Grant Agreement Number: 723265

Project acronym: **Clusters 2.0**

Project full title: Clusters 2.0 - Open network of hyper connected logistics clusters towards Physical Internet

# D.3.2

# CNI API Description

**Due delivery date: 07/02/2018**
**Actual delivery date: 07/02/2018**
Organization name of lead participant for this deliverable: Nallian

| Project co-funded by the European Commission within Horizon 2020 | | |
|---|---|---|
| Dissemination level | | |
| PU | Public | x |
| PP | Restricted to other programme participants | |
| RE | Restricted to a group specified by the consortium | |
| CO | Confidential, only for members of the consortium | |

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

Document Control Sheet

| | |
|---|---|
| Deliverable number: | D3.2 |
| Deliverable responsible: | Nallian |
| Work package: | WP3 |
| Editor: | Ivo Fremau |

| Author(s) – in alphabetical order | | |
|---|---|---|
| Name | Organisation | E-mail |
| Eric Cauchi | Seability | SEA@SEAbility.eu |
| Frans Cruijssen | Argusi | f.cruijssen@argusi.org |
| Ivo Fremau | Nallian | Ivo.fremau@nallian.com |
| Marcel Huschebeck | PTV | Marcel.Huschebeck@ptvgroup.com |
| Paul Delbar | Nallian | Paul.delbar@nallian.com |
| Wout Vanduffel | City Depot | wout.vanduffel@citydepot.be |

| Document Revision History | | | |
|---|---|---|---|
| Version | Date | Modifications Introduced | |
| | | Modification Reason | Modified by |
| v0.1 | 02/09/2017 | Initial start | Nallian |
| v0.1.1 | 01/10/2017 | Peer review | Seability |
| v0.1.2 | 15/10/2017 | Context description | Nallian |
| v0.2 | 16/10/2017 | Figure and table update | Nallian |
| v0.3 | 21/10/2017 | Chapters 3, 4 and 5 | Nallian |
| v0.3.1 | 30/10/2017 | Content extension request | PTV |
| v0.3.2 | 16/11/2017 | Extra content + figures | Nallian |
| v0.3.3 | 03/12/2017 | Peer review | Seability |
| v0.4 | 15/12/2017 | Chapter 2: extra scoping content | Nallian |
| v0.5 | 04/01/2018 | Overall updated version | Nallian |
| v0.5.1 | 16/01/2018 | Extra input on chapters 2, 3, 4 and 5 | PTV |
| v0.5.2 | 18/01/2018 | Extra input on chapters 2, 4 and 5 | Citydepot |
| v0.6 | 23/01/2018 | Final version for peer review | Nallian |
| v1.0 | 30/01/2018 | Final version | Nallian |

Abstract

WP3 aims to establish logistics clusters integration into a high performing synchromodal transportation network on a EU scale. WP3 addresses the shift towards low emission transport modes and consolidated freight management between logistics clusters following a demand driven approach. One of the ways to achieve this is to develop added value services of enhanced collaboration across logistics clusters. These value added services will create insights and applications that will lead to efficient cargo pooling. This deliverable focuses on the way the value added service providers will interface with the platform through API calls.

## Legal Disclaimer

The information in this document is provided “as is”, and no guarantee or warranty is given that the information is fit for any particular purpose. The above referenced consortium members shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law. © 2017 by Clusters 2.0 Consortium.

## Abbreviations and Acronyms

| Acronym | Definition |
|---|---|
| API | Application Program Interface |
| APP | Application |
| CNI | Clusters Network Integration |
| JSON | JavaScript Object Notation |
| LSP | Logistic Service Provider |
| OAuth2 | Authorization framework that enables applications to obtain limited access to user accounts |
| VAS | Value Added Services |
| WP | Work Package |

# Table of Contents

## List of Figures

# Executive Summary

The Clusters 2.0 CNI platform creation and management task will develop the CargoStream platform in order to enable the optimized network design based on

i) the Clusters aggregated transport demand data and
ii) the available services,

to provide bundling and new services recommendations.

The CargoStream platform will provide an extensive database of historical cargo movements for various shippers. In order to find collaboration opportunities, it is necessary to retrieve some of this information in an aggregated fashion.

After reviewing the functional requirements with all actors, we define in this document the API needed to connect Value Added Service (VAS) providers in order that they are able to get the necessary platform data and to deliver insights back to the platform. Targeted insights are

i) optimized network design,
ii) cargo pooling,
iii) imbalanced lane optimization between different European logistic corridors and
iv) backhaul opportunity detection.

The VAS providers will deliver their insights via Apps on the platform. The different VAS possibilities are detailed in deliverables D3.7 and D3.8.  This deliverable describes the API as needed by the VAS providers.

# 1. Introduction

## 1.1 Purpose of Document

This document describes the high-level requirements of the API that will allow user interfaces and VAS providers to retrieve this data.

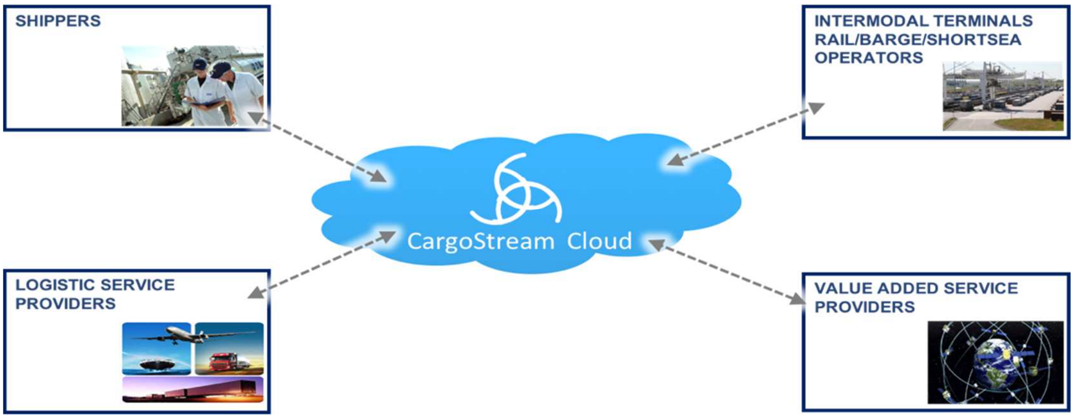
The API is meant for VAS providers, but also for other members such as Logistic Service Providers (LSP), terminals and hubs. Acting in the role of VAS they can get API access.

Before giving details on the technical scope of the API it is crucial to explain the overall benefit of CargoStream: Why do we need it, what is it and how does it work?

## 1.2 Why is there a need for CargoStream?

It is our common responsibility to drive the sustainability of our supply chains, while we also have to improve service, inventory and cost levels. On top, congestion and truck driver shortages are very important upcoming challenges that will impact the reliability of the transportation system. Horizontal supply chain collaboration between all stakeholders provides an answer to the challenges above, as it enables either an optimization of road transportation through round trips and empty mile reduction or a modal shift, moving transportation from road to rail or inland waterways. Currently there is not enough scale however to industrialize horizontal supply chain collaboration. A challenge for which CargoStream provides the solution.

**Figure 1 – CargoStream Cloud: type of actors**



## 1.3 What is CargoStream?

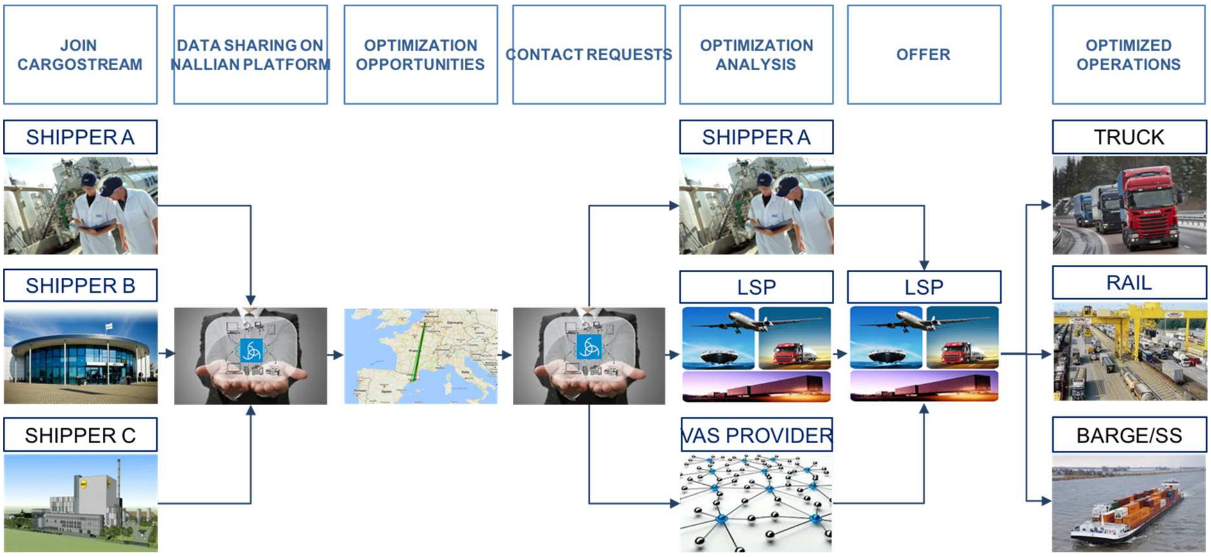
CargoStream is an independent Pan-European platform that creates scale among shippers to drive horizontal supply chain collaboration through bundling their transportation needs with other shippers. With CargoStream we support the vision of the EU Commission to drive more sustainability in transportation.

CargoStream is an interconnected, neutral and open network on which shippers, intermodal terminals, rail & barge operators, logistic service providers, trustees and optimizers collaborate by synchronizing supply chain requirements with the right mix of transport modes (see figure 1)

## 1.4 How does CargoStream work?

Shippers communicate their regular transportation needs to the CargoStream platform. The CargoStream platform anonymizes and aggregates the needs of multiple shippers, and makes these data available to VAS, LSP's and terminals or hubs, who analyse, optimise, and generate collaborative proposals that benefit the community. An example of a typical business flow from the point of view of a shipper is given in Figure 2.

**Figure 2 – CargoStream: typical business flow for a shipper**



The CargoStream platform contains a growing set of Apps which offer additional functionalities for the community. Think about Apps showing the potential efficiencies on your own lanes and on collaborative round trips, or Apps showing the availability of free slots on intermodal lanes, or even the creation of new lanes.

# 2. Scope

## 2.1 General scope

In the context of the Clusters 2.0 project, CargoStream will enable independent third-party application providers (called VAS providers) to connect their insights App-based to the data platform.

There are 2 main reasons why there is a need to do so:

i) VAS Providers often have the knowledge and the application to generate intelligent transport optimizations but lack the ability to have access to European shippers' demand data. Often their data scope is limited to their own clientele. Via CargoStream they can access a much broader data set of European companies.

ii) Shippers seek for the best insights, generating optimizations for their transport requirements. However, for every new application from a third-party company, they have to reconnect their data with that VAS provider, which is each time a cumbersome process. Often lack of available ICT time is a show stopper at shippers' side to continue the effort to work with the new application.

As CargoStream acts as a neutral platform, not owned by a shipper nor a logistic transport provider, both shippers and VAS providers feel confident to join the platform. Shippers benefit from a many-2-many (M2M) connection via one single data integration effort and avoid a lock-in situation. VAS providers can leverage their knowledge without investment in time consuming one-on-one shipper contacts.

The scope of this document is to describe the API needed to connect Value Added Service providers in order that they are able to get the necessary platform data and that they can deliver insights back to the platform. The insights from the VAS providers could be optimized network design, cargo pooling, imbalanced lane optimization between different European logistic corridors and backhaul opportunity detection. Deliverables D3.7 and D3.8 will give detailed examples of such insights. The VAS providers will deliver their insights via Apps on the platform.

## 2.2 Data scope

Currently, the scope of the data used will be

- historical transport instructions

but the scope of the solution will need to be extendable in the future, and could include

- provided by shippers (not LSP or 4PL parties)
- real-time or future/forecasted transport instructions
- available capacity on specific lanes
- patterns of capability (i.e. for multimodal/rail offerings)

## 2.3 Functional scope

Initial scope will include

- the ability to search the database to identify possible opportunities (manual search)
- support for optimizers (i.e. using an adapter to retrieve the data needed and to inject the resulting opportunities)

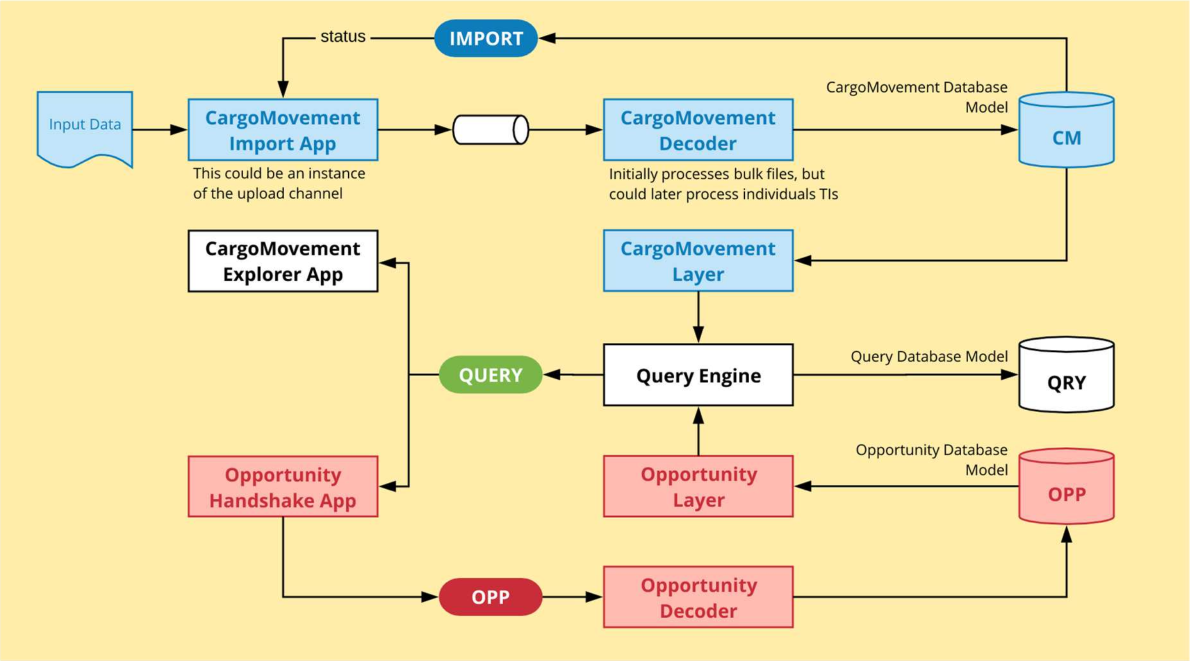
In the future, we will want to add things like

- providing saved searches
  - ability to re-run a saved search when new data comes in
  - automatically run when new data comes in and generate notifications
- added data layers for LSP capacity and capability
- additional search possibilities (beyond perimeter or location aggregation level)

## 2.4 Architectural view

The overall architecture of the data input and interfacing with the CargoStream platform is explained in figures 3 to 5.

**Figure 3 – CargoStream architecture for data input and retrieval**


status
IMPORT
Input Data
CargoMovement Import App
This could be an instance of the upload channel
CargoMovement Decoder
Initially processes bulk files, but could later process individuals TIs
CargoMovement Database Model
CM
CargoMovement Explorer App
CargoMovement Layer
QUERY
Query Engine
Query Database Model
QRY
Opportunity Database Model
Opportunity Handshake App
Opportunity Layer
OPP
OPP
Opportunity Decoder

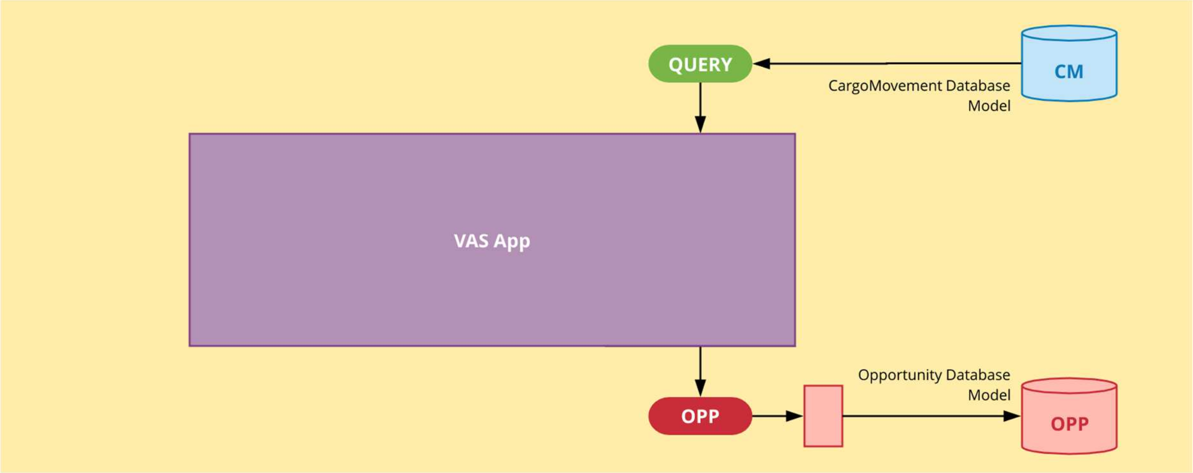
As shown in figure 3, the main channel for data input is an import app allowing shippers to upload historical transport instructions. This app will also support the follow-up of the processing of the data provided.

Central to the architecture is the availability of the QUERY API, which is used to retrieve all cargo movement data. A first app to use this API is the Explorer app, which allows a user to search the database based on geographical (location, proximity …), temporal (time range and bucketization) and other properties (type of transport …) and visually represent the results.

The API will have the capability to report on multiple layers of data matching the query specifications. Minimally, this will include the **historical transport instructions** (aggregated as requested) and existing **collaboration opportunities**, suggested by either a manual creation (using the Explorer) or generated by a VAS App.

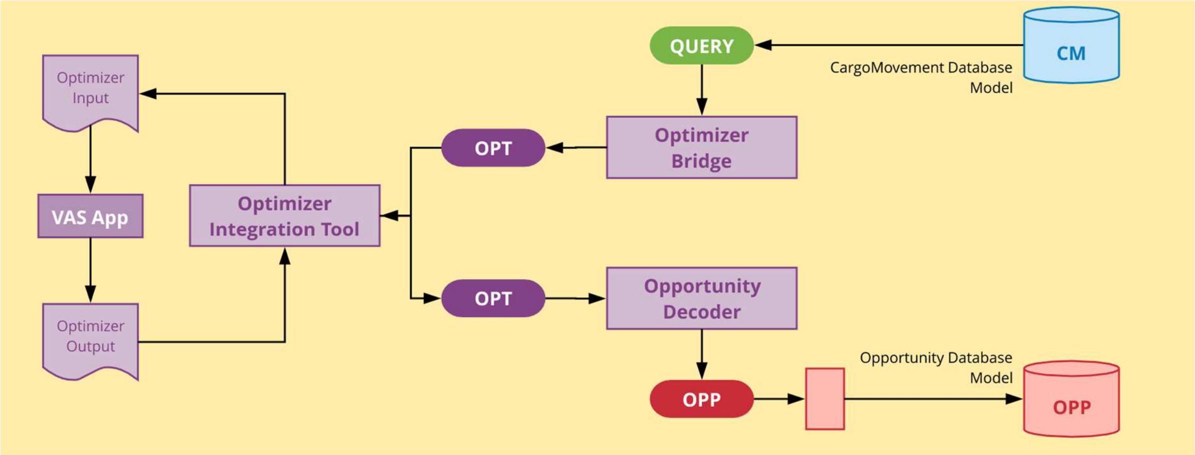
Basic opportunities can be created by a shipper using the Explorer, but this would be more of an exception situation: the bulk of the opportunities will be detected and created by VAS Apps.

**Figure 4 – CargoStream architecture for VAS App interfacing**



A VAS App (see figure 4) will normally use the same query API to retrieve an aggregated data set. However, we anticipate that an extended architecture may be required to provide additional integration options, for example for optimizers which work in a more offline mode. In this case, figure 5 would be a more accurate representation of the architecture used.

**Figure 5 – CargoStream architecture for extended VAS App interfacing**



## 2.5 Interfacing and access scope

The integration between the VAS App and the platform will be based on **OpenID/OAuth2,** requiring the VAS to use an access token when calling the API. In an initial step, the VAS App needs to request and receive a **client ID/client secret** from the platform in order to be able to identify itself as an accepted client application. The client ID is required for **authentication**.

An access token is received as result of an **authentication** request. For this, the VAS App will need authenticate the user requesting the data from the platform as the OAuth2 identity. This is usually done by an OpenID redirect request to the platform login page, where all parties providing information are registered. Alternately, a machine-to-machine authentication flow can be used, which does not require user redirection.

As a result of an authentication request, the platform will issue an **access token** which can then be used to call the API.

## 2.6 Current situation and future development paths

Today, end 2017, and independently from Clusters 2.0, CargoStream has 2 applications from third party companies, testing the business flow and investigating how to connect to the platform. Next to those existing applications, and within the scope of the Clusters 2.0 project, Argus-i and PTV are investigating how to connect their knowledge to the platform. PTV looks how to bring operational insights to CargoStream users, Argus-i does it from a tactical point of view. More detail on both will be given in D3.7 and D3.8 and will depend on what the learnings from the living labs and from D5.3 will bring.

# 3. Data model

CargoStream ingests historical **transport instructions** and converts them into **cargo movements**:

- it isolates the origin and destination **locations** and geocodes them to enrich the location info with coordinates and location aggregation identifiers, and prepare them for easy searching
- it creates a **lane** for this shipper indicating that there are loads traveling along this from/to pair
- it may create additional indexes and intermediate tables which facilitate searching.

All data related to a shipper is linked to that shipper. Community data is created on the fly when retrieving it, as the definition of ‘community’ is different for every request.

The description of the data input to be uploaded to CargoStream is provided in D3.1. We will not repeat that part here. However, it is important to foresee a methodology of continuous updating of data and a methodology to improve the data input list.

## 3.1 Data update process

On an individual shipper level, data that has been uploaded to the platform must regularly be updated to guarantee accurate optimization suggestions. The platform gives the possibility to do so for shippers who connect via a B2B integration (EDI, XML, …), but also for shippers who prefer manual uploads of .xls or .csv sheets. This is explained in chapter 4 of D3.1.

## 3.2 Data input list improvement process

Todays’ data input list is based on knowledge from the current CargoStream members and their requirements. This is mainly knowledge from the shippers’ side. As a lot of transport optimization knowledge is at VAS provider level, we expect that regular requests for extra data elements will be made by them. That is why the data input list will evolve. We will make a version overview of the data input list available for all members and we will enable existing members to enlarge their currently uploaded data towards the latest version of the data input list.

Ideally if a shipper wishes to use an App on CargoStream, this App will inform the shipper if his available data set is insufficient and the App will suggest which data elements should be uploaded extra.

# 4. Retrieving data

Any request for cargo movement data will pass through a central API which passes the request to one or more data sources. Any API request will need to indicate

- the data layer to retrieve (identifying the data source to query)
- the selection criteria for the query
- the aggregation level requested
- the details to return.

In both cases, the API will require an authentication by the requester, identifying the party (for instance, the shipper) requesting the information.

## 4.1 Simple queries using GET /data

A first option will be to use a short-form GET request to **/data** using URL parameters. Example:

```
GET HOST/data?starting=20170101&ending=20171231
```

## 4.2 Complex queries using a Query object

A more complex approach is to require the creation of a **Query** structure:

- the caller creates this by a **POST /query** returning a query JSON with a unique id
- updates / modifications are possible using a standard **PUT /query/<id>** request
- results can be retrieved using **GET /query/<id>/data**

Example of a Query structure returned:

```
{
    id: “032b4d76-2676-4db8-be9a-8ad78438ffe7”,
    requester: “b4aeab44-b73a-4849-a2f2-4821484d9cb4”,
    createdAt: “2017-09-22T14:33:21Z”,
    modifiedAt: “2017-09-22T14:49:17Z”,
    select: {
       …
       },
    groupBy: {
       …
       },
   return: {
       layers: [ “cmh”]
       }
}
```

# 5. Request parameters

## 5.1 Data layer

Initially, we will have only one data source (one data layer), but as we know this will change, we want the architecture to reflect this immediately. The layers will be identified by providing

- a URL parameter like

```
layers=layer1,layer2
```

- a JSON request option like

```
{
    layers: [ ”layer1”, “layer2” ]
}
```

The initial data layer will be called `cmh` (short for cargo movements, historical). If no `layers` value is specified, we will assume `cmh` is intended.

Each data layer needs to specify which data will be returned and how the selection criteria will be applied. For historical cargo movements, these will be minimally aggregated by **lane** (individual cargo movements are never returned).

## 5.2 Range criteria

### 5.2.1 Temporal criteria

This selection would allow to include only movements in a specified time period, expressed on the TransportDateTime reference value.

| Field name | Field type | Field description / condition |
|---|---|---|
| **starting** | DateTime | TransportDateTime >= starting |
| **ending** | DateTime | TransportDateTime <= ending |

These parameters would be mandatory, as there is no sensible default value.

### 5.2.2 Location and geo criteria

On the from and/or to locations, it is possible to select

- the value at one of the aggregation levels (city, region …) or a match pattern for it
- two- or three-digit post codes of a set of these
- locations within a certain region around a specified point
    - a circle with a specified diameter in KM
    - other geometric constructs (hyperboles, subplanes …)
- only lanes in the direction specified by the search, or include return loads

| Field name | Field type | Field description / condition |
|---|---|---|
| **fromCity** | Text(35) | |
| **fromRegion** | Text(35) | Should match region as identified by the response by geocoding |
| **fromPostalCode** | Text(17) | |
| **fromCountryCodeISO2** | Text(2) | |
| **fromRange** | JSON | A description of a range, combining all of the parameters below |
| **lat** | Decimal degrees: DDD.DDDDD | Latitude |
| **lon** | Decimal degrees: DDD.DDDDD | Longitude |
| **radius** | Decimal (10; 5,5) | Radius of the search circle in km |

Each of these parameters (except the range) can be an array of values, allowing for multiple selections to be specified in one query.

The range parameters for URL use are **fromRangeLat**, **fromRangeLon** etc.

When using a Query structure, these parameters can be used as follows:

```
{
   select: {
      from: {
           city: “Vilvoorde”
           },
      to: {
           countryCodeISO2: “FR”
           }
      }
}
```

```
{
   select: {
      from: {
           range: {
               lat: 50.8503,
               lon: 4.3517,
               radius: 50
               }
           }
      }
}
```

### 5.2.3 General criteria

Callers need to be able to limit the selection using a number of parameters. Each of these parameters can be an array of values, allowing for multiple selections to be specified in one query.

| Field name | Field type | Field description / condition |
|---|---|---|
| **transportModeCode** | Codelist (4) | 10 = Maritime transport<br>20 = Rail transport<br>30 = Road transport<br>40 = Air transport<br>60 = Multimodal transport<br>80 = Inland water transport |

| Field name | Field type | Field description / condition |
|---|---|---|
| **packagingCategory** | Text (3) | CNT = container<br>BUL = Bulk (solid)<br>LIQ = Liquid (bulk)<br>PAL = Pallets<br>BOX = Boxes<br>DRM = Drum<br>OTH = Other |

## 5.3 Aggregation level

### 5.3.1 Aggregation parameters (GROUP BY)

#### *5.3.1.1 Shipper and community*

Normally, movements will be aggregated as ‘this shipper’ versus ‘the community’. We need to investigate how we want to handle situations where we need to show the individual shippers, without disclosing their names, to facilitate handshakes.

#### *5.3.1.2 Lane*

If requested, the totals per party are subdivided into geo parameters (at the most in individual lanes).

#### *5.3.1.3 Time buckets*

Bucketization (for instance by month) can be useful to provide insight into the periods where we actually have data, and to show seasonality. Options offered for the **groupBy.buckets** parameter will be

- week aggregates by year + weeknumber, e.g. 2017W48
- month aggregates by year + month, e.g. 201711
- quarter aggregates by year + quarter, e.g. 2017Q1
- (none) resulting in all data aggregated for the entire period

### 5.3.2 Filter criteria (HAVING)

Select only lanes with a minimum weight, volume or instruction count, like:

| Field name | Field type | Field description / condition |
|---|---|---|
| **grossWeightMin** | Decimal (18; 12,6) | Range for the aggregated gross weight of the lane |
| **grossWeightMax** | Decimal (18; 12,6) | |
| **grossVolumeMin** | Decimal (18; 12,6) | Range for the aggregated gross volume of the lane |
| **grossWolumeMax** | Decimal (18; 12,6) | |
| **movementCountMin** | Integer | Range for the aggregated number of movements of the lane |
| **movementCountMax** | Integer | |
| **distanceMin** | Decimal (10; 5,5) | Range for the straight-line distance of the lane |
| **distanceMax** | Decimal (10; 5,5) | |

## 6. Response format

The response format will have to be defined to be very easy to parse and use in calling applications. In general terms, the response will contain

- a request parameter section, repeating the criteria used to generate the data
- a summary section, with overall totals (number of movements, gross weight …)
- a detailed data section
    - level 1 grouping by buckets
    - level 2 grouping by party
    - level 3 grouping by lane or geo property (city or region)
    - each 'cell' containing info on how many parties contributed, how many records provided data for the total (to calculate sparsity) …

This format will need to be evaluated based on specific needs by users as well.

Prototype for the detailed data section:

```
bucket;party;laneCount;movementCount;grossWeight;grossWeightPresent;grossVolume;grossVolu
mePresent;***
2017W01;self;3;22;15000;22;352;7;***
2017W01;community;17;241;423745;241;65433;114;***
…
```

For the first week of 2017, the requesting shipper had 22 movements in 3 lanes totaling 15000 KGM, while only 7 had a volume field specified.

*** stands for the lane details, if such aggregation is requested:

```
fromCity;fromRegion;fromCountry;toCity;toRegion;toCountry;laneDistance
Vilvoorde;Brussels;BE;Roma;Roma;IT;1325.54
```

## 7. Conclusion and outlook

This API description document is a first version of how the CargoStream API towards external applications from VAS providers could look like. In order to build the API, detailed feedback from VAS providers is necessary and iterate testing should be organized. At this stage, is too early for API test iterations. We will focus during M9-M18 on feedback from dedicated VAS providers and preparation of a first version of the API.