",                             // string
  "timestamp-stop": "2017-10-03T13:03:19Z",           // date-time
  "cpp-type": "vehicle",                               // string
  "measurement-channel-id": "1",                      // string
  "geo-bounding-box": {
    "latitude-max": 48.804771423339844,                // number
    "longitude-max": 3.060849905014038,                // number
    "latitude-min": 48.804771423339844,                // number
    "longitude-min": 3.060849905014038                 // number
  },
  "data": [                                             // values in the package
    {
      "value": 102,                                    // number
      "timestamp": "2017-10-03T13:03:13Z"              // date-time
    },
    {
      "value": 100,
      "timestamp": "2017-10-03T13:03:16Z"
    },
    { ... }
  ],
  "submit-time": "2019-04-30T15:17:47.941912Z",        // date-time
  "type": "time-series",                              // string
  "datapackage-id": "b72dfe16-b53d-4d69-ba3f-0976289cda4f", // uuid
  "number-of-samples": 3,                             // number
  "signatures": [                                     // security signatures array
    {
      "signatory": "MARKETPLACE",                     // string
      "checksum": "---",                              // encoded string
      "signature": "-----",                         // encoded string
      "marketplace-validation": {
        "integrity": true,                            // boolean

```



```

        "authenticated": false                                // boolean
    }
},
"marketplace-validation": {
    "integrity": true,                                       // boolean
    "authenticated": false                                   // boolean
}
}

```

Listing 15. Get data package details

3.3.2.4. Get Data Request received Data Packages

This request can be used to retrieve a list of data packages received through a concrete data request.

REQUEST		GET DATA REQUEST DATA PACKAGES
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/ServiceProviderOffer/{{serviceProviderOfferId}}/CvimDataPackages	
Headers	X-Auth-Token	{{access_token}}

Table 19. Get Data Packages received for a Data Request request

Additional parameters:

This request accepts additional parameters in the request url to act as filters of the amount of data packages to receive:

Parameter	Value	Description
Limit	Integer	Number of data packages to be received. If not provided, default value is 1000. Maximum value is 1000.
From	Integer	If provided, it defines the offset from the first result you want to fetch. If not provided, default value is 0
Page	Integer	If provided, it defines the first page you want to fetch. If not provided, default value is 0

Sort	String	If provided, it defines the field and order to sort results
Order	String	If provided, it defines order to sort results

Table 20. Query Parameters

Request example:

api.datagora.eu/api/ServiceProviderOffer/{{serviceProviderOfferId}}/CvimDataPackages?limit=100&page=1&order=asc

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

Request ok response body:

[{ **DataPackage** }]

Response is an array of data packages. See [3.3.2.3 Get Data Package Details above](#)

3.3.2.5. Get Data Request received Metadata Packages

This request can be used to retrieve a list of metadata packages received from a data request.

REQUEST	GET DATA REQUEST METADATA PACKAGES	
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/ServiceProviderOffer/{{serviceProviderOfferId}}/CvimMetadata	
Headers	X-Auth-Token	{{access_token}}

Table 21. Get Metadata packages received for a Data Request request

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

Request ok response body:

```
{
  "total": 342,                                // integer
  "size": 342,                                // integer
  "metadata": [                               // metadata packages array
    {
      "datapackage-id": "---",                 // uuid
      "cpp-type": "vehicle",                   // string
      "timestamp-start": "2017-10-03T13:03:13Z", // date-time
      "submit-time": "2019-04-30T15:17:47.941912Z", // date-time
      "number-of-samples": 3,                  // integer
      "cvim-version": "1.2.0",                 // string
      "type": "time-series",                   // string
      "timestamp-stop": "2017-10-03T13:03:19Z",
      "geo-bounding-box": {
        "latitude-max": 48.804771423339844,    // number
        "longitude-min": 3.060849905014038,    // number
        "latitude-min": 48.804771423339844,    // number
        "longitude-max": 3.060849905014038     // number
      },
      "measurement-channel-id": "1",           // string
      "duration": 6,                           // integer
      "marketplace-tags": {
        "location": [
          "u0djt"
        ],
        "geoHashPrecision": 5                  // integer
      }
    },
    { ... }
  ]
}
```

Listing 16. Get Data Request Metadata

3.4. Toolbox API

3.4.1. Data Views

Tool that allows the allocating of certain data through a set of specific filters including signal values, and that allows to categorize the results.

3.4.1.1. Create Data View

REQUEST CREATE NEW DATA VIEW		
Method	POST	
Url	https://api.datagora.eu/api	
Endpoint	/DataView	
Headers	X-Auth-Token	{{access_token}}
	Content-type	application/json
Body	<pre>{ "offer": "5d1da759f4cd991e00d0a308", // id "channels": ["602", "610", "604"], // array "filters": [], // object array "category": "5f92c00d9f2b082100062ccc", // id (optional) "name": "Weather control", // string "description": "Assess outside weather" // string }</pre>	

Table 22. Create new Data View

Body explanation:

- **offer**: id of the data request from which data will be collected
- **channels**: array of measurement channels from which data received will be collected
- **filters** *: Array of objects that the data from the selected channels that will be collected.
- **category**: Id of the assigned category. Can be assigned later.
- **name**: String to identify the Data View.
- **description**: String defining the purpose of the Data View.

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

Request ok response body:

See [3.4.1.3 Get Data View details below](#)

3.4.1.2. Get Data Views list

This request can be used to retrieve a list of your requested Time Series Analytics.

REQUEST GET DATA VIEWS LIST		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/ServiceProvider/{{serviceProviderId}}/DataView	
Headers	X-Auth-Token	{{access_token}}

Table 23. Get list of active Data Views

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

Request ok response body:

Response is an array of Data Views. See [3.4.1.3 Get Data View details](#) below

3.4.1.3. Get Data View details

This request can be used to get a DataView details.

REQUEST GET DATA VIEW DETAILS		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/DataView/{{DataViewId}}	
Headers	X-Auth-Token	{{access_token}}

Table 24. Get Data View details

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

Request ok response body:

```

{
  "name": "Weather control", // string
  "description": "Assess outside weather", // string
  "offer": {...}, // offer object
  "channels": ["602", "610", "604"], // array
  "filters": [], // array
  "outputAeonSubscriptionUrl": "https://{aeon_url}/subscribe/{uuid}", // string
  "category": {
    "name": "weather", // string
    "enum": [...] // string array
  },
  "createdAt": "2020-10-23T11:37:27.568Z", // date-time
  "updatedAt": "2020-10-23T11:37:27.978Z", // date-time
}

```

Listing 17. Data View response

3.4.1.4. Retrieve Data View

This request returns the latest data collected in the Data View in form of rows. Result is an array of the latest rows collected, up to 100.

REQUEST	RETRIEVE DATA VIEW COLLECTED DATA	
Method	POST	
Url	https://api.datagora.eu/api	
Endpoint	/DataView/{{DataViewId}}/retrieve	
Headers	X-Auth-Token	{{access_token}}
	Content-type	application/json
Body	{}	

Table 25. Retrieve data view collected data

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

Request ok response body:

```

{
  "status": "success",           // string
  "data": [                     // rows array
    {
      "#": 1,                   // integer (row number)
      "608": 11,                // channel value type
      "900": null,
      "charger_availability": "low" // enum string (category value)
      "tstamp": "2020-06-30T02:30:00+02:00", // string
    },
    { ... }
  ],
}
```

Listing 18. Data View retrieve response

Please note that the category key-pair will only be sent once a category is assigned to the Data View.

3.4.1.5. Data View category

The next requests are used to create and manage category and the assignment of these to Data Views and their collected rows.

3.4.1.5.1. Create new category

This request is used to create a category. Categories are stored without being assigned to a Data View to be used as many times the Service Provider need.

REQUEST	CREATE CATEGORY
Method	POST

Url	https://api.datagora.eu/api	
Endpoint	/SEIcategory	
Headers	X-Auth-Token	{{access_token}}
	Content-type	application/json
Body	<pre>{ "serviceProvider": "5cd421a79c0f591e006515a5", "name": "charger availability", "enum": ["low", "normal", "high",], }</pre>	

Table 26. Create category request

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

Request ok response body:

See [3.4.1.5.3 Get Category details below](#)

3.4.1.5.2. Get categories list

This request can be used to retrieve the list of your created categories.

REQUEST	GET DATA VIEWS LIST	
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/ServiceProvider/{{serviceProviderId}}/SEIcategory	
Headers	X-Auth-Token	{{access_token}}

Table 27. Get list of created categories

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

Request ok response body:

Response is an array of categories. See [3.4.1.5.3 Get Category details](#) below

3.4.1.5.3. Get Category details

This request can be used to get a Category details.

REQUEST	GET CATEGORY DETAILS	
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/SElcategory/{{categoryId}}	
Headers	X-Auth-Token	{{access_token}}

Table 28. Get Category details

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

Request ok response body:

```
{
```

```

    "serviceProvider": {...},                                // service provider object
    "id": "5f7f2fc48d3e0323004fcdc7",                       // string
    "name": "charger availability",                          // string
    "enum": [                                                // string array
        "low",
        "normal",
        "high"
    ],
    "createdAt": "2020-10-08T15:27:00.953Z",                // date-time
    "updatedAt": "2020-10-08T15:27:00.953Z",                // date-time
}

```

Listing 19. Get categories list response

3.4.1.5.4. Assign Data View category

This request assigns a category to an uncategorized Data View. Once assigned the /retrieve response will include the category as an additional key-value in each row.

REQUEST			ASSIGN CATEGORY TO DATA VIEW	
Method	POST			
Url	https://api.datagora.eu/api			
Endpoint	/DataView/{{DataViewId}}/category			
Headers	X-Auth-Token	{{access_token}}		
	Content-type	application/json		
Body	{ "category_id": "5f7f2fc48d3e0323004fcdc7", // string }			

Table 29. Assign category to an uncategorized Data View

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

3.4.1.5.5. Assign category values

This request returns the latest data collected in the Data View in form of rows.

REQUEST ASSIGN CATEGORY VALUES TO DATA VIEW ROWS		
Method	POST	
Url	https://api.datagora.eu/api	
Endpoint	/DataView/{{DataViewId}}/assign	
Headers	X-Auth-Token	{{access_token}}
	Content-type	application/json
Body	<pre>{ "row": "value", // key value pair ... }</pre>	

Table 30. Assign category values to Data View rows

Body explanation:

- **row**: number of the row (#). This value can be found at each row of the /retrieve response
- **value**: category value. This value must be a string in the category enum.

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

3.4.2. Analytics

Analytics are requests for analytical measurements of the data available for a set of filters, including specific data requests and measurement channels.

These analytics can be created by Service Providers with the data received from created data requests (see Create Data Request) with the **include-analytics** parameter set to **true**. To be sure which data requests are eligible for analytics perform a query for service providers data requests including the filter for said attribute:

<https://api.datagora.eu/api/ServiceProvider/{{serviceProviderId}}/offers?include-analytics=true>

Please, take into consideration that although all analytics has a similar model, some attributes may differ.

Currently available analytics types:

- Time Series
- Trajectory Analysis
- Networks
- Machine Learning

3.4.2.1. Time Series Analytics

Toolbox devoted to the analysis of time series, with a special focus on drift detection.

The identification of drifts, i.e. sudden changes in the dynamics, has numerous applications. For instance, a user may decide when to access a given stream of data, like only when significant changes have been detected

3.4.2.1.1. Create new Time Series Analytics

REQUEST CREATE NEW TIME SERIES ANALYTICS		
Method	POST	
Url	https://api.datagora.eu/api	
Endpoint	/analytics/TimeSeries	
Headers	X-Auth-Token	{{access_token}}
	Content-type	application/json
Body	<pre>{ "description": "", // string "model": "", // string "offer": "", // id "channels": "", // string "submissionDate": {}, // Object "travelDates": {}, // Object "travelledDuration": {}, // Object "geoBoundingBox": {}, // Object "variables": {}, // Object }</pre>	

Table 31. Create new Time Series analytics

Body explanation:

- **description:** descriptive name of the analytics.
- **model **:** type of Time Series analytics.
- **offer:** id of the data request from which data will be analyzed

- **channels**: array of measurement channels from which data received will be analyzed
- **submissionDate** *: Object representing the dates in which the data has been stored in the marketplace, including max and min values between which data will be searched.
- **travelDates** *: Object representing the dates in which the data has been generated by the signal, including max and min values between which data will be searched.
- **travelledDuration** *: Object representing the duration in seconds of each data package, including max and min values between which data will be searched.
- **geoBoundingBox** *: Object representing the geographical rectangular area within which the data is desired, including longitude and latitude max and min values.
- **variables** **: Object containing the different variables needed for each type of time series model. All variables of each model are mandatory.

* for every filter: if sent empty that filter will be count as NO FILTER (retrieve all)

** refer to subsections for special treatment depending on the Time Series model

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

Request ok response body:

See [3.4.2.1.3 Get Time Series Analytics details below](#)

3.4.2.1.1.1. SampleEntropy

Probabilistic measure of the complexity of a time series.

Given a specific embedding dimension e and tolerance t , the algorithm returns the probability (as a negative logarithm) that if two sets of e simultaneous data points have distance t , two sets of $(e+1)$ simultaneous data points will also have distance t .

Variables

- **embeddingDimension**: (numeric - integer) number of points to consider in each set.
- **tolerance**: (numeric - integer) estimated distance for each set of simultaneous data.
- **normalized**: (boolean) whether or not to normalize the time series' values.

3.4.2.1.1.2. PermutationEntropy

Measure of the complexity of a time series.

Given the ordinal pattern of sequential sets of e points (each spaced by d points from the other) in a time series, it measures the variability of its behavior according to all possible changes in each set's ordinal pattern.

Variables

- **embeddingDimension:** (numeric - integer) number of points to consider in each set.
- **delay:** (numeric - integer) number of spacing points between sets

3.4.2.1.1.3. Irreversibility

Quantitative measure of the behavior of a time series when its sequence of timestamps is reversed. Computed over sets of data points.

Variables

- **embeddingDimension:** (numeric - integer) number of points to consider in each reversed set.

3.4.2.1.1.4. PearsonCorrelation

Correlation coefficient formulas are used to find how strong a relationship is between data. The formulas return a value between -1 and 1, where:

- 1 indicates a strong positive relationship.
- -1 indicates a strong negative relationship.
- A result of zero indicates no relationship at all.

Pearson correlation is the most used measure of correlation in science. It quantifies the strength of the association of two variables. To obtain meaningful results the data must fulfill the following requirements:

- Both variables should be normally distributed.
- Absence of outliers. These are values which lie an abnormal distance from other values.
- Variables should be continuous.
- Variables must have a linear relationship.
- Homocedasticity: equal variance across variables.

Mandatorily needs at least 2 trajectories to be provided in the data field.

Variables

None

3.4.2.1.1.5. SpearmanCorrelation

Correlation coefficient formulas are used to find how strong a relationship is between data. The formulas return a value between -1 and 1, where:

- 1 indicates a strong positive relationship.
- -1 indicates a strong negative relationship.
- A result of zero indicates no relationship at all.

Spearman's coefficient measures the rank correlation between two functions. To properly understand this correlation, it is necessary to know what a monotonic function is. A monotonic function is one that either never increases or never decreases as its independent variable increases.

Spearman's correlation coefficient is a statistical measure of the strength of a monotonic relationship between paired data.

Mandatorily needs at least 2 trajectories to be provided in the data field.

Variables

None

3.4.2.1.1.6. RegressionTree

A decision tree can be described as a flowchart where each internal node describes a test on a learning variable, each branch represents a result of the test and each sheet contains the value of the numeric variable.

Variables

- **forecast:** (numeric - integer) number of new data points to forecast.
- **division:** (numeric - integer) index of the time series point at which the division between training and test sets is to be performed.

3.4.2.1.1.7. NeuralNetwork

Multilayer Perceptron-based prediction of new data points for a given time series.

Variables

- **forecast:** (numeric - integer) number of new data points to forecast.
- **division:** (numeric - integer) index of the time series point at which the division between training and test sets is to be performed.

3.4.2.1.1.8. Arima

Time series forecasting method based on the following principles:

- Autoregression (AR): uses the dependent relationship between an observation and some number of lagged observations.
- Integrated (I): differencing of raw observations ($obs_i - obs_{i-1}$) to make the series stationary.
- Moving Average (MA): uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Variables

- **forecast**: (numeric - integer) number of new data points to forecast.
- **alpha**: (numeric) value for the statistical significance (of the response variable) of the hypothesis test conducted within the model (0.05 is recommended as a widespread practice).
- **arimaOrder**:
 - o **p**: (numeric - integer) related to AR's regular part. Number of lag observations included in the model.
 - o **d**: (numeric - integer) related to I's regular part. Number of times that the raw observations are differenced.
 - o **q**: (numeric - integer) related to Q's regular part. Size of the moving average window.
 - o **seasonalP**: (numeric - integer) related to AR's stationary part. Number of lag observations included in the model.
 - o **seasonalD**: (numeric - integer) related to I's stationary part. Number of times that the raw observations are differenced.
 - o **seasonalQ**: (numeric - integer) related to Q's stationary part. Size of the moving average window.

3.4.2.1.2. Get Time Series Analytics list

This request can be used to retrieve a list of your requested Time Series Analytics.

REQUEST		GET TIME SERIES ANALYTICS LIST	
Method	GET		
Url	https://api.datagora.eu/api		
Endpoint	/ServiceProvider/{{serviceProviderId}}/analytics/TimeSeries		
Headers	X-Auth-Token	{{access_token}}	

Table 32. Get list of requested Time Series analytics

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

Request ok response body:

Response is an array of Time Series analytics. See [3.4.2.1.3 Get Time Series Analytics details](#) below

3.4.2.1.3. Get Time Series Analytics details

This request can be used to retrieve a list of your requested Time Series Analytics.

REQUEST GET TIME SERIES ANALYTICS DETAILS		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/analytics/TimeSeries/{{analyticsId}}	
Headers	X-Auth-Token	{{access_token}}

Table 33. Get list of requested Time Series analytics

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

Request ok response body:

```
{
  "id": "----",
  // id
```

```

"model": "---", // string
"description": "my analytics", // string
"serviceProvider": { }, // service provider object
"offer": { }, // data request object
"channels": [{ }], // channel array
"createdAt": "2019-05-20T09:20:54.222Z", // date-time
"updatedAt": "2019-05-20T09:20:54.499Z", // date-time
"inputAeonChannelId": "---", // id
"inputAeonSubscriptionUrl": "https://{aeon_url}/subscribe/{uuid}", // string
"outputAeonChannelId": "---", // id
"outputAeonPublicationUrl": "https://{aeon_url}/publish/{uuid}", // string
"outputAeonSubscriptionUrl": "https://{aeon_url}/subscribe/{uuid}" // string
"travelledDuration": {
  "min": -, // date *
  "max": - // date *
},
"submissionDate": {
  "min": "2019-06-03", // date *
  "max": "2019-06-27" // date *
},
"travelDates": {
  "min": -, // date *
  "max": - // date *
},
"geoBoundingBox": {
  "latitude-max": 58, // number **
  "latitude-min": 38, // number **
  "longitude-max": 25, // number **
  "longitude-min": -5 // number **
},
"variables": { }, // variables object
}

```

Listing 20 Time Series analytics response

* If this value is null or undefined the object will be sent but will not contain the key

** If any of the values is null or undefined, the object will be sent empty, as if there is no filter

3.4.2.1.4. Delete Time Series Analytics details

This request can be used to stop the analysis of the data received for a Time Series Analytics.

REQUEST			GET TIME SERIES ANALYTICS DETAILS		
Method		DELETE			
Url		https://api.datagora.eu/api			
Endpoint		/analytics/TimeSeries/{{analyticsId}}			
Headers		X-Auth-Token	{{access_token}}		

Table 34. Delete Time Series analytics

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

3.4.2.2. Trajectory Analysis

Toolbox for the analysis of trajectories of cars, both individually and as a group.

Several options are provided, from simple statistics on the trips performed by cars; up to the detection of relationships (like correlations or causalities) between the corresponding movements.

Eligible measurement channel and data requests:

Only location channel (measurement-channel-id = 2) is eligible. Therefore, only data requests including this channel (and those including all channels) are eligible.

3.4.2.2.1. Create new Trajectory Analytics

REQUEST	CREATE NEW TRAJECTORY ANALYSIS	
Method	POST	
Url	https://api.datagora.eu/api	
Endpoint	/analytics/TrajectoryAnalysis	
Headers	X-Auth-Token	{{access_token}}
	Content-type	application/json
Body	<pre>{ "description": "", // string "model": "", // string "offer": "", // id "submissionDate": {}, // Object "travelDates": {}, // Object "travelledDuration": {}, // Object "geoBoundingBoxConstraint": {}, // Object "variables": {}, // Object }</pre>	

Table 35. Create new Trajectory analysis

Body explanation:

- **description**: descriptive name of the analytics.
- **model** **: type of trajectory analysis.
- **offer**: id of the data request from which data will be analyzed
- **submissionDate** *: Object representing the dates in which the data has been stored in the marketplace, including max and min values between which data will be searched.
- **travelDates**: Object representing the dates in which the data has been generated by the signal, including max and min values between which data will be searched.
- **travelledDuration** *: Object representing the duration in seconds of each data package, including max and min values between which data will be searched.
- **geoBoundingBoxConstraint**: Object representing the geographical rectangular area within which the data is desired, including longitude and latitude max and min values.
- **variables** **: Object containing the different variables needed for each type of trajectory model. All variables of each model are mandatory.

* if sent empty that filter will be count as NO FILTER (retrieve all)

** refer to subsections for special treatment depending on the Trajectory model

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

Request ok response body:

See [3.4.2.2.3 Get Trajectory Analysis details below](#)

3.4.2.2.1. Statistics

The statistics functionality provides a series of summarizing properties for a given trajectory, namely the total distance traveled in meters, its duration in seconds and its average velocity in meters per second.

Variables

None

3.4.2.2.1.2. Interpolation

The interpolation functionality provides a mean of resampling a given trajectory. The type of interpolation performed is linear.

By providing a desired time resolution (in seconds) the user is able to set the frequency of appearance of each point in the new resampled trajectory.

Variables

- **time-resolution** (numeric-integer): to which the trajectory is intended to be resampled as a numeric value. This value should fulfill the following conditions:
 - o Be an integer.
 - o Be greater than 0.
 - o Be smaller than the time difference between the initial and last coordinates' timestamps.

3.4.2.2.1.3. Clustering

The clustering functionality enables the grouping of similar trajectories. Neighbouring trajectories are labeled as belonging to the same cluster based on a metric of pair-wise distance between coordinates.

Since the adjustment of the parameters intrinsic to the algorithm are adjusted automatically, the user only needs to provide the trajectories to be grouped, with no additional adjustable fields.

Variables

None

3.4.2.2.1.4. Anomaly Detection (anomaly_detection)

The anomaly detection functionality allows to identify abnormal points in a given trajectory. After associating the input trajectory to a cluster from the system's stored information, it compares the pair-wise distance between each of its coordinates and the group-centroid's. If this distance surpasses a certain threshold, the associated coordinates are marked as outlying.

The user has the option to change the type of threshold to apply in this process between a standard deviation protocol and an interquartile range one.

Variables

- **threshold-type** (string): threshold method to apply for the identification of outliers. If the user does not provide this parameter, then the standard deviation protocol is applied by default:
 - o 'std' for standard deviation.
 - o 'iqr' for interquartile range.

3.4.2.2.2. Get Trajectory Analysis list

This request can be used to retrieve a list of your requested Trajectory Analysis.

REQUEST GET TRAJECTORY ANALYSIS LIST		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/ServiceProvider/{{serviceProviderId}}/analytics/trajectoryAnalysis	
Headers	X-Auth-Token	{{access_token}}

Table 36. Get list of requested Time Series analytics

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

Request ok response body:

Response is an array of Trajectory analytics. See [3.4.2.2.3 Get Trajectory Analysis details](#) below

3.4.2.2.3. Get Trajectory Analysis details

This request can be used to get the details of a Trajectory Analysis.

REQUEST GET TRAJECTORY ANALYSIS DETAILS		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/analytics/trajectoryAnalysis/{{analyticsId}}	
Headers	X-Auth-Token	{{access_token}}

Table 37. Get list of requested Time Series analytics

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

Request ok response body:

```

{
  "id": "---", // id
  "model": "---", // string
  "description": "my analytics", // string
  "serviceProvider": { }, // service provider object
  "offer": { }, // data request object
  "status": "ok" / "pending" / "error", // string
  "createdAt": "2019-05-20T09:20:54.222Z", // date-time
  "updatedAt": "2019-05-20T09:20:54.499Z", // date-time
  "travelledDuration": {
    "min": -, // date *
    "max": - // date *
  },
  "submissionDate": {
    "min": "2019-06-03", // date *
    "max": "2019-06-27" // date *
  },
  "travelDates": {
    "min": -, // date *
    "max": - // date *
  },
  "geoBoundingBoxConstraint": {
    "latitude-max": 58, // number **
    "latitude-min": 38, // number **
    "longitude-max": 25, // number **
    "longitude-min": -5 // number **
  },
  "variables": { }, // variables object
}

```

Listing 21. Trajectory Analysis response

* If this value is null or undefined the object will be sent but will not contain the key

** If any of the values is null or undefined, the object will be sent empty, as if there is no filter

3.4.2.2.4. Get Trajectory Analysis results

This request can be used to get the results of a Trajectory Analysis.

REQUEST GET TRAJECTORY ANALYSIS DETAILS

Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/analytics/trajectoryAnalysis/{{analyticsId}}/results	
Headers	X-Auth-Token	{{access_token}}

Table 38. Get results of requested Time Series analytics

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

Request ok response body:

Response is an array of objects that differ depending on the model of the Trajectory Analysis

3.4.2.2.4.1. Statistics

```
[
  {
    "id": "----", // id
    "datapackage-id": "----", // id
    "analysis": "----", // id
    "distance": 10330.636916737218, // number
    "duration": 1017, // number
    "velocity": 10.157951737204737, // number
  },
  ...
]
```

Listing 22. Trajectory statistics response

3.4.2.2.4.2. Interpolation

```
[
  {
    "id": "----", // id
    "datapackage-id": "----", // id
    "analysis": "----", // id
    "new_data": [
```



```

    {
      "id": "----", // id
      "interpolation": "----", // id
      "value": [
        51.499995, // number
        7.5129, // number
      ],
      "timestamp": "----", // date-time
      "createdAt": "----", // date-time
      "updatedAt": "----", // date-time
    },
  ],
  ...
]

```

Listing 23 Trajectory interpolation response

3.4.2.2.4.3. Clustering

```

[
  {
    "_id": "----", // string
    "total": "----", // integer
    "packages": [
      "----", // datapackage-id
    ]
  },
  ...
]

```

Listing 24. Trajectory clustering results

3.4.2.2.4.4. Anomaly Detection

```

[
  {
  },
  ...
]

```

Listing 25. Trajectory anomaly detection results

3.4.2.3. Networks

Toolbox devoted to the analysis of relation between data collected from different sources for the same measurement channel of a Data Request.

3.4.2.3.1. Create new Network

REQUEST	CREATE NEW NETWORK
Method	POST

Url	https://api.datagora.eu/api	
Endpoint	/analytics/network	
Headers	X-Auth-Token	{{access_token}}
	Content-type	application/json
Body	<pre>{ "offer": "", // id "measurement-channel-id": "", // string }</pre>	

Table 39. Create new Network

Body explanation:

- **offer**: id of the data request from which data will be analyzed
- **measurement-channel-id**: array of measurement channels from which data received will be analyzed

Take into account that before creating the network a check is conducted to know the availability of nodes inside it. Only networks with more that one node available will be generated.

Also a network for the combination of Data Request and measurement channel is unique.

3.4.2.3.2. Get Network list

This request can be used to retrieve a list of your generated networks.

REQUEST	GET NETWORK LIST	
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/ServiceProvider/{{serviceProviderId}}/analytics/network	
Headers	X-Auth-Token	{{access_token}}

Table 40. Get list of generated networks

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

Request ok response body:

Response is an array of networks. See [3.4.2.3.3 Get Network](#) below

3.4.2.3.3. Get Network

This request can be used to get a network details.

REQUEST GET NETWORK DETAILS		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/analytics/network/{{analyticsId}}	
Headers	X-Auth-Token	{{access_token}}

Table 41. Get Network details

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

Request ok response body:

```
{
  "id": "----",                                // id
  "offer": { },                                // data request object
  "measurement-channel": { },                  // measurement channel object
  "measurement-channel-id": "1",                // string
  "inputAeonChannelId": "----",                  // id
  "inputAeonSubscriptionUrl": "https://{aeon_url}/subscribe/{uuid}", // string
  "createdAt": "2019-05-20T09:20:54.499Z",      // date-time
  "updatedAt": "2019-05-20T09:20:54.499Z",      // date-time
}
```

Listing 26. Network response

3.4.2.3.4. Get Network status

This request can be used to get the current status and information of an active network.

REQUEST GET NETWORK STATUS		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/analytics/network/{{analyticsId}}/info	
Headers	X-Auth-Token	{{access_token}}

Table 42. Get Network status

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

Request ok response body:

```
{
  "result": "success",                                // string
  "response": "---",                                  // id
  "network-id": "---",                                // id
  "network-efficiency": 0.02...,                       // number
  "link-density": 0.25,                                // number
  "adjacency-matrix": [                                // object array
    {
      "source": 0,                                     // number
      "target": 1,                                     // number
      "value": 34.816...,                               // number
    },
    ...
  ],
  "node-eccentricity": [                                // object array
    {
      "node-id": 0,                                     // number
      "value": 34.816...,                               // number
    }
  ]
}
```

```

    },
    ...
  ]
}
}

```

Listing 27. Network status response

3.4.2.4. Machine Learning

Toolbox devoted to the analysis of categorization of data rows obtained through Data Views to predict the values of the incomplete rows.

3.4.2.4.1. Create ML model

REQUEST CREATE NEW ML MODEL		
Method	POST	
Url	https://api.datagora.eu/api	
Endpoint	/analytics/MachineLearning	
Headers	X-Auth-Token	{{access_token}}
	Content-type	application/json
Body	<pre> { "serviceProvider": "5cd421a79c0f591e006515a5", "method": "build", "algnam": "sklearn.neural_network.MLPClassifier", "config": {}, "data": { "sei_uri_base": "https://vian-dev.fit.vutbr.cz/cross-cpp/", "view_id": {{dataViewId}}, "limit": -100, "page": 0 } } </pre>	

Table 43. Create new ML model

Body explanation:

- **serviceProvider**: id of the service provider
- **method**: method of creation. Can be "build" or "import"
- **algnam** *: string identifying the algorithm to be used by the ML model.
- **config**: configuration object. Currently managed by the ML component.
- **data**: object identifying the data to be used to train the ML model.
 - o **sei_uri_base**: location of the Data View. Currently not to be changed.
 - o **view_id**: id of the Data View to use

- **limit and page:** selector of the rows from the Data View. Recommended to go be **limit=-100** and **page=0** to use the 100 most recent data rows collected.

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

Request ok response body:

See [3.4.2.4.3 Get ML model details below](#)

3.4.2.4.2. Get ML model list

This request can be used to retrieve a list of your created Machine Learning models.

REQUEST	GET ML MODELS LIST	
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/ServiceProvider/{{serviceProviderId}}/analytics/machineLearning	
Headers	X-Auth-Token	{{access_token}}

Table 44. Get list of created ML models

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

Request ok response body:

Response is an array of Trajectory analytics. See [3.4.2.4.3 Get ML model details below](#)

3.4.2.4.3. Get ML model details

This request can be used to get the details of a Machine Learning model.

REQUEST GET ML MODEL DETAILS		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/analytics/MachineLearning/{{analyticsId}}	
Headers	X-Auth-Token	{{access_token}}

Table 45. Get details of a Machine Learning model

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

Request ok response body:

```
{
  "id": "5f7afd821288ac2100488533"           // string
  "alguuid": "a1869a51-5992-4243-a3ca-1579450e462b", // string
  "serviceProvider": {...},                 // service provider object
  "method": "build",                        // string
  "algnam": "sklearn.neural_network.MLPClassifier", // enum string
  "config": {},                             // object
  "data": {                                 // object
    "sei_uri_base": "https://vian-dev.fit.vutbr.cz/cross-cpp/",
    "view_id": "5ef5ad9dce7ee04300f06b00",
    "limit": -100,
    "page": 0,
  },
  "type": "batch",                          // string
  "status": "check",                        // string
  "createdAt": "2020-10-05T11:03:30.648Z",   // date-time
  "updatedAt": "2020-10-05T11:03:30.648Z",  // date-time
}
```

```
}
```

Listing 28. Machine Learning model details

3.4.2.4.4. Export ML model

This request can be used to export the ML model as it is stored in the ML component.

REQUEST EXPORT ML MODEL		
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/analytics/MachineLearning/{{analyticsId}}/export	
Headers	X-Auth-Token	{{access_token}}

Table 46. Export Machine Learning model

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

Request ok response body:

```
{
  "id": "5f7afd821288ac2100488533"           // string
  "alguuid": "a1869a51-5992-4243-a3ca-1579450e462b", // string
  "serviceProvider": {...},                 // service provider object
  "method": "build",                        // string
  "alname": "sklearn.neural_network.MLPClassifier", // enum string
  "config": {},                             // object
  "data": {                                 // object
    "sei_uri_base": "https://vian-dev.fit.vutbr.cz/cross-cpp/",
    "view_id": "5ef5ad9dce7ee04300f06b00",
    "limit": -100,
    "page": 0,
  },
  "type": "batch",                          // string
  "status": "check",                        // string
  "createdAt": "2020-10-05T11:03:30.648Z",   // date-time
}
```



```

    "updatedAt": "2020-10-05T11:03:30.648Z",           // date-time
    "model": "...",                                   // string with the exported model
}

```

Listing 29. Export Machine Learning model

3.4.2.4.5. Get ML model status

This request can be used to get the current status of the ML model training.

REQUEST		GET ML MODEL STATUS
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/analytics/MachineLearning/{{analyticsId}}/status	
Headers	X-Auth-Token	{{access_token}}

Table 47. Export Machine Learning model

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

Request ok response body:

```

{
  "status": "success"           // string
  "data": {                     // object
    "status": ""                // string
  }
}

```

Listing 30. Get Machine Learning model status

Status can be one of these possible values:

- **failed**: the training has failed. Model should be deleted
- **initialized**: the model has been instantiated but is not yet running.
- **running**: the model is currently under training
- **finished**: the model training has finished successfully and can be used

- **imported**: the model has been imported successfully and can be used

3.4.2.4.6. Evaluate ML model

This request can be used to get an evaluation of a ML model training.

REQUEST		GET ML MODEL STATUS
Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/analytics/MachineLearning/{{analyticsId}}/evaluate	
Headers	X-Auth-Token	{{access_token}}

Table 48. Export Machine Learning model

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

Request ok response body:

```
{
  "status": "success"           // string
  "data": {                     // object
    "score": 1                  // double
  }
}
```

Listing 31. Evaluate Machine Learning model

3.4.2.4.7. Apply ML model

This request can be used to apply the current training of the ML model to the whole Data View.

It will assign a category value on every row based on the training with the manually assigned values.

REQUEST	APPLY ML MODEL
---------	----------------

Method	GET	
Url	https://api.datagora.eu/api	
Endpoint	/analytics/MachineLearning/{{analyticsId}}/apply	
Headers	X-Auth-Token	{{access_token}}

Table 49. Export Machine Learning model

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

Request ok response body:

```
{
  "status": "success"                                // string
  "data": [                                           // rows array
    {
      "#": 1,                                          // integer (row number)
      "608": 11,                                     // channel value type
      "900": null,
      "charger_availability": "low"                   // enum string (category value)
      "tstamp": "2020-06-30T02:30:00+02:00",         // string
    },
    { ... }
  ],
}
```

Listing 32. Apply Machine Learning model

3.4.2.4.8. Kill ML model

This request can be used to stop a ML model.

REQUEST	KILL ML MODEL
Method	GET
Url	https://api.datagora.eu/api

Endpoint	/analytics/MachineLearning/{{analyticsId}}/kill	
Headers	X-Auth-Token	{{access_token}}

Table 50. Kill Machine Learning model

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

3.4.2.4.9. Delete ML model

This request can be used to stop and delete a ML model.

REQUEST DELETE ML MODEL		
Method	DELETE	
Url	https://api.datagora.eu/api	
Endpoint	/analytics/MachineLearning/{{analyticsId}}	
Headers	X-Auth-Token	{{access_token}}

Table 51. Delete Machine Learning model

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

4. AEON

4.1. What is AEON

AEON is a cloud platform to create applications with real time communications channels. The architecture is based on the strongly communication needs that we need to face nowadays, with billions of interconnected devices and short times of response. Thus, the technological solutions used for the implementation are based on strong requirements about performance, response and scalability, making use of the most advanced cutting-edge technologies.

AEON platform offers a shared cloud-based message queuing framework enabling messaging between various entities that wish to communicate with each other seamlessly and reliably using standard vendor neutral protocols

Benefits:

- Communicate applications and services through a real time network
- Easy to use, easy to integrate in developments: AEON provides an SDK to connect your services and devices over a globally scaled real-time network
- Performance, Scalability and Reliability: High performance for message delivery and data exchange between business processes and devices and from device to device. AEON is able to handle multiple types and priorities of messages, whilst at the same time providing the necessary Quality of Service. AEON provides reliable messaging with durability and persistence and needs to scale well for extremely large volumes.
- Big Data: AEON can take care of the cloud messaging of the data capture from M2M environments and big data flows.

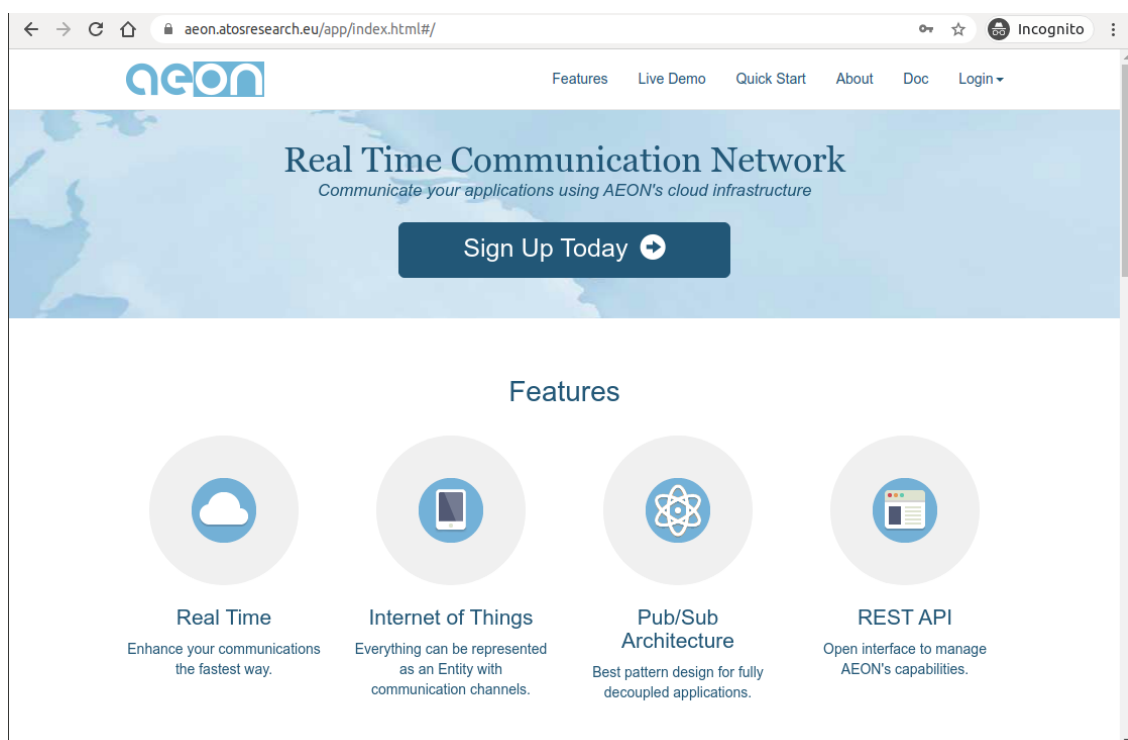


Figure 31. AEON dashboard.

4.2. Configuring AEON channels

AEON users can create AEON channels to publish/subscribe messages through their subscription and publications URLs. The Cross-CPP marketplace uses AEON with its own user to create communication channels for Data Requests (offers), active streaming analytics, Data Views, etc. to get data notifications in real-time.

Data consumers or Service providers will subscribe to the subscriptions URLs to receive real-time data from the Data Requests, analytics, Data Views etc. The configuration is provided in the detailed information view of each one in the Cross-CPP Marketplace frontend and API.

4.3. Subscribing to AEON

AEON provides an SDK (Node.js, JavaScript and Java) that encapsulates the complexity of connecting to a socket server.

You can find an example in section [1.4.2 Push Approach Data Retrieval](#) and visit <https://aeon.atosresearch.eu:3000/public/doc/html/quickstart/nodejs.html> for further documentation.

5. Context Monitoring and Extraction (CME) guide

The CME module provides one customisation endpoint for Service Providers:

- Customisation of existing or creation of new Reasoning Rules

The customisation explained here and any further customisation of the CME components, e.g. for adding reasoning rules for extraction, can be made by downloading and changing the CME module as provided in the open source code project on GitHub⁴ under the EPL 2.0 license or by placing a customisation request to the CME team at context-support@cross-cpp.eu.

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.

5.1. 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 CME administrator:

- setup of the reasoning rules for the context-sensitive data discovery
- 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 32 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.

⁴ Link to be added

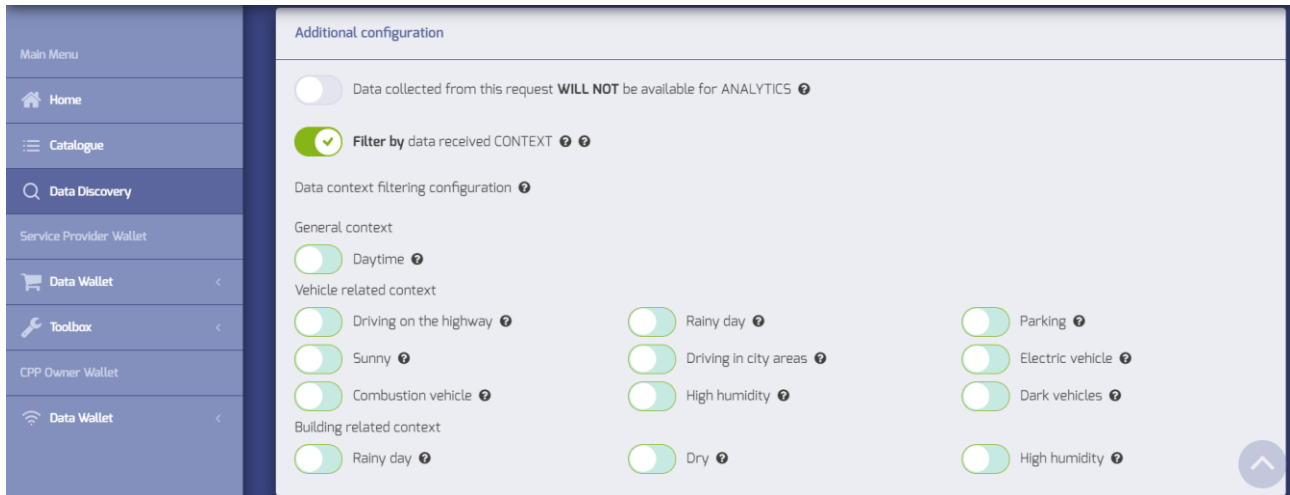


Figure 32. Context-sensitive data filtering in the Cross-CPP Marketplace UI

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 33 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 ?	<input checked="" type="checkbox"/> Weekday	<input type="checkbox"/> Weekend
Activity ?	<input type="checkbox"/> Business	<input checked="" type="checkbox"/> Leisure

Figure 33. 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>
</ruleConfiguration>
```

```
</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. 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 classes⁵:

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

⁵ Link to github repository to be added here

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 be 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
```

Further customisation is possible by developers as the CME module is provided as an open source code project on github⁶ under the EPL 2.0 license. Furthermore, customisation can also be requested via context-support@cross-cpp.eu.

⁶ Link to be added

FAQ

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, Time Series) and retrieve those datasets in a seamless experience thanks to the open SDK-API created.

Q: How do I, as data consumer, register into CROSS-CPP data-marketplace?

A: You can find the registration form by clicking the "Sign on!" button in the landing page. Select "Service Provider" 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. Then, access the link in your email, login and accept the consent to start using the Cross-CPP Marketplace.

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. You must also be familiar with AEON, as it is the communication channel used to send the data. You can find information for both in this guide.

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 measurement channel?

A: A measurement 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 Data Discovery component and how does it work?

A: The Data Discovery component is a tool that allows you to find what data you can access through the marketplace. There you can use the filters provided to narrow or enlarge your results and create Data Request based on the configured search. You can think of it like a test of what would you receive if you publish that request.

Q: What is the Context Monitoring and Extraction module and how can help my Organization?

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

Data Requests

Q: What is a Data Request?

A: A Data Request is a set of filters that defines which type of data would you like to receive. You would receive data from data owners that have accepted these requests through each request unique AEON channel.

Q: How do I create a Data Request?

A: You have to get to the Data Discovery and define the data you are interested in through the filters given. Once set you give it a descriptive name, so data owners guess in a glance the nature of the request.

Q: How do I receive data from a Data Request?

A: In the very moment a Data Request is created an AEON channel is assigned. You can find the channel configuration in each data request details view. Data from data owners that have accepted the request will be sent through each AEON channel assigned to each eligible data

request, meaning you can receive data from the same user and signal from more than one request.

Q: Can I modify my Data Request one created?

A: No. The acceptance of a request by a data owner implies a consent from its side. Modifying the request would make invalid such consent. Therefore, you can create another Data Request with the new desired configuration.

Q: Can I use the data collected for other purposes not described in my Data Request?

A: No. The acceptance of a request by a data owner implies a consent from its side. That user allows certain usage of the data given and only for the purposes described in the request.

Toolbox

Q: What is the toolbox?

A: The Toolbox is a set of tools that offers Service Providers a way to generate analytics from the data obtained from Data Requests or further filter this data in order to get exact measurements or use the service as a notification system.

Q: How can I request analytics?

A: Any analytics uses data packages received from a Data Request, meaning that first the data Request must have been accepted by CPP Owners and started receiving data. Then, on the Cross-CPP Marketplace a Service Provider can create analytics based on those Data Requests.

Q: How do I get my analytics results?

A: Depending on the analytics type results can be instantly shown on the screen, as a diagram, chart, map and so on, or a new AEON channel is provided in order to subscribe to upcoming analytics results. Every analytics type explains the way to get the results on screen.

Q: How can I consult my analytics?

A: Any created analytics can be consulted on the Cross-CPP Marketplace under Analytics section. There a Service Provider can get the AEON channel for subscription, see results of one-time analytics, or even delete them.

Q: Which Time Series forecasting method should I use for my time series predictions?

A: Prior to attempting to predict future values of a time series, you should verify the assumptions your selected algorithm does on the input data. For example, ARIMA will assume that your data is autoregressive and Regression trees will not be able to detect trends on data (thus not being advisable for a time series in which trend is a significant feature). If you are unsure about the underlying patterns of your data, Neural Networks may work best for you, as it does not rely on any a-priori assumption.

Q: How does the embedding dimension parameter “m” affect the entropy metrics?

A: The time-series are defined by sequences of points. The embedding dimension refers to the number of points use to evaluate. So then, smaller embedding dimensions would yield more

detailed information while larger numbers tend to be more general. Bear in mind if the embedding dimension is too low maybe the results may be chaotic.

Q: How do I choose between a Pearson and a Spearman's correlation coefficient?

A: This decision will depend on the nature of your data, as these coefficients measure different types of association between variables (Pearson quantifies linear relationship, while Spearman does so with non-linear patterns). Their use is not exclusive, so even both could possibly be valuable to your analysis.

Q: What are the units of the trajectories module?

A: For distances, the meter, and for times the second. Then the velocity is expressed in m/s.

Q: What is the interpretation of the results of clustering the trajectories?

A: Each trajectory included in the data is assigned to a numerical ID that refers to the cluster that is being assigned. By instance, all trajectories of similar length from east to west are grouped in the cluster number 1, so the results are 1 for each trajectory.

Q: What do the nodes represent in the networks module? And the links?

A: Each node is an object. It could represent a house, building, a car. The links are an abstract representation of some magnitude that is being measured. Links could be based on distances by instance.

Q: What is a Data View?

A: A Data View is a configuration to get data filtered by specific values constraints for one or more Measurement Channels included in a Data Request.

Q: How can I create a Data View?

A: Go to Data Views section under Toolbox in the Cross-CPP Marketplace. There a step by step guide will be offered.

Q: How do I get the data view results?

A: Service Providers can consult their generated Data Views through Data Views under Toolbox section. There the configuration can be consulted, as well as the AEON channel to subscribe to. Also, the options of retrieving the latest data or even deleting the Data View are offered there.

Q: How can I use a created ML model?

A: A machine learning model that was built in a previous step can be applied on new (unseen, unannotated) Data View rows by invoking the Apply function. The service estimates the category on the given data rows and outputs it as the response.

Q: My service processes a lot of data, is the ML component ready for big data application?

A: Yes, even a large neural network model loads in less than 480 ms and it can apply the category prediction with the speed of more than 5000 rows per second on the Cross-CPP testing infrastructure.

Q: What ML methods should I try first?

A: Although the ML components support all new and fancy neural network methods available in Scikit-learn, Google TensorFlow, and Facebook PyTorch libraries, we suggest to start with the simple linear method of Stochastic Gradient Descend or the basic Multi-Layer Perceptron neural network for your initial experiments. Often, the quality of results provided by these models is satisfactory and performance gains can be brought by additional annotations of the data.

AEON

Q: What is AEON?

A: AEON is a cloud platform to create applications with real time communications channels

Q: How can I use AEON to subscribe to my data requests and data analytics?

A: AEON provides an SDK (Node.js, JavaScript and Java) that encapsulates the complexity of connecting to a socket server. Please refer to section 6.4, document examples and online page for extended documentation.

Q: How do I create or configure an AEON channel?

A: You don't have to create or configure any channel. All needed AEON channels, such as for data requests or analytics, are created and assigned by Cross-CPP. You only have to use the channels given. The channels configuration can be found in the details view of each data request or analytics.

Glossary

Administrator: Cross-CPP marketplace system administrator

Autoregressive data: In a time series domain, it refers to data which values depend on prior data points from the same time series.

AEON: AEON application

AEON application: publication/subscription based communication application

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

Analytics Toolbox: set of available analytics functions to be requested by the Service Provider

CIDM: Common Industrial Data Model

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

CME: Context Monitoring and Extraction

Company Backend: system of an OEM that provides its users data to the Cross-CPP marketplace

Contract: entity that resumes the acceptance of a data request from a CPP owner

CPP: cyber-physical product

CPP Data: data created by a CCP and sent to the system by the Company Backend

CPP owner: CPP owner which CPP is registered in the Cross-CPP data-marketplace

Cross-CPP: System

CSS: Context Sensitive Security

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

Data View: set of configured filters to receive specific values from a Data Request through a different notification channel

Entropy: It is usually explained as the order of a system. It is more accurate to understand the entropy as the lost information of a system. This definition for data classification problems implies the algorithms search for the variables that reduces the lost information of the system, those are the best classifiers.

Homoscedasticity: property of a multivariate domain in which the variance of each variable's error term is equal.

Machine Learning Analytics: A type of Analytics included in the toolbox

Marketplace: Marketplace Web Application

Measurement Channel: sampler of the data the signals process

Monotonic relationship: A type of association between variables that occurs when two variables tend to increase or decrease in the same direction, but not following a linear pattern (linear relationship).

MP: Marketplace

Multilayer Perceptron: A type of Artificial Neural Network with a varying number of hidden (processing) layers.

Networks Analytics: A type of Analytics included in the toolbox

Network Diameter: Value indicating the shortest distance between the two most distant nodes in a network.

Network Efficiency: Measure of how well information is exchanged between the nodes of a networks.

Node Eccentricity: Value representing the centrality of a network's node, or how close it is to the other nodes in the network.

OEM: Original Equipment Manufacturer

Rank Correlation: A type of correlation measure that quantifies ordinal association between two variables.

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

Service Provider Wallet: group of MP functionalities for Service Providers

Signal: information provider of the data the CPP sensors generate

Stationarity: Property of a time series indicating that its statistical properties (e.g. mean, variance...) do not change over time.

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

Time Series Analytics: A type of Analytics included in the toolbox that analyses drifts in the data flow of time-series type channels.

Time Series Complexity: Measure of the presence of nonlinear patterns that explain the behaviour of a time series' data.

Trajectories Analysis: A type of Analytics included in the toolbox that uses trajectory related signals.

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

Figures

Figure 1. Cross-CPP Marketplace general workflow	8
Figure 2. Cross-CPP Marketplace sign up form	9
Figure 3. Postman example - Marketplace user authentication request.....	9
Figure 4. Postman example - Marketplace user authentication response	10
Figure 5. Cross-CPP Data Discovery view	10
Figure 6. Data Discovery step 1.....	11
Figure 7. Data Discovery step 2.....	12
Figure 8. Data Discovery step 3	12
Figure 9. Discovery results - general statistics	13
Figure 10. Discovery results - heatmap and charts	14
Figure 11. Create data Request from Data Discovery	15
Figure 12. Data Request details.....	16
Figure 13. Data Request data packages received list	17
Figure 14. Data Request metadata pull example.....	18
Figure 15. Data Request data pull example	19
Figure 16. Data package details pull request.....	20
Figure 17. Data Request AEON subscription url.....	21
Figure 18. Analytics Toolbox workflow.....	24
Figure 19: Create Time Series analytics request – select analysis type.....	25
Figure 20: Create Time Series analytics request – select Data Request.....	26
Figure 21: Create Time Series analytics request – requested analytics information	26
Figure 22: Request trajectory analytics – select Data Request	27
Figure 23: Request trajectory analytics – configure filters	27
Figure 24: Request trajectory analytics – select analysis type.....	28
Figure 25: Trajectory analysis results	28
Figure 26: Trajectory analysis API – get analytics information.....	29
Figure 277: Request Networks module – Data Request Generation.....	30
Figure 288: Request Networks module – select signal.....	30
Figure 299: Request trajectory analytics – select analysis type	31
Figure 30. Cross-CPP OpenAPI Specification	41
Figure 31. AEON dashboard.	101
Figure 32. Context-sensitive data filtering in the Cross-CPP Marketplace UI.....	103
Figure 33. context-sensitive data access control options in the Cross-CPP Marketplace UI	104

Tables

Table 1. Request headers.....	42
Table 2. Get Access Token.....	42
Table 3. Get user profile request.....	43
Table 4. Get Catalogue request.....	44

Table 5. Get single Signal request	45
Table 6. Get all signals request	47
Table 7. Get single measurement channel request	47
Table 8. Get all channels request	48
Table 9. Data Discovery request	49
Table 10. Get channel suggestions	53
Table 11. Create Data Request request	55
Table 12. Delete Data Request request	56
Table 13. Get Data Requests request	57
Table 14. Get Data Request details request	58
Table 15. Get Data Request contracts request	59
Table 16. Get Data Transactions resume request	61
Table 17. Get Data Request transactions request	62
Table 18. Get Data Package details request	63
Table 19. Get Data Packages received for a Data Request request	64
Table 20. Query Parameters	65
Table 21. Get Metadata packages received for a Data Request request	65
Table 22. Create new Data View	67
Table 23. Get list of active Data Views	68
Table 24. Get Data View details	68
Table 25. Retrieve data view collected data	69
Table 26. Create category request	71
Table 27. Get list of created categories	72
Table 28. Get Category details	72
Table 29. Assign category to an uncategorized Data View	73
Table 30. Assign category values to Data View rows	74
Table 31. Create new Time Series analytics	75
Table 32. Get list of requested Time Series analytics	79
Table 33. Get list of requested Time Series analytics	80
Table 34. Delete Time Series analytics	81
Table 35. Create new Trajectory analysis	82
Table 36. Get list of requested Time Series analytics	85
Table 37. Get list of requested Time Series analytics	85
Table 38. Get results of requested Time Series analytics	87
Table 39. Create new Network	89
Table 40. Get list of generated networks	89
Table 41. Get Network details	90
Table 42. Get Network status	91
Table 43. Create new ML model	92
Table 44. Get list of created ML models	93
Table 45. Get details of a Machine Learning model	94
Table 46. Export Machine Learning model	95
Table 47. Export Machine Learning model	96

Table 48. Export Machine Learning model.....	97
Table 49. Export Machine Learning model.....	98
Table 50. Kill Machine Learning model.....	99
Table 51. Delete Machine Learning model.....	99

Lists

Listing 1. Get Access Token response ok	42
Listing 2. Get user profile response ok.....	44
Listing 3. Get Catalogue response ok.....	45
Listing 4. Signal response.....	46
Listing 5. Channel response.....	48
Listing 6. Discovery request example	51
Listing 7. Discovery response example	53
Listing 8. Channel suggestions response	54
Listing 9. Create data request response.....	56
Listing 10. Get my Data Requests.....	58
Listing 11. Get Data Request details.....	59
Listing 12. Get Data Request contracts	60
Listing 13. Get transactions resume.....	61
Listing 14. Get Data Request transactions	62
Listing 15. Get data package details.....	64
Listing 16. Get Data Request Metadata	66
Listing 17. Data View response	69
Listing 18. Data View retrieve response.....	70
Listing 19. Get categories list response	73
Listing 20 Time Series analytics response	81
Listing 21. Trajectory Analysis response	86
Listing 22. Trajectory statistics response	87
Listing 23 Trajectory interpolation response.....	88
Listing 24. Trajectory clustering results.....	88
Listing 25. Trajectory anomaly detection results	88
Listing 26. Network response	91
Listing 27. Network status response.....	92
Listing 28. Machine Learning model details	95
Listing 29. Export Machine Learning model.....	96
Listing 30. Get Machine Learning model status.....	96
Listing 31. Evaluate Machine Learning model.....	97
Listing 32. Apply Machine Learning model.....	98

+ Build innovative services upon cross-sectorial data streams

The future is connected.

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

Every effort has been made to ensure that all statements and information contained herein are accurate, however the Cross-CPP Project Partners accept no liability for any error or omission in the same.

© 2020 Copyright in this document remains vested in the Cross-CPP Project Partners.