



MARINFO

Interface Specification

Version 1.0.0

Date: 30/05/2023

Table of Contents

| | |
|---|-----------|
| 1. Introduction | 3 |
| 2. FTP | 4 |
| 2.1 Introduction | 4 |
| 2.2 MAR_VERIF4.STEPS | 4 |
| 2.3 MAR_VERIF4.PARAM_FTP | 5 |
| 2.4 MAR_VERIF4.PARAM_LOTS | 5 |
| 2.5 MAR_VERIF4.PARAM_FILES | 6 |
| 2.6 MAR_VERIF4.PARAM_FILE_FIELDS | 6 |
| 3. API | 7 |
| 3.1 Architectural Design | 7 |
| 3.2 Overview of available API calls | 7 |
| 3.3 Developer Portal | 9 |
| 3.3.1 EMSA Developer Portal URL per Environment | 9 |
| 3.3.2 Signing Up and logging in to the developer Portal | 10 |
| 3.3.3 Creating and Requesting a Subscription | 10 |
| 3.3.4 Testing the API on the Developer Portal | 11 |
| 3.3.4.1 Understanding the different sections of information on the portal | 13 |
| 3.3.5 Trying out the API | 16 |
| 3.4 Testing the API with external applications (Postman) | 18 |
| 3.5 Possible API Calls & Usage | 21 |
| 3.6 Different Types of API Calls for the same data | 21 |
| 3.6.1 Definition of the Request Parameters | 21 |
| 3.6.2 Usage of the Request Parameters | 22 |
| 3.7 Where/How to manage API Management Service | 23 |
| 3.7.1 API's | 23 |
| 3.7.2 Users | 24 |
| 3.8 Who to Contact? | 24 |
| CSV Exports | 25 |
| 3.9 Introduction | 25 |
| 3.10 MAR_VERIF4.VIEWS | 25 |

1. Introduction

In this document you will find a current specification of the FTP interface: where do the data providers deliver their files, which are the rules implemented for accepting (or not) the files, and which specific rules for the fields to be accepted and introduced in the MARINFO database. In a second part, we list the current APIs (web services) specifications, where we define for each web services the url(s) to use, the parameters to be used for the calls, and the expected results. Lastly, we will have a look at the csv exports. Which files are being exported and where to find them.

2. FTP

2.1 Introduction

Marinfo files are delivered as zip files on an external FTP server. These files are extracted by Data Factory and stored on the datalake. During the process of transforming and ingesting these files into the SQL database, a number of checks are performed on this data. The parameters of this process are stored on the MARINFO database. It contains the list of files that need to be loaded, a step by step description of the ETL flow, description of what is inside the zip files and the order they need to be loaded, and a description of each column inside each table with their expected specifications. All these describe the rules of the ETL process from fetching the data from the FTP to processing them and storing them in the SQL database.

The full pipeline can be found in data factory, by clicking the link below:

https://adf.azure.com/en/authoring/pipeline/Ingest_data?factory=%2Fsubscriptions%2F49dbba4a-f9cd-453f-acce-1f036b6c5e2c%2FresourceGroups%2Fmarinfo-prod-west-europe%2Fproviders%2FMicrosoft.DataFactory%2Ffactories%2Fmarinfo-prod-west-europe-df

2.2 MAR_VERIF4.STEPS

This table, available on the MARINFO database, serves as a description of all steps in the ETL process. The process can be described as follows:

1. Check which zip files need to be fetched from the FTP server.
2. Try to extract a data from the zipfile name. These dates are used to determine the order of processing.
3. The ZIP file has been successfully unzipped on a temporary folder
4. The delivery has been retrieved and the packaging looks OK
5. Phase 1 has been started (validation of the delivery)
6. A file named "Checksum.txt" is present in the delivery
7. The Checksum File has a correct structure
8. The Checksum File has been loaded in the database
9. The zipfile should contain the first digit of the lot number of the zip file. This is necessary because different lots can have files with the same name as input.
10. All files have authroised method and it is clear whether the loading procedure is total or incremental
11. Check if the number of files in the CheckSum matches the number of files in the zip
12. Check whether there are duplicated files in the Checksum file
13. The Checksum File has entries for all the files in the delivery
14. All files in the delivery are described in the Checksum file
15. The MD5 of the files in the checksum have been verified successfully

16. The files provided in the zip file are known files in the database
17. This check only applies to zip files with .all method. Check that all the files of the lot are present
18. Check whether the files have the correct columns and whether they are in the correct order.
19. The number of rows for each file has been successfully checked against the value in the Checksum file
20. The content of each file of the delivery is syntactically correct
21. Check the primary key, foreign key and mandatory column requirement
22. The files have been checked and loaded successfully
23. Materialize the views that are needed for the webservices
24. This step gets logged incase there has been a severe unforeseen error in the processing of the zip file.

2.3 MAR_VERIF4.PARAM_FTP

This table contains all necessary details on how to access the external FTP server, and which files we need to look for.

| ID | LOT_ID | PROTOCOL | HOSTNAME | PORT | FTP_USER | PATH_TO_FILE | FILEMASK | TRUEFTP | CONTACT_EMAIL |
|----|--------|----------|----------|------|----------|--------------|----------|---------|---------------|
| 1 | 10 | SFTP | | | | | | | |
| 2 | 12 | SFTP | | | | | | | |
| 4 | 21 | SFTP | | | | | | | |
| 5 | 22 | SFTP | | | | | | | |
| 6 | 20 | SFTP | | | | | | | |
| 7 | 23 | SFTP | | | | | | | |
| 9 | 24 | SFTP | | | | | | | |
| 10 | 25 | SFTP | | | | | | | |
| 11 | 11 | SFTP | | | | | | | |
| 12 | 26 | SFTP | | | | | | | |

Please note that in above picture, some confidential information of how to access the data is left out. This can be accessed by querying the table on the MARINFO database. These zip files will be extracted and loaded on the MARINFO datalake.

2.4 MAR_VERIF4.PARAM_LOTS

This table contains detailed information about the LOTS of MARINFO, how the records inside are separated, and in which order they should be loaded.

| ID | LOT_NUMBER | LOT_DESCRIPTION | PROVIDER_NAME | RECORD_SEPARATOR | LOAD_ORDER | NB_TRY | NB_SUCCESS_LOAD |
|----|------------|---|---------------|------------------|------------|--------|-----------------|
| 10 | 1A | Modula A - Port Calls | | \r\n | 2 | 0 | 0 |
| 11 | 1A | Module A - Berth Callings | | \r\n | 3 | 0 | 0 |
| 12 | 1B | Module B - Ports and Anchorages | | \r\n | 1 | 0 | 0 |
| 20 | 2A | Module A - Ship current details and history | | \r\n | 4 | 0 | 0 |
| 21 | 2B | Module B - Casualties | | \r\n | 5 | 0 | 0 |
| 22 | 2C | Module C - Port State Control | | \r\n | 6 | 0 | 0 |
| 23 | 2D | Module D - Ship Engine Data | | \r\n | 7 | 0 | 0 |
| 24 | 2E | Module E - Demolition of ships | | \r\n | 8 | 0 | 0 |
| 25 | 2E | Module E - Newbuilding | | \r\n | 9 | 0 | 0 |
| 26 | 2F | Module F - STS | | \r\n | 10 | 0 | 0 |

2.5 MAR_VERIF4.PARAM_FILES

This table contains the information on which files are available within each of the LOT zipfiles, and how to process them. Below image shows some of the fields available in the MARINFO database table.

| ID | LOT_ID | FILEFAMILY | DATATABLENAME | LOAD_ORDER | FIELD_SEPARATOR | ENCODAGE | HISTORY | FILE_DESCRIPTION |
|------|--------|------------------------|-----------------------|------------|-----------------|----------|---------|---|
| 1001 | 10 | lot1a_movementdata | MOVEMENTS | 1 | . | ISO-8859 | N | Port calls (including ports and anchorages) an... |
| 1101 | 11 | lot1a_berthcallings | BERTH_CALLINGS | 1 | . | ISO-8859 | Y | Berth callings |
| 1201 | 12 | lot1b_portsdata | PORTS | 1 | . | ISO-8859 | Y | Ports and anchorages |
| 1202 | 12 | tblportberth | PORT_TANKER_BERTH | 4 | . | ISO-8859 | Y | Tanker Berths |
| 1203 | 12 | tblportterminal | PORT_TANKER_TERMINAL | 3 | . | ISO-8859 | Y | Tanker Terminals |
| 1204 | 12 | tblzones | ZONES | 2 | . | ISO-8859 | Y | Zones |
| 1205 | 12 | tblportmarpol | PORT_MAR_POL | 6 | . | ISO-8859 | Y | Marpol |
| 1206 | 12 | tblportberthconnection | PORT_BERTH_CONNECTION | 5 | . | ISO-8859 | Y | Berth connections |

It provides information on things such as load_order, field_seperator, expected encoding, whether a history table is needed and a description of what is inside the file.

2.6 MAR_VERIF4.PARAM_FILE_FIELDS

Lastly, this table contains details/restrictions on each field of all files. Some of these are shown in the picture below.

| FILE_ID | FIELD_ID | FIELD_NAME | DATA_TYPE | DATA_LENGTH | DATA_FORMAT | PK | MANDATORY | FULL_UPD_KEY | DATA_DESCRIPTION |
|---------|----------|---------------------|-----------|-------------|---------------------|------|-----------|--------------|--|
| 1001 | 1 | CALLID | INTEGER | 10.00000 | NULL | 1 | Y | 0 | NULL |
| 1001 | 2 | LRNOIMOSHIPNO | NVARCHAR | 7.00000 | NULL | NULL | Y | 0 | The IMO DECIMAL based on the LR No is a unique Nu... |
| 1001 | 3 | SHIPNAME | NVARCHAR | 50.00000 | NULL | NULL | Y | 0 | Current name of the vessel |
| 1001 | 4 | SHIPTYPE | NVARCHAR | 255.00000 | NULL | NULL | Y | 0 | Statcode5 is an industry standard shiptype descriptive s... |
| 1001 | 5 | MOVEMENTTYPECODE | NVARCHAR | 1.00000 | NULL | NULL | N | 0 | NULL |
| 1001 | 6 | COUNTRY | NVARCHAR | 40.00000 | NULL | NULL | N | 0 | Free text. This field will be blank where record is a trans... |
| 1001 | 7 | OLDPORTID | NVARCHAR | 6.00000 | NULL | NULL | N | 0 | NULL |
| 1001 | 8 | PORTID | INTEGER | 10.00000 | NULL | NULL | N | 0 | Field will be blank where record is a transit |
| 1001 | 9 | PORTNAME | NVARCHAR | 40.00000 | NULL | NULL | N | 0 | Each port/anchorage is a defined zone as applied by o... |
| 1001 | 10 | PORTGEOID | INTEGER | 10.00000 | NULL | NULL | N | 0 | Port ID can be linked to Port ID in the Ports data set |
| 1001 | 11 | ZONENAME | NVARCHAR | 100.00000 | NULL | NULL | N | 0 | NULL |
| 1001 | 12 | ANCHORAGEPARENTID | INTEGER | 10.00000 | NULL | NULL | N | 0 | NULL |
| 1001 | 13 | ANCHORAGEPARENTNAME | NVARCHAR | 40.00000 | NULL | NULL | N | 0 | NULL |
| 1001 | 14 | ARRIVALDATE | DATE | 10.00000 | YYYY-MM-DD | NULL | N | 0 | Date vessel enters port zone |
| 1001 | 15 | ARRIVALDATEFULL | DATETIME | NULL | YYYY-MM-DD HH:mm:ss | NULL | N | 0 | NULL |

For each field we check what is the expected data type, what is the expected maximum length of the content of the field, which format we expect for date/datetime fields, whether these fields are mandatory/primary key fields etc... These rules are checked during the ETL process and need to be followed by external data providers.

3. API

3.1 Architectural Design

The general purpose of this API data exposure is to query data from the model that is stored in an SQL database and expose this to the user. That way the user can customize this query to his / her needs. We expose the datamodel through the API Developer Portal.



Figure 1: Design of the current setup

3.2 Overview of available API calls

For the EMSA MARINFO project, we agreed on exposing the data model through several API calls which are available for the EMSA users. Below you can find a list of all the available calls (GET & POST):

Get

- Company Current
- Company Details
- Company Fleet
- Company Previous
- Crew
- Incidents
- Inspections
- Portcalls
- SatCom
- Vessel Call Sign History
- Vessel Flag History
- Vessel MMSI History
- Vessel Name History

Post

- Company Current
- Company Details
- Company Fleet
- Company Previous
- Crew
- Incidents

- Inspections
- Portcalls
- SatCom
- Vessel Call Sign History
- Vessel Flag History
- Vessel MMSI History
- Vessel Name History

3.3 Developer Portal

To further assist the user in executing API Calls and retrieving data from the model. An [EMSA developer portal](#) has been deployed and published. We can see this as a UI Layer on top of the existing API Management resource in Azure which allows the user to execute and test API calls to retrieve the correct data from the data model.

3.3.1 EMSA Developer Portal URL per Environment

- Developer Environment: <https://marinfo-dev-we-apim.developer.azure-api.net/>
- Test Environment: <https://marinfo-test-we-apim.developer.azure-api.net/>
- Production Environment : <https://marinfo-prod-we-apim.developer.azure-api.net/>

When clicking on one of the previous URL's, the user should land on the following page

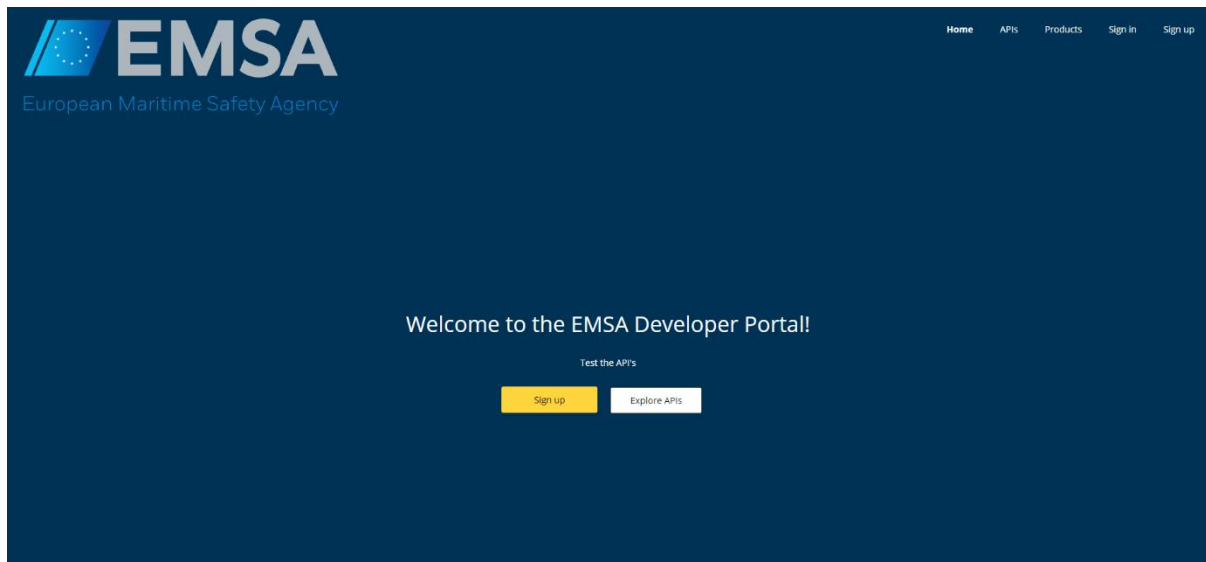


Figure 2: EMSA Developer Portal Landing Page

If a user wants to make use of the developer portal, he / she can do this by performing the steps explained in the rest of this document.

3.3.2 Signing Up and logging in to the developer Portal

In order to be able to execute API Calls, the user has to sign up first to get a subscription. This subscription will allow him / her to start testing the API.

Important Note: The subscriptions and accounts are NOT shared across environments. If a user wants to test API calls on the TEST and on the DEV environment, he will have to sign up separately for each environment (by clicking on both URL's above).

1. To start signing up, the user should click on one of the 'Sign Up' buttons shown on the picture below.



Figure 3: Developer Portal Sign Up Clarification

2. This will take the user to the following page, here the user should fill in the required information in order to sign up to the developer portal

Sign up

Already a member? [Sign in.](#)

| | |
|--------------------|--|
| Email * | <input type="text" value="e.g. name@example.com"/> |
| Password * | <input type="password"/> |
| Confirm password * | <input type="password"/> |
| First name * | <input type="text" value="e.g. John"/> |
| Last name * | <input type="text" value="e.g. Doe"/> |

Figure 4: Sign Up Form

3. After filling in all the required information and resolving the captcha. The user should have received a new mail on the e-mail address that was filled in in the sign up form.
4. After confirming your e-mail address you are now able to log in to the developer portal. Click on the Sign In button and provide the correct information.

3.3.3 Creating and Requesting a Subscription

Before the user can start testing the API's, there is one more step that needs to be completed, and that is creating and requesting a subscription for the user. The API Developer Portal has a built-in security rule that every call to the system has to be done with a valid subscription key (Ocp-Apim-Subscription-Key).

Important Note: When a user requested a (new) subscription, it is advised to contact and send an email to the responsible people for the API's & the Developer Portal (See 3.7). They will validate and accept your subscription which is obligatory for testing out the API.

To request the subscription key the user has to:

1. Go to Products (in the top right corner, in the navigation bar) and click on the MARINFO Product. Since this is the product that is linked to all the current API's.
2. Normally the user will see that he / she has no subscriptions under 'Your subscriptions'. To request a subscription, give a name to your subscription – best practice is to respect the following format:
 - MARINFO <FirstName> <LastName>
 - Example: MARINFO Senne Eeraerts

After completing the above step, the user will be redirected to the 'Profile Overview'.

Important to note is that the State of the subscription will be 'Submitted' until someone from EMSA accepts this Subscription.

If the user loses this subscription key it can always be displayed under the 'Profile' tab. (press Show on the right side of the subscription key).

Subscriptions

| Subscription details | | | Product | State | Action |
|----------------------|--|---|---------|-----------|------------------------|
| Name | MARINFO SENNE EERAERTS | Rename | MARINFO | Submitted | Cancel |
| Requested on | 05/27/2021 | | | | |
| Primary key | xx | Show Regenerate | | | |
| Secondary key | xx | Show Regenerate | | | |

Figure 5: Profile Overview

3.3.4 Testing the API on the Developer Portal

After clicking on APIs → MARINFO, the user will get an overview of the available API calls.

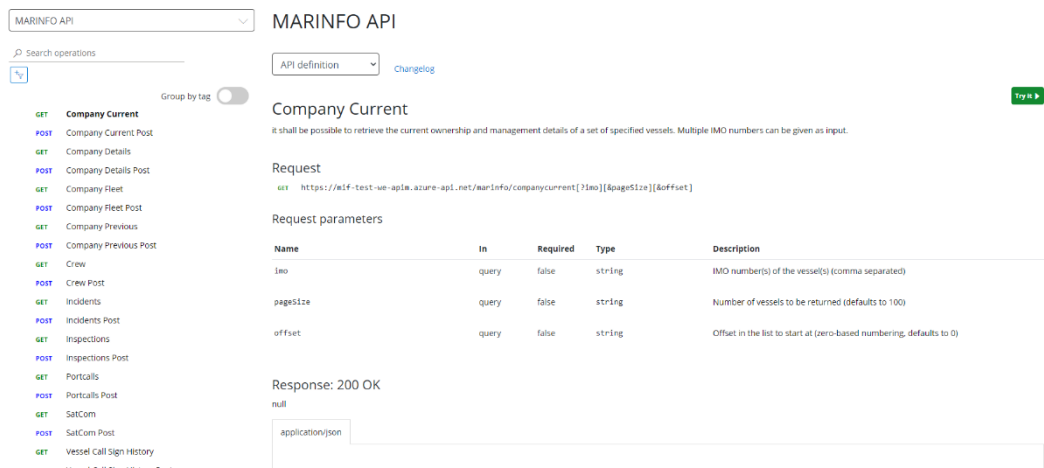


Figure 6: API Call Overview

For this example, we will make use of the GET call of 'Company Current'.

3.3.4.1 Understanding the different sections of information on the portal

The screenshot above shows an overview of information about the API call. This page is split up into the following sections:

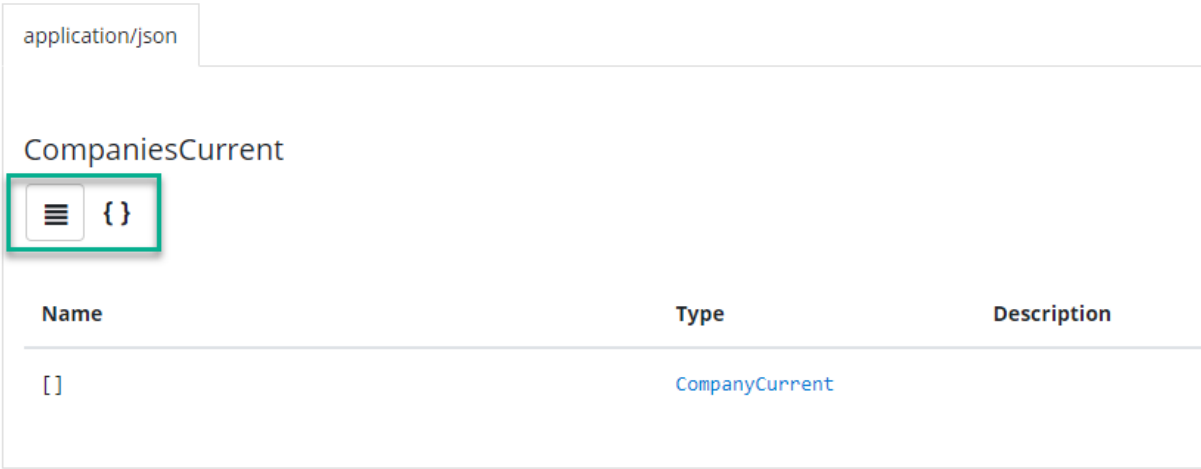
- Name and Description of the API call.
- The Request URL - this is the URL that should be called when testing this API from outside the developer portal
- Request Parameters:
 - Name
 - In
 - Where should the Parameter be provided, in this case it's in the query
 - Required
 - Whether the parameter is required or not in the request
 - Executing the api call without providing the correct required parameters is not possible.
 - Type of the parameter¹
 - String: An alphanumeric sequence of letters and/or numbers
 - Integer: A whole number
 - Boolean: True or False value
 - Object: Key-value pairs in JSON format
 - Array: A list of values
 - Description
 - A brief description of the meaning of the parameter

¹ https://dratherbewriting.com/learnapidoc/docapis_doc_parameters.html

- Response
 - There are a lot of response status codes, the most commonly used are:²
 - 200 (OK)
 - 301 (Moved Permanently)
 - 401 (Unauthorized)
 - 403 (Forbidden)
 - 404 (Not Found)
 - 408 (Request Timeout)
 - 500 (Internal Server Error)

Response: 200 OK

null



| Name | Type | Description |
|------|----------------|-------------|
| [] | CompanyCurrent | |

Figure 7: Response Status Code + Representation

As seen in the screenshot above, there are 2 clickable buttons under the Response section.

- The first button will give you the name of the representation of this status code. This means that when the response 200 (OK) is returned, this data object will be returned from the API.
- The second button is a code example of the representation (data structure in JSON)

Generally speaking, the response representation is split into CompaniesCurrent and CompanyCurrent. This is because the complete data that this call returns is a set (array) of objects (CompaniesCurrent) and 1 object represents CompanyCurrent. Please refer to figure 7 & 8 to see the different structures with examples.

² <https://blog.bearer.sh/common-http-status-codes/>

CompaniesCurrent

{ }

JSON

Copy

```

{
  "type": "array",
  "items": {
    "$ref": "#/definitions/CompanyCurrent"
  }
}

```

Figure 8: CompaniesCurrent Array Representation

CompanyCurrent

{ }

JSON

Copy

```

{
  "required": [
    "IMO",
    "REGOMINERNAME",
    "REGOMINERCODE",
    "GROUPBENEFNAME",
    "GROUPBENEFCODE",
    "SHIPMANAGERNAME",
    "SHIPMANAGERCODE",
    "OPERATORNAME",
    "OPERATORCODE",
    "TECHMANAGERNAME",
    "TECHMANAGERCODE",
    "DOCCOMPANYNAME",
    "DOCCOMPANYCODE"
  ],
  "type": "object",
  "properties": {
    "IMO": {
      "type": "string",
      "description": ""
    },
    "REGOMINERNAME": {
      "type": "string",
      "description": ""
    },
    "REGOMINERCODE": {
      "type": "string",
      "description": ""
    },
    "GROUPBENEFNAME": {
      "type": "string",
      "description": ""
    },
    "GROUPBENEFCODE": {
      "type": "string",
      "description": ""
    },
    "SHIPMANAGERNAME": {
      "type": "string",
      "description": ""
    },
    "SHIPMANAGERCODE": {
      "type": "string",
      "description": ""
    }
  }
}

```

Figure 9: CompanyCurrent Object Representation

CompanyCurrent is 1 object inside the CompaniesCurrent Array dataset. An example can be found in the screenshot below.

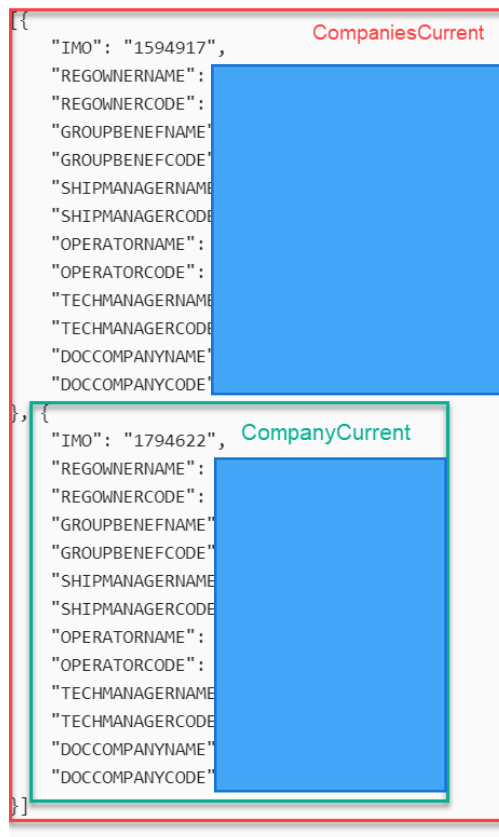


Figure 10: Example Of Representations

3.3.5 Trying out the API

You can try out the API by clicking on the 'Try It' button on the right side of the screen.

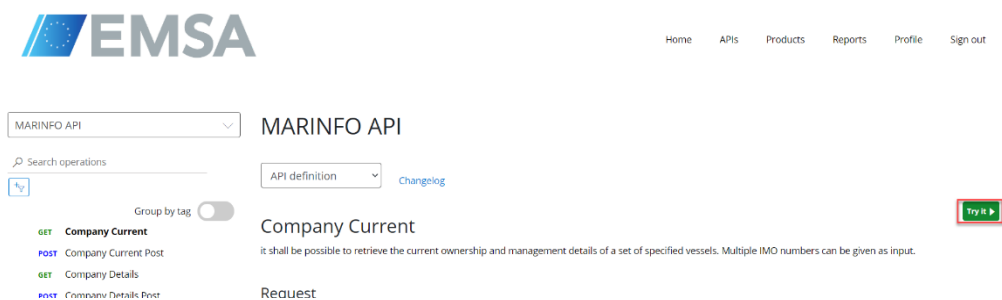


Figure 11: Location of the Try It Button

When your subscription is already validated, the 'Authorization' field will be automatically filled in. The parameters for the Get Company Current request are not required. The query is built in a way that, when no parameters are provided, the request will just return a select of the first 100 rows for that call. Under the 'Headers' section, the user is able to find code snippets which are useful when integrating the API calls in applications in different programming languages such as C#, Java, PHP, Python. They also provide snippets on how to format the request as a HTTP request, or a request via curl etc. After clicking on 'Send', the user is able to see the data returned by the API.

HTTP response

```

HTTP/1.1 200 OK

content-encoding: gzip
content-length: 7513
content-type: application/json
date: Fri, 28 May 2021 09:04:39 GMT
vary: Accept-Encoding,Origin

[
  {
    "IMO": "1594917",
    "REGOWNERNAME": [REDACTED],
    "REGOWNERCODE": [REDACTED],
    "GROUPBENEFNAME": [REDACTED],
    "GROUPBENEFCODE": [REDACTED],
    "SHIPMANAGERNAME": [REDACTED],
    "SHIPMANAGERCODE": [REDACTED],
    "OPERATORNAME": [REDACTED],
    "OPERATORCODE": [REDACTED],
    "TECHMANAGERNAME": [REDACTED],
    "TECHMANAGERCODE": [REDACTED],
    "DOCCOMPANYNAME": [REDACTED],
    "DOCCOMPANYCODE": [REDACTED]
  },
  {
    "IMO": "1794622",
    "REGOWNERNAME": [REDACTED],
    "REGOWNERCODE": [REDACTED],
    "GROUPBENEFNAME": [REDACTED],
    "GROUPBENEFCODE": [REDACTED],
    "SHIPMANAGERNAME": [REDACTED],
    "SHIPMANAGERCODE": [REDACTED],
    "OPERATORNAME": [REDACTED],
    "OPERATORCODE": [REDACTED],
    "TECHMANAGERNAME": [REDACTED],
    "TECHMANAGERCODE": [REDACTED],
    "DOCCOMPANYNAME": [REDACTED],
    "DOCCOMPANYCODE": [REDACTED]
  },
  {
    "IMO": "1891975",
    "REGOWNERNAME": [REDACTED],
    "REGOWNERCODE": [REDACTED],
    "GROUPBENEFNAME": [REDACTED],
    "GROUPBENEFCODE": [REDACTED],
    "SHIPMANAGERNAME": [REDACTED],
    "SHIPMANAGERCODE": [REDACTED]
  }
]

```

Figure 12: Return Data For the CompanyCurrent Call

3.4 Testing the API with external applications (Postman)

It speaks for itself that it's also possible to test the API Calls with other applications such as Postman. There are a few things that have to be taken into account before the user will be able to send successful requests to the API.

Azure APIM always works with an Ocp-Apim-Subscription-Key parameter that has to be provided in the Header of EVERY call to the api. This Subscription key can be found under the Profile section on the Developer Portal. Not providing this key will result in an 'Unauthorized' response error with the following Message:

```
HTTP/1.1 401 Unauthorized

content-length: 152
content-type: application/json
vary: Origin
www-authenticate: AzureApiManagementKey realm="https://mif-dev-we-apim.azure-api.net/marinfo",name="Ocp-Apim-Subscription-Key",type="header"

{
  "statusCode": 401,
  "message": "Access denied due to missing subscription key. Make sure to include subscription key when making requests to an API."
}
```

Figure 13: Example of the Unauthorized Error

This will be the case when trying to send requests via applications like Postman but also on the developer portal itself.

The parameters for the GET request have to be provided in the URL of the request where as the parameters for the POST request have to be provided in the request body. The user can use the following screenshots as a reference when calling via both GET (Figure 13 & 14) and POST (Figure 15 & 16) calls to the API in Postman.

The first call is a GET call:

- <https://marinfo-dev-we-apim.azure-api.net/marinfo/companycurrent?imo=1594917&pagesize=100&offset=0>
 - Take note of the Ocp-Apim-Subscription-Key in the Headers section

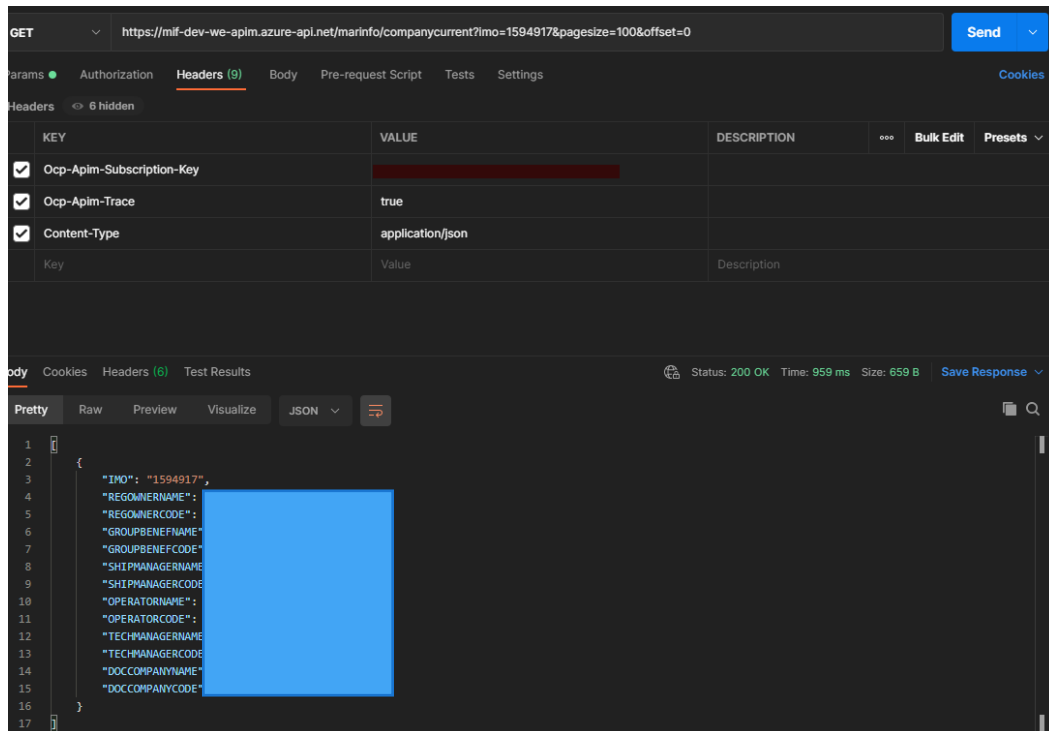


Figure 14: Example Of A GET Call Via Postman

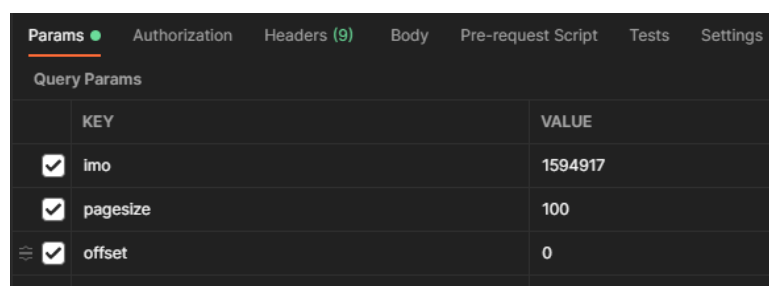


Figure 15: Parameter Example

The second call is a POST call:

- <https://marinfo-dev-we-apim.azure-api.net/marinfo/companycurrent>
 - Don't forget to provide the Ocp-Apim-Subscription-Key in the Headers section
 - It is important to also provide the parameters in the body, starting with a '@' symbol

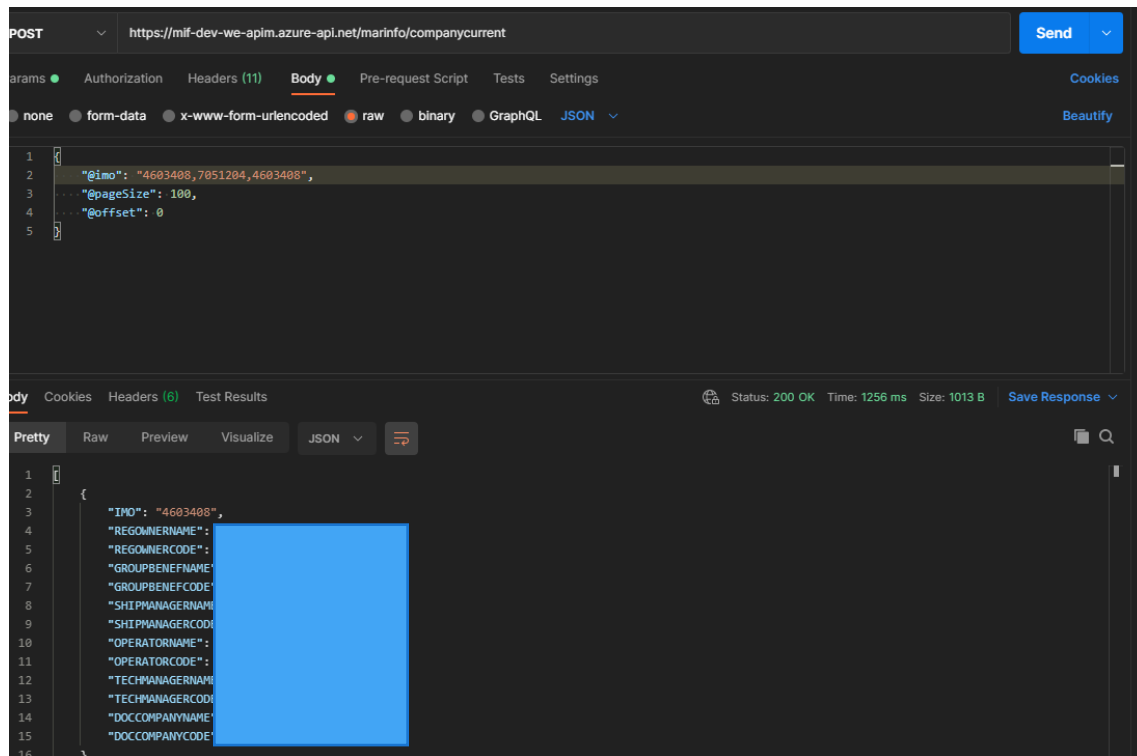


Figure 16: Example Of A POST Call Via Postman

3.5 Possible API Calls & Usage

For the EMSA MARINFO project, the data of the datamodel can be exposed via 2 different types of API calls, GET & POST.

In 3.4, the usage of the GET call was already covered. The way the POST call works is similar to the GET call. The only difference is that the parameters have to be provided in the BODY (keys of the object starting with the '@' symbol) of the call instead of in the URL.

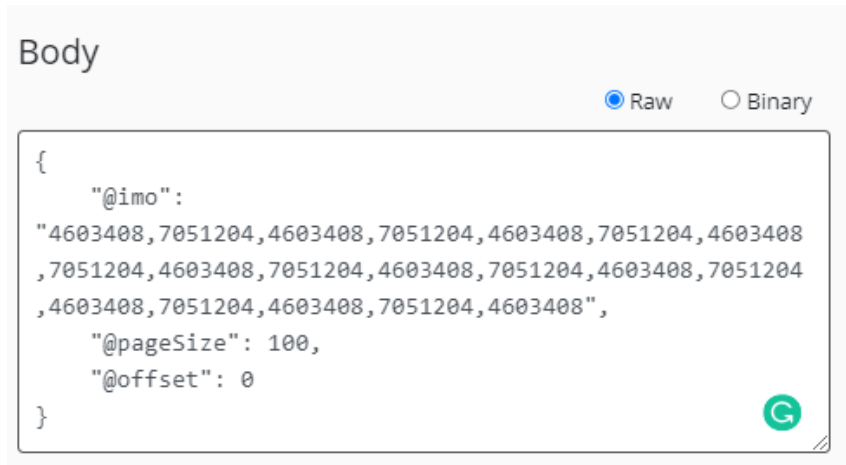


Figure 17: Example Body Of A POST Call

3.6 Different Types of API Calls for the same data

The SQL query is built in such a way that the parameters (in the GET Company Current example this is the 'IMO' parameter) that can be send with the request can be 'comma separated'. This allows the user to provide a list of different IMO numbers in this case. The query will return a record for each match between the provided IMO number and the IMO number in the data model.

The only limitation when using GET requests is that the URL has a max length of 2 048 characters. This implies that when the user sends a request with a list of IMO numbers in the URL that has more than 2 048 characters, the call will fail.

For this reason, we implemented a workaround so that lists exceeding the maximum amount of allowed characters, can still be send to the API. The solution for this is putting them in the body of a POST request, since here is no limitation on character length.

The conclusion is that for every request that has a rather high (> 40) amount of parameters, the POST request should be used instead of the GET request.

3.6.1 Definition of the Request Parameters

- Pagesize

This parameter represents the amount of objects (read: rows) that will be returned for the specific call.

- Offset

These parameters serves the use of a 'skip' in your resultset.

- IMO

Stands for: IMO number(s) of the vessel(s) (comma separated)

3.6.2 Usage of the Request Parameters

- Pagesize

If a GET call retrieves all the rows in the data model, but the pagesize is set to 5, it will only return the 5 first results to the user. (The default value of this parameter is set to 100 when not provided in the request)

- Offset

If the user would not want the first retrieved row for a specific provided parameter, but he wants the resultset to start from the second row that was retrieved, this value should be set to 1. (The default value of this parameter is set to 0 when not provided in the request, this means that no records will be skipped in showing the resultset)

- IMO

The user is able to retrieve a resultset based on providing 1 or many IMO's as parameter. The IMO(S) can be provided as 'comma seperated'. The way this should be implemented is as follows:

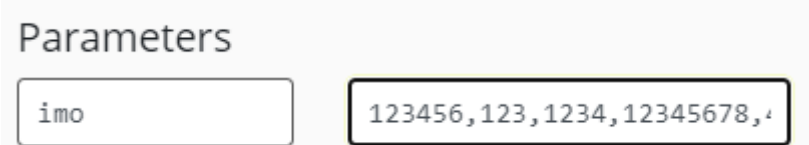


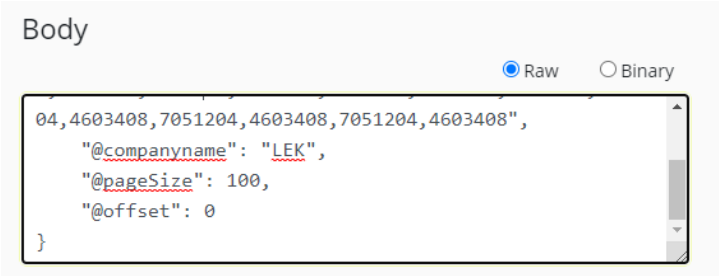
Figure 18: IMO Parameter Comma Seperated Usage

- Company Name

This is *string* parameter that can be provided to the following queries in order to filter the response by Company Name, a list with comma separated values can also be provided. More information can be found in the picture below:

```
GET https://mif-dev-we-apim.azure-api.net/marinfo/companydetails?companyname=EZK%2CDZER%2CDTR HTTP/1.1
```

Figure 19: Get Example Of The Company Details Request



```
{
  "04,4603408,7051204,4603408,7051204,4603408",
  "@companyname": "LEK",
  "@pageSize": 100,
  "@offset": 0
}
```

Figure 20: Post Example Of The Company Details Request

- Start / Enddate

Some queries require a start and end date to be provided as query parameters. The GET request won't be a valid request until both start- & enddate are filled in in the following format: yyyyymmdd (Ex. 19990101).

In the post request the values will be defaulted to startdate = 00000101 & enddate = 99993112.

NOTE:

When the user sends a POST request, and the user does not want to provide a startdate nor an enddate, and wants to retrieve the entire query. The user should not provide the values for these parameters. In an example this would be formatted like so:

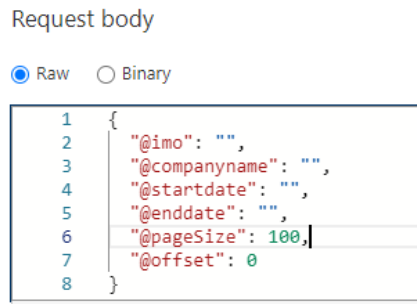


Figure 21: Example Of Executing A Post Request Without Date Parameters

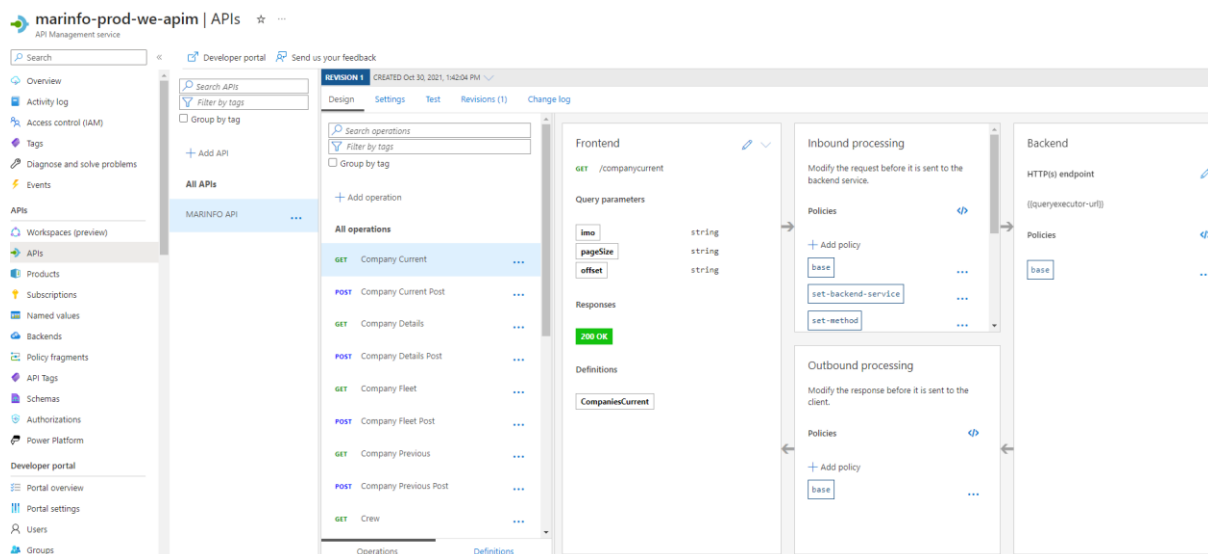
3.7 Where/How to manage API Management Service

<https://portal.azure.com/#/@emsaeuropa.onmicrosoft.com/resource/subscriptions/49dbba4a-f9cd-453f-acce-1f036b6c5e2c/resourceGroups/marinfo-prod-we-r/providers/Microsoft.ApiManagement/service/marinfo-prod-we-apim/overview?api-version=2022-03-01>

Clicking the link above will direct you to the home page of the API management service of the production environment.

3.7.1 API's

In the API tab, the API definitions, designs can be modified and tested.



3.7.2 Users

Access and users can be consulted/managed in the users tab:

marinfo-dev-we-apim | Users

API Management service

Search

«

+ Add

+ Invite

Columns

Refresh

Developer portal

Portal overview

Portal settings

Users

Groups

Identities

Delegation

OAuth 2.0 + OpenID Connect

Issues (deprecated)

Monitoring

Search to filter items...

| Full name | Email | Auth type | Groups |
|-----------|-------|-----------|----------------------------|
| | | Azure | Administrators, Developers |
| | | Basic | Developers |
| | | Basic | Developers |
| | | Basic | Developers |
| | | Basic | Developers |
| | | Basic | Developers |
| | | Basic | Developers |

3.8 Who to Contact?

Available contacts for questions concerning the API's, The Developer Portal and (Creating / Editing)

Profile information are:



CSV Exports

3.9 Introduction

The ETL pipeline has an additional functionality that exports certain views as csv files and stores them on the datalake. These can then be accessed by external third parties that have the necessary rights. The specifics on which data will be exported and where they can be found are also available as master data in the MARINFO SQL database.

3.10 MAR_VERIF4.VIEWS

This table contains the name of the views that will be exported and the path to which they will be exported.

| VIEW_ID | VIEW_NAME | VIEW_SCHEMA | TABLE_NAME | TABLE_SCHEMA | STAGING_SCHEMA | TO_BE_MATERIALIZED | TO_BE_EXPORTED | TO_BE_EXPORTED_PATH |
|---------|-------------|-------------|------------|--------------|----------------|--------------------|----------------|-------------------------|
| 14 | LOTA_MAX_MV | SSN | NULL | NULL | NULL | 0 | 1 | exports/SSN/LOTA_MAX_MV |
| 15 | V_PORTSDATA | SSN | NULL | NULL | NULL | 0 | 1 | exports/SSN/PORTSDATA |

When the ETL pipeline is running, as a final step, it will look in this table and check which data should be placed on the data lake. The above views are being materialized and exported to the datalake.

These csv files can be found when clicking the link below:

https://portal.azure.com/#view/Microsoft_Azure_Storage/ContainerMenuBlade/~/_/overview/storageAccountId/%2Fsubscriptions%2F49dbba4a-f9cd-453f-acce-1f036b6c5e2c%2FresourceGroups%2Fmarinfo-prod-wedl%2Fproviders%2FMicrosoft.Storage%2FstorageAccounts%2Fmarinfo-prod-wedl/path/exports/etag/%220x8D996FD547EF415%22/defaultEncryptionScope/%24account-encryption-key/denyEncryptionScopeOverride~/false/defaultId/publicAccessVal/None

marinfo-prod-wedl | Containers >

ports

Container

h <<

view

ose and solve problems

s Control (IAM)

d access tokens

ge ACL

s policy

rties

Upload Add Directory Refresh Rename Delete Change tier Acquire lease

Authentication method: Access key (Switch to Azure AD User Account)

Location: exports / SSN

Search blobs by prefix (case-sensitive)

| Name | Modified | Access tier | Archive |
|--------------------------------------|----------|-------------|---------|
| <input type="checkbox"/> [..] | | | |
| <input type="checkbox"/> LOTA_MAX_MV | | | |
| <input type="checkbox"/> PORTSDATA | | | |

In above picture, you can find the storage container and paths to the exported csv files. Currently, whenever an external party fetches the csv file from the datalake, it will also delete this file. Each week, a new csv file will be exported to these directories.

European Maritime Safety Agency

Praça Europa 4
1249-206 Lisbon, Portugal
Tel +351 21 1209 200
Fax +351 21 1209 210
emsa.europa.eu

