Enabling procurement data value chains for economic development, demand management, competitive markets and vendor intelligence

Deliverable D5.2
Procurement APIs and platform release v1

<table>
<thead>
<tr>
<th>Date</th>
<th>01/06/2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Francisco Yedro, Oscar Corcho (UPM)</td>
</tr>
<tr>
<td>Dissemination level</td>
<td>PU</td>
</tr>
<tr>
<td>Work package</td>
<td>5</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Document metadata

Quality assurors and contributors

<table>
<thead>
<tr>
<th>Quality assuror(s)</th>
<th>George Konstantinidis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contributor(s)</td>
<td>Francisco Yedro, Oscar Corcho</td>
</tr>
</tbody>
</table>

Version history

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>17/04/2018</td>
<td>0.1</td>
<td>Initial ToC</td>
</tr>
<tr>
<td>10/05/2018</td>
<td>0.2</td>
<td>Initial version of the API catalogue after discussions at the plenary meeting</td>
</tr>
<tr>
<td>15/05/2018</td>
<td>0.3</td>
<td>Architecture diagram added</td>
</tr>
<tr>
<td>23/05/2018</td>
<td>0.4</td>
<td>Initial versions of the three APIs</td>
</tr>
<tr>
<td>01/06/2018</td>
<td>0.5</td>
<td>Comments added on the current version of APIs and architecture diagram, and deliverable ready for external QA</td>
</tr>
<tr>
<td>30/06/2018</td>
<td>1.0</td>
<td>QA comments received and added to the document</td>
</tr>
</tbody>
</table>
Executive summary

This document presents Deliverable D5.2 “Procurement APIs and platform release v1” of the TheyBuyForYou project. This deliverable is developed as part of Work Package 5 (WP5) “Standards, best practices, and integration”, task T5.2 “Design principles for procurement APIs”.

This document provides an initial version of the architectural diagram for the TheyBuyForYou platform and APIs, together with a catalogue of the current offering of APIs that are provided by the TheyBuyForYou partners.

The TheyBuyForYou architecture has been developed following state-of-the-art principles in software development, considering a low decoupling among all the software components, the use of REST principles for data provisioning, and the consideration of a distributed data governance for all the data items that are relevant to procurement APIs.

More specifically, the APIs that are currently described are those that allow dealing with:

- Tenders and their related entities, according to the Open Contracting Data Standard (OCDS), currently supported by OpenOpps.
- Organisations, currently supported by OpenCorporates.
- Tenders and their related entities, according to the Public Procurement Ontology (PPROC), supported by the city council of Zaragoza.
- Invoices, supported by the city council of Zaragoza.

This is to be considered the first step towards developing a more comprehensive procurement platform and API catalogue, which will be the subject of the upcoming deliverables from WP5, and which will integrate all the resources that are relevant to provide a full-fledged procurement data platform that can support the business cases developed in the context of WP6, and which will be fully aligned with the TheyBuyForYou public procurement ontology network.
Table of Contents

| DOCUMENT METADATA | 2 |
| EXECUTIVE SUMMARY | 3 |
| 1 INTRODUCTION | 6 |
| 2 THEYBUYFORYOU ARCHITECTURE | 6 |
| 3 CURRENT API CATALOGUE | 9 |

| 3.1 OPENCORPATES API | 9 |
| 3.1.1 General description | 9 |
| 3.1.2 Data model | 12 |
| 3.1.3 Method calls | 13 |
| 3.1.3.1 companies/jurisdiction_code/company_number | 13 |
| 3.1.3.2 companies/search | 17 |
| 3.1.3.3 companies/jurisdiction_code/company_number/filings | 21 |
| 3.1.3.4 companies/jurisdiction_code/company_number/network | 23 |
| 3.1.3.5 companies/jurisdiction_code/company_number/statements | 25 |
| 3.1.3.6 companies/jurisdiction_code/company_number/data | 25 |
| 3.1.3.7 officers/search | 25 |
| 3.1.3.8 officers/id | 26 |
| 3.1.3.9 corporate_groupings/name | 26 |
| 3.1.3.10 corporate_groupings/search | 27 |
| 3.1.3.11 filings/id | 27 |
| 3.1.3.12 data/id | 27 |
| 3.1.3.13 statements/gazette_notices/search | 28 |
| 3.1.3.14 statements/control_statements/search | 28 |
| 3.1.3.15 statements/trademark_registrations/search | 29 |
| 3.1.3.16 statements/id | 29 |
| 3.1.3.17 placeholders/id | 30 |
| 3.1.3.18 placeholders/id/network | 30 |
| 3.1.3.19 placeholders/id/statements | 31 |
| 3.1.3.20 jurisdictions | 32 |
| 3.1.3.21 jurisdictions/match | 32 |
| 3.1.3.22 industry_codes | 32 |
| 3.1.3.23 industry_codes/code_scheme_id | 32 |
| 3.1.3.24 industry_codes/code_scheme_id/code | 33 |
| 3.1.4 API accounts, authentication and authorisation | 33 |

| 3.2 OPEN REFINE RECONCILIATION API | 34 |
| 3.2.1 General description | 34 |
| 3.2.2 Endpoint | 34 |
| 3.2.3 API accounts, authentication and authorisation | 34 |

| 3.3 OPENOPPS API | 35 |
| 3.3.1 General description | 35 |
| 3.3.2 Data model | 37 |
| 3.3.3 Method calls | 39 |
| 3.3.3.1 tbly/ocds | 39 |
| 3.3.3.2 tenders/ocid | 41 |
| 3.3.4 API accounts, authentication and authorisation | 41 |

| 3.4 ZARAGOZA API AND SPARQL-BASED ACCESS | 41 |
| 3.4.1 General Description | 41 |
| 3.4.2 Data model | 48 |
3.4.3 Method calls and predefined SPARQL queries..............................................................49
3.4.3.1 SPARQL queries for public contracting data .................................................................49
3.4.3.2 Invoices search: servicio/factura.................................................................................50
3.4.3.3 Invoices search: servicio/factura/id ............................................................................50
3.4.4 API accounts, authentication and authorisation..............................................................51
4 CONCLUSIONS AND FUTURE WORK .....................................................................................52
5 ANNEX I. THEYBUYFORYOU API CHEATSHEET FOR DEVELOPERS.................................53
1 Introduction

This document provides an initial version of the architectural diagram for the TheyBuyForYou platform and APIs, together with a catalogue of the current offering of APIs that are provided by the TheyBuyForYou partners.

This is to be considered the first step towards developing a more comprehensive procurement platform and API catalogue, which will be the subject of the upcoming deliverables from WP5, and which will integrate all the resources that are relevant to provide a full-fledged procurement data platform that can support the business cases developed in the context of WP6, and which is fully aligned with the TheyBuyForYou public procurement ontology network.

The document is structured as follows. In section 2, we provide a description of the current high-level architecture of the TheyBuyForYou platform, which allows understanding how data is governed in the context of the project and how it will be made accessible for business cases and external third party developers. As part of the agile software development methodology that is being applied in the context of the project, as described in deliverable D5.1, the architecture may be evolving, so as to provide an up-to-date overview of the API offering. Such up-to-date version will be always available in the main GitHub repository of the project: http://github.com/tbfy/general. Section 3 describes the current catalogue of APIs and access mechanisms to data sources that are currently being offered by several of the TheyBuyForYou partners, and which will inform the project API design and development, together with the new functionalities that will be identified in the upcoming months. This API catalogue will especially evolve as soon as the TheyBuyForYou procurement ontology network is selected and stable. Section 4 discusses on the authentication and authorisation mechanisms that are currently being applied for the existing APIs, which may evolve in the course of the project. Section 5 discusses on the next steps towards the development of the TheyBuyForYou API, what will guide the development in the next phases of the project. Finally, Annex I provides a cheat sheet that can be used by developers to have a quick overview of the current API offering. An always up-to-date documentation of APIs will be offered at the project GitHub repository http://github.com/tbfy/knowledge-graph.

2 TheyBuyForYou Architecture

The TheyBuyForYou architecture has been developed following state-of-the-art principles in software development, considering a low decoupling among all the software components, the use of REST principles for data provisioning, and the consideration of a distributed data governance for the data items that are relevant to procurement APIs. Our API catalogue is mostly focused on providing access mechanisms to any software developers (in-house or external) that want to make use of the knowledge graph. Therefore, they are mostly focused on providing access to such data through the HTTP GET verb and the API catalogue is organised around the main entities that are relevant for public procurement (and in more general terms, for the handling of the economic information available in public administrations): tenders, agents (companies, organisations, individuals), invoices, etc.

The final characterisation of the specific resources that will need to be handled in our knowledge graph (and hence in our API) will be determined by the choice of public procurement ontologies that will be done in the course of the project. Currently, there are several ontologies and data formats that are being used for this purpose, as described in D5.1 (OCDS¹, PPROC², the upcoming EU eProcurement Ontology³, euBusinessGraph

---

¹ http://standard.open-contracting.org/latest/en/
² http://contsem.unizar.es/def/sector-publico/pproc
³ https://github.com/eprocurementontology/eprocurementontology/
ontology\(^4\), etc.), from which decisions will be taken in the course of the project for the selection of the final ontology network that will be used in the project and consequently in the development of our API.

Figure 1 provides a general high-level overview of the current TheyBuyForYou architecture. On the left-hand side we include the extraction, transformation and load processes that will be done to incorporate data sources into the knowledge graph, mostly via the databases of OpenCorporates (for agents – organisations, companies, individuals -) and OpenOpps (for OCDS-related data). The task of incorporating such data sources has been already done as part of the normal practices of these two organisations, and the work from WP1 and WP2 will improve the coverage and quality of those data sources.

The transformation into the RDF that will be stored in the knowledge graph infrastructure will be done using the DataGraft platform, as well as other ad-hoc scripts when required. Transformations will not necessarily be on the whole set of data available from these APIs, but may just provide some partial views on the data, depending on the needs from the business cases and the characteristics of compulsory and optional attributes identified in the TheyBuyForYou ontology network.

On the right-hand side we provide an overview of the main data storage mechanisms, including a triple store for the RDF-based data and a document store for the documents associated to public procurement (tender notices, award notices, etc.). Access to the pre-existing APIs of OpenCorporates and OpenOpps will be also made available as part of the knowledge graph infrastructure (in this case, the underlying infrastructure to serve data will be considered as a black box from the point of view of the TheyBuyForYou project). The knowledge graph will include owl:sameAs links to the corresponding URIs, so as to facilitate navigation across these different data sources. Documents will be connected to the main entities of the knowledge graph using rdfs:seeAlso or another relevant property, what will be decided when the ontology network is fixed.

The knowledge graph will be accessible via a Linked-Data enabled REST API. This means that the URIs that will be used to identify contracts, tenders, agents, invoices, etc. (that is, the main types of resources handled by the knowledge graph) will be de-referenceable and with content negotiation (at least providing RDF, JSON, JSON-LD, CSV and HTML views of the data). There will be also SPARQL endpoint access, for those developers willing to make ad-hoc queries to the knowledge graph, as well as a range of additional services to enable search over the knowledge graph and document store, and possibly reconciliation services to facilitate third parties the usage of TheyBuyForYou URIs (with the aim of increasing uptake). All these APIs will be available under an authentication/authorisation layer.

Finally, the range of online tools and business case implementations from the project (and also from any other outside contributions) will be based on this data access layer.

In the following section we will focus only on those APIs that are currently available from some of the consortium partners (OpenCorporates, OpenOpps, Zaragoza), since these APIs will inform the future development required in the project, as well as serve to identify implementation gaps in the proposed architecture.

---

Figure 1. TheyBuyForYou High-level Architecture
3 Current API Catalogue

In this section of the deliverable, as discussed, we are presenting the initial APIs developed by project partners, based on their in-house pre-existing APIs, to offer support to some of the envisage functionalities. This will serve as a starting point to build a common project API, which will be presented in the following versions/updates of this deliverable.

More specifically, the APIs that are described in this section are those that allow dealing with:

- Organisations, currently supported by OpenCorporates.
- Tenders and their related entities, according to the Open Contracting Data Standard (OCDS), currently supported by OpenOpps.
- Tenders and their related entities, according to the Public Procurement Ontology (PPROC), supported by the city council of Zaragoza.
- Invoices, supported by the city council of Zaragoza.

3.1 OpenCorporates API

3.1.1 General description

OpenCorporates is a database with information about approximately 100 million companies worldwide, obtained from a wide range of data sources, and which can be accessed via an API\(^5\). Data are available either as share-alike attribution open data or commercially.

The contents of this section are based on the current online documentation available at the OpenCorporates API. An always up-to-date version is available there. The objective of this section is to provide a general overview of the main data models used by the API as well as the main types of resources and calls that are made available by it.

Among the main data offered by the API, we can find the legal name of the company, the identifier given to the company by the company register, the date the company was incorporated, the previous or alternative names of a company, registered address and so on. More details are provided in section 3.1.2. As an example, next we show the response to https://api.opencorporates.com/companies/nl/17087985:

```
"api_version": "0.2",
"results": {
  "company": {
    "name": "Bover B.V.",
    "company_number": "17887985",
    "jurisdiction_code": "nl",
    "incorporation_date": null,
    "dissolution_date": null,
    "company_type": "Besloten Vennootschap",
    "registry_url": "https://server.db.kvk.nl/TST-BIN/FU/TSWS001?BUTT=17087985",
    "branch": null,
    "branch_status": null,
    "inactive": false,
    "current_status": "Active",
    "created_at": "2011-01-12T21:57:57+00:00",
    "updated_at": "2012-02-02T18:36:46+00:00",
    "retrieved_at": "2011-08-25T14:37:37+01:00",
    "opencorporates_url": "https://opencorporates.com/companies/nl/17087985",
```

\(^5\) https://api.opencorporates.com/documentation/API-Reference
The endpoint for the API is api.opencorporates.com. The API currently supports both http and https, but is likely to move to https only in the near future. Besides, the API can be used without API keys, but it is restricted to certain usage limits. Finally, the API by default returns JSON, although XML is also supported (by adding format=xml as a query parameter).

We will now focus on some of the main characteristics of the API:

**Identifiers.** Resources are identified with an OpenCorporates URL (which provides access to the HTML version). For example, https://opencorporates.com/companies/gb/00102498. In order to access this data in the OpenCorporates API, the following URL should be used: https://api.opencorporates.com/companies/gb/00102498. This does not follow Linked Data principles, as we will be focusing on in the context of the TheyBuyForYou API, although this may be fixed in the future.

**Versioning.** The API uses a versioning system. If a version number is supplied, that version will be used, provided it is still supported. If no version number is supplied, the current version will be used.

The v0.1 form of the GET request to get information about a company is, for example: https://api.opencorporates.com/v0.1/companies/gb/00102498. The unversioned form is: https://api.opencorporates.com/companies/gb/00102498.

To get information about an existing version, the versions method call can be used. For example: https://api.opencorporates.com/v0.4/versions.
Pagination. Those calls returning a number of results (e.g. searches, or filings for a company) return a paginated response. By default 30 objects are returned, together with the current page number, and total number of pages. For users with an API key, the number per page can be increased (to up to 100) by supplying a `per_page` query parameter, and the page number is specified with the `page` query parameter (the `page` parameter is limited to 100 to ensure that the API is as fast as possible for all users).

Provenance. Provenance is attached to multiple objects, including statements and placeholders, but also the links between, for example, placeholders and companies, or between placeholders and statements. The provenance object consists of the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source_url</td>
<td>this is url from which the data was obtained, or the OpenCorporates url for the user, if it was contributed by a user</td>
</tr>
<tr>
<td>confidence</td>
<td>a confidence that the associated data object is correct (numeric value between 0 and 100). We will be publishing more about this value in the future</td>
</tr>
<tr>
<td>source_type</td>
<td>the type of source. Possible values are 'external' (i.e. data from a public source, such as a company register, or government website), 'internal' (i.e. data within the OpenCorporates system), or 'induction' (for example where data has been reconciled to a company by OpenCorporates)</td>
</tr>
<tr>
<td>actor_type</td>
<td>the type of the ‘actor’ that contributed the data. Possible values are 'bot' (an OpenCorporates Bot) or 'user' (in which case the source_url will contain a link to the user page on OpenCorporates)</td>
</tr>
<tr>
<td>log_message</td>
<td>a note on the source, particularly if the data was added by a user (e.g. a corporate network relationship)</td>
</tr>
<tr>
<td>created_at</td>
<td>the date the provenance was created (and thus the date we retrieved the data or made the match). Note we often regularly check statements are true, and therefore a statement may have many provenances, which act as timestamps for the statement</td>
</tr>
</tbody>
</table>

Source. Detailed responses include the provenance of the data as a source object, consisting of the following:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>retrieved_at</td>
<td>a timestamp of when the information was retrieved from the source, or the data was created.</td>
</tr>
<tr>
<td>source_type</td>
<td>this is either 'user' if the information was contributed by a user, or 'external' if the information was obtained from an external source.</td>
</tr>
<tr>
<td>url</td>
<td>where the information was retrieved from. This may be the URL of a specific datum, or the download URL of a data_dump, or if this is not available the URL of the publisher. If the information was user-contributed it will be the OpenCorporates url of the user.</td>
</tr>
<tr>
<td>publisher</td>
<td>If the information came from an external source, this is the original publisher of the information; if the information was user-contributed, this will be the user's name.</td>
</tr>
<tr>
<td>terms</td>
<td>if we are aware that the external publisher of the data has applied an explicit licence to the data (e.g. in the UK it is often Crown Copyright, or sometimes the Open Government Licence). A Null value for this does <em>not</em> imply there is no underlying licence.</td>
</tr>
</tbody>
</table>
Filters. Date and term filters can be specified. Date filters can be supplied either as a specific date or as a date range. If a date range is given it should be in the form 2009-08-22:2012-01-08. Either the start date or the end date may be omitted to make the date range unbounded at the start or end, e.g.:2012-01-08 would represent any date before 8 Jan 2012. Term filters can be supplied as a single term, a comma separated list (the record must match all terms; e.g. foo,bar,baz), or a pipe separated list (the record need only match one term; e.g. foo|bar|baz).

Search results. All method calls that return search results are case insensitive, and return results in a predefined order (alphabetical, temporal, etc., depending on the type of objects returned). All these results can be ordered differently to this default ordering, by using the ranking provided by the ElasticSearch search index. This can be done by passing the order=score query parameter to those calls. Additionally, results can be restricted by a number of ‘facets’.

3.1.2 Data model

In this section we provide a brief description of the main elements in the data model of OpenCorporates, which is graphically described in

Figure 2.
3.1.3 Method calls

In this subsection we provide details of the methods implemented in the OpenCorporates API, related to the previous data model. A complete summary of these methods is provided in Annex I. All of the methods use the HTTP verb GET.

3.1.3.1 companies/:jurisdiction_code/:company_number

This returns the core data for the given company. The jurisdiction code is the code for the jurisdiction which registered the company. If this is a country it is simply the two-letter ISO code for that country, e.g. Spain = es, United Kingdom = gb. If this is a state or province it is an underscore version of the ISO 3166-2 code for the jurisdiction, eg. Michigan in the US is us_mi.

Included in the response will be the core data on the company (incorporation date, company type, registered address, etc), a summary of other data held on the company (e.g. trademarks, payments from government, other addresses), including the most recently added data, and the most recent statutory filings, and parent company, if we have this information (see the company_network method call for more detailed hierarchy data). If you don’t need the filings and the other data, or believe there are not likely to be such results, you can pass sparse=true as a query parameter, and this will be both quicker, and potentially a considerably smaller response:
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>the legal name of the company</td>
</tr>
<tr>
<td>company_number</td>
<td>the identifier given to the company by the company register</td>
</tr>
<tr>
<td>jurisdiction_code</td>
<td>the code for the jurisdiction in which the company is incorporated</td>
</tr>
<tr>
<td>company_type</td>
<td>the type of company (e.g. LLC, Private Limited Company, GBMH)</td>
</tr>
<tr>
<td>current_status</td>
<td>the given description of the filing</td>
</tr>
<tr>
<td>incorporation_date</td>
<td>the date the company was incorporated</td>
</tr>
<tr>
<td>dissolution_date</td>
<td>the date the company was dissolved</td>
</tr>
<tr>
<td>inactive</td>
<td>a flag indicating if the company is 'inactive'. This is an OpenCorporates mapping from a number of different inactive company statuses, including dissolved, removed, liquidated, etc. Note that not all companies make company statuses available, so that it cannot be inferred that a company is active just because inactive is not true</td>
</tr>
<tr>
<td>opencorporates_url</td>
<td>the url of the company on OpenCorporates</td>
</tr>
<tr>
<td>previous_names</td>
<td>an array of previous name objects. Each previous name object has a company_name attribute and type attribute (e.g. 'trading', 'legal') and the following optional attributes: start_date, end_date, language as ISO-639 code (for example if it is an alternative legal name in another language)</td>
</tr>
<tr>
<td>alternative_names</td>
<td>an array of alternative name objects. Each alternative name object will has a company_name attribute and an optional con_date attribute, which is the date the company's name changed from the previous name (and thus can be considered to be an end_date)</td>
</tr>
<tr>
<td>registered_address_in_full</td>
<td>the registered address of the company, as a single string</td>
</tr>
<tr>
<td>registered_address</td>
<td>the registered address of the company, as a structured object</td>
</tr>
<tr>
<td>industry_codes</td>
<td>Returns an array of industry_codes representing the industries the company operates in (see below for more on industry_codes). This is part of our new industry code functionality and the code objects returned are different from the previous representation</td>
</tr>
<tr>
<td>identifiers</td>
<td>Returns an array of identifier objects, including the identifier system name and code, and the identifier uid. For example for Canadian business numbers the uid would be the business number (e.g. &quot;800322042RC0001&quot;) and &quot;ca_bn&quot; for the identifier_system_code and &quot;Canadian Business Number&quot; for the identifier_system_name</td>
</tr>
<tr>
<td>registry_url</td>
<td>the url of the companies page in the company register. Note, not all company registers provide persistent urls for the companies in the register</td>
</tr>
<tr>
<td><strong>controlling_entity</strong></td>
<td>This is the controlling parent of the company in question, and is derived either from explicit subsidiary or parent data, or from a company having a majority equity stake in the company</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ultimate_beneficial_owners</strong></td>
<td>These are the ultimate beneficial owners of the company in question, if we know them</td>
</tr>
<tr>
<td><strong>branch</strong></td>
<td>A flag to indicate if a company is a 'branch'. If the flag is 'F' it indicates this the entry in the company register relates to an an out-of-jurisdiction company (sometimes called a 'foreign corporation' in the US). If it is 'L' it is a local office of a company (few company registers collect this information). If it is null value, it means either the company is not a branch, or the company register does not make the information available</td>
</tr>
<tr>
<td><strong>branch_status</strong></td>
<td>A descriptive text version of the 'branch' flag</td>
</tr>
<tr>
<td><strong>home_company</strong></td>
<td>When a company is a branch of an out-of-jurisdiction company, this is the 'home' company (if we know it), including the name, jurisdiction_code, company_number and opencorporates_url</td>
</tr>
</tbody>
</table>

Example:

```json
https://api.opencorporates.com/v0.4/companies/gb/00102498
https://api.opencorporates.com/v0.4/companies/gb/00102498?sparse=true
```

```json
{
  "api_version": "0.4",
  "results": {
    "company": {
      "branch_status": null,
      "company_number": "00102498",
      "company_type": "Public Limited Company",
      "corporate_groupings": [
        {
          "corporate_grouping": {
            "name": "bp",
            "opencorporates_url": "https://opencorporates.com/corporate_groupings/bp",
            "updated_at": "2014-02-16T11:22:24+00:00",
            "wikipedia_id": "BP"
          }
        }
      ],
      "created_at": "2010-10-21T18:20:50+01:00",
      "current_status": "Active",
      "data": {
        "most_recent": [
          {
            "datum": {
```
"data_type": "CompanyAddress",
"description": "1 St James's Square, London SW1Y 4PQ, GB",
"id": 9788778,
"opencorporates_url": "https://opencorporates.com/data/9788778",
"title": "Company Address"
},
{
"datum": {
"data_type": "Website",
"description": "http://www.bp.com/sectiongenericarticle.do?categoryId=9021231&contentId=7039279",
"id": 8474113,
"opencorporates_url": "https://opencorporates.com/data/8474113",
"title": "Website"
}
},
{
"datum": {
"data_type": "OfficialRegisterEntry",
"description": "register id: 313807",
"id": 2452824,
"opencorporates_url": "https://opencorporates.com/data/2452824",
"title": "SEC Edgar entry"
}
}],
"total_count": 125,
"url": "https://opencorporates.com/companies/gb/00102498/data",
"dissolution_date": null,
"filings": [
{
"filing": {
"date": "2014-02-13",
"id": 199825350,
"opencorporates_url": "https://opencorporates.com/filings/199825350",
"title": "Return of purchase of own shares",
"uid": "284acfe3j2Rd2YeX140hPyU905S1bNhYQj2MFcdxFgMDA8dJQv9/Cw8TUXmMzAyCuzvi0/KlcksqcVCd1mAPA2MnLtbd4IOTExIdxRguYMk10FkOIwNDEwMjQ1MnNef831LEvM2UY0/88uJ5K6s1Xx/3cCsrz9zE9FQo5Z1aSaqybgYGBkYgZq1ZI8rMwJWIGYDYNyYg5gB11dAm1uANC4NJA="
}
},
{
"filing": {
"date": "2014-02-13",
"id": 199825349,
"opencorporates_url": "https://opencorporates.com/filings/199825349",
"title": "Notice of cancellation of shares",
"uid": "771a5ae3j2Rd2YeX140hPyU905S1bNhYQj2MFcdxFgMDA8dJQv9/1w8XUxMzAyCuzvi0/KlcksqcVCd1mAPA2MnLtbd4IOTExIdxRkuYMk10FkOIwNDEwMjQ1MnNef831LEvM2UY0/88uJ5K6s1Xx/3cCsrz9zE9FQo5Z1aSaqybgYGBkYgZq1ZI8rMwJWIGYDYNyYg5gB11dAm1uAOT2NK0="
}
]}
"inactive": false,
"incorporation_date": "1909-04-14",
"industry_codes": [
{
  "industry_code": {
    "code": "70100",
    "description": "Activities of head offices",
    "code_scheme_id": "uk_sic_2007",
    "code_scheme_name": "UK SIC Classification 2007"
  }
}
]
,"jurisdiction_code": "gb",
"name": "BP P.L.C.",
"officers": [
  {
    "officer": {
      "end_date": null,
      "id": 32609673,
      "name": "DAVID JOHN JACKSON",
      "opencorporates_url": "https://opencorporates.com/officers/32609673",
      "position": "secretary",
      "start_date": "2003-07-24",
      "uid": null
    }
  },
  {
    "officer": {
      "end_date": "2012-03-30",
      "id": 32609674,
      "name": "DAVID JOHN PEARL",
      "opencorporates_url": "https://opencorporates.com/officers/32609674",
      "position": "secretary",
      "start_date": "2001-11-01",
      "uid": null
    }
  },
  {
    "officer": {
      "end_date": null,
      "id": 32609675,
      "name": "PAUL MILTON ANDERSON",
      "opencorporates_url": "https://opencorporates.com/officers/32609675",
      "position": "director",
      "start_date": "2010-02-01",
      "uid": null
    }
  },
  {
    "officer": {
      "end_date": null,
      "id": 32609676,
      "name": "FRANK BOWMAN",
      "opencorporates_url": "https://opencorporates.com/officers/32609676",
      "position": "director",
      "start_date": "2010-11-08",
      "uid": null
    }
  }
],
"opencorporates_url": "https://opencorporates.com/companies/gb/00102498",
"previous_names": [}
3.1.3.2 companies/search

This returns a collection of companies whose name matches the given search term (submitted as :q in the query parameters). It's important to note that the search is deliberately quite loose, requiring the returned companies to have all the searched-for words (in any order), but allowing other words to be present too (so 'Bank Barclays' is the same as 'Barclays Bank'). You can also search for company names beginning with the given search term by ending the term with a '*', so that 'Barclays Bank*' will match 'Barclays Bank PLC' but not 'Barclays UK Bank' or 'Barclays Big Bank'.

As discussed in the general characteristics of the API, search is case-insensitive. In this case, it returns companies with previous names matching the term as well as current name, and some normalisation of the company names is done, removing non-text characters (e.g. dashes, parentheses, commas), common 'stop words' (e.g. 'the', 'of'), and normalising common company types (e.g. Corp, Inc, Ltd, PLC) so that both the short and long versions can be used. Other modifiers applicable to searches (changing order using ElasticSearch, facets, etc., are applicable to this call):

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jurisdiction_code Term filter</td>
<td>The companies searched for can be restricted to a given jurisdiction by passing a</td>
</tr>
<tr>
<td>country_code Term filter</td>
<td>country_code query parameter. Note that in most cases the country and jurisdiction are</td>
</tr>
<tr>
<td>company_type Term filter</td>
<td>The type of the company, as defined by the company register.</td>
</tr>
<tr>
<td>current_status Term filter</td>
<td>The current status of the company, as defined by the company register.</td>
</tr>
<tr>
<td>industry_codes Term filter</td>
<td>One or more industry codes representing the industries the company operates in. The</td>
</tr>
<tr>
<td></td>
<td>codes can be those in the company’s jurisdiction’s native code scheme, or any mapped</td>
</tr>
</tbody>
</table>
schemes, or any parent of those codes (when the schemes are hierarchical). This is part of our new industry code functionality and the values submitted should be the industry_code uid, consisting of the industry_code :code_scheme_id and :code joined with a hyphen (see below for more on industry_codes).

<table>
<thead>
<tr>
<th>Term filter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>registered_address</td>
<td>The companies searched for can be restricted by passing in an address or parts of an address. This performs a similar search to the basic companies search, so searching for ‘52 London’ will search for address with 52 and London in them, e.g. ‘52 Acacia Ave, London, NW1’, or ‘52 London Road, St Albans’</td>
</tr>
<tr>
<td>created_since</td>
<td>Companies added to OpenCorporates after the given date.</td>
</tr>
<tr>
<td>incorporation_date</td>
<td>Companies incorporated on the given date</td>
</tr>
<tr>
<td>dissolution_date</td>
<td>Companies dissolved on the given date</td>
</tr>
<tr>
<td>incorporated_before</td>
<td>Restrict to companies with an incorporation_date before the given date. Deprecated in favour of new incorporation_date functionality.</td>
</tr>
<tr>
<td>incorporated_since</td>
<td>Restrict to companies with an incorporation_date after the given date. Deprecated in favour of new incorporation_date functionality.</td>
</tr>
<tr>
<td>dissolved_before</td>
<td>Restrict to companies with a dissolution_date before the given date. Deprecated in favour of new dissolution_date functionality.</td>
</tr>
<tr>
<td>dissolved_since</td>
<td>Restrict to companies with a dissolution_date after the given date. Deprecated in favour of new dissolution_date functionality.</td>
</tr>
<tr>
<td>exclude_inactive</td>
<td>Restrict to companies that are active/inactive. Deprecated in favour of new inactive functionality.</td>
</tr>
<tr>
<td>inactive</td>
<td>Filter by inactive status (boolean). This replaces the exclude_inactive filter from previous versions. If true is supplied it will restrict to inactive companies. If false is supplied it will exclude inactive companies. If no value is supplied it will not filter by inactive status.</td>
</tr>
<tr>
<td>branch</td>
<td>Filter by branch status (boolean). This replaces the exclude_branches filter from previous versions. If true is supplied it will restrict to branch companies. If false is supplied it will exclude branch companies. If no value is supplied it will not filter by branch status.</td>
</tr>
<tr>
<td>nonprofit</td>
<td>Filter by nonprofit company type (boolean). If true is supplied it will restrict to entities that we have identified as being of nonprofit company type. If false is supplied it will exclude such companies. If no value is supplied it will not filter by nonprofit company type.</td>
</tr>
<tr>
<td>identifier_uids</td>
<td>The companies searched for can be restricted to those with the given third-party identifier, such as business or charity numbers, tax identifiers, etc (only for users with API key).</td>
</tr>
</tbody>
</table>
**fields**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>By default when searching with a term (e.g. 'Barclays Bank') it mirrors the search on the main OpenCorporates website, searching name, normalised name, company number, previous names etc. By using the fields parameter, you can limit to a given field or fields. Options are name, normalised_name, company_number, heavily_normalised_number (company number excluding leading zeroes, for example), previous_names, postal_codes, trademark_registration_mark_texts.</td>
</tr>
</tbody>
</table>

**normalise_company_name**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalise company name before searching for it (boolean). Unlike the main OpenCorporates website, the API does not normalise the company name before searching for it (so that restrictions work effectively). If true is supplied, this will normalise the submitted company name before searching (so that Ltd will be normalised to Limited, Corp to Corporation etc).</td>
</tr>
</tbody>
</table>

**types_of_data_held**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter results by companies that have the the supplied types of related data. For valid options see the list visible when using the search feature on opencorporates.com.</td>
</tr>
</tbody>
</table>

Additional sorting options (from v0.4): Users with an API key can also sort by 4 date fields (all in reverse order, i.e. newest first):

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>created_at</td>
<td>a timestamp of when the company was added to OpenCorporates.</td>
</tr>
<tr>
<td>updated_at</td>
<td>the datetime the company was updated in any way. The company record will be considered to be updated in many circumstances, including when any associated data is changed.</td>
</tr>
<tr>
<td>incorporation_date</td>
<td>the date of incorporation as given by the company register. Please note that not all company registries record and publish this date.</td>
</tr>
<tr>
<td>dissolution_date</td>
<td>the date of dissolution as given by the company register. Please note that not all company registries record and publish this date.</td>
</tr>
</tbody>
</table>

Example:

```
https://api.opencorporates.com/v0.4/companies/search?q=barclays+bank
https://api.opencorporates.com/v0.4/companies/search?q=barclays&per_page=50&page=2
https://api.opencorporates.com/v0.4/companies/search?q=barclays+bank&jurisdiction_code=gb
https://api.opencorporates.com/v0.4/companies/search?q=barclays+bank&jurisdiction_code=gb&current_status=Active
https://api.opencorporates.com/v0.4/companies/search?q=bank+of+scotland
https://api.opencorporates.com/v0.4/companies/search?q=bank+of+scotland&order=score
https://api.opencorporates.com/v0.4/companies/search?q=bank+of+scotland&inactive=false
https://api.opencorporates.com/v0.4/companies/search?q=bank+of+scotland&order=incorporation_date
https://api.opencorporates.com/v0.4/companies/search?q=bank+of+scotland&order=incorporation_date&incorporation\date=2013-12-03:2014-05-22
https://api.opencorporates.com/v0.4/companies/search?q=bank+of+scotland&fields=name,previous_names
https://api.opencorporates.com/v0.4.1/companies/search?industry_codes=eu_nace_2-869,eu_nace_2-691
```

```json
{
    "api_version": "0.4",
    "results": {
        "companies": [
```
| Company | Branch Status | Company Number | Company Type | Created At | Current Status | Dissolution Date | Inactive | Incorporation Date | Jurisdiction Code | Name                        | Opencorporates URL | Previous Names | Registry URL | Retrieved At | Source | Updated At | Registered Address In Full |
|---------|--------------|----------------|--------------|------------|---------------|-----------------|-----------|--------------------|-----------------|----------------|--------------------------|-------------------|---------------|-------------|-------------|--------|------------|-------------------------|


3.1.3.3 companies/:jurisdiction_code/:company_number/filings

This returns the statutory filings for the given company. Each filing will be returned with the date of filing, the OpenCorporates id and url for the filing, and one or many of the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uid</td>
<td>the id given to the filing by the company registry</td>
</tr>
<tr>
<td>title</td>
<td>this is either the given title of the filing, or the filing_type, or a generic field containing a human-friendly version of the date. This means that every filing can be relied upon to have a filing_title</td>
</tr>
<tr>
<td>description</td>
<td>the given description of the filing</td>
</tr>
<tr>
<td>url</td>
<td>the url of the filing. This may be a url of an HTML page, or of the document itself, for those more enlightened jurisdictions which make this available without charge</td>
</tr>
<tr>
<td>filing_type_name</td>
<td>a human-readable name of the type of filing. This might have been computed from the filing_code</td>
</tr>
<tr>
<td>filing_code</td>
<td>the code given by the company registry to this type of filing</td>
</tr>
</tbody>
</table>

Example:

https://api.opencorporates.com/v0.4/companies/gb/00102498/filings
https://api.opencorporates.com/v0.4/companies/gb/00102498/filings?page=2

"api_version": "0.4",
"results": {
"filings": [
{
    "filing": {
        "date": "2014-02-13",
        "description": "RETURN OF PURCHASE OF OWN SHARES",
        "filing_code": "SMH3",
        "filing_type": "Return of purchase of own shares",
        "id": "199825350",
        "opencorporates_url": "https://opencorporates.com/filings/199825350",
        "title": "Return of purchase of own shares",
        "uid": "284acfe3xjZ8d2YnXl4oMfPy19S5SIOBNHrYQJzMFcdxFgMDA8djQy9/Cw8BTuxMzAycuzvi8/KlcksqVCdx1mAPA2Mnlttb41OTE1idxRguVMuK1QFkDIWzDWfMjQ+MnNef83ILEVzUYo/8U0jUK6sIXx/3cc5r9zE9FQo5Z1a5nYygG8FkYg7zg1ZlBlAwJWIGYDYhYg5gBiT1DmAmjUwN4NJa="
    }
}
]
}
3.1.3.4 companies/jurisdiction_code/:company_number/network

This returns the immediate ‘computed corporate network’ for the given company as a set of control relationships (i.e. one company is thought to control or influence another company). This is the same data you can see on a company's network page on the main OpenCorporates site.

A control relationship is a relationship of control from the subject company to another company (a ‘child’ relationship), or from another company to the subject company (a ‘parent’ relationship). These relationships are computed automatically from a number of relationship statements associated with each company, and the network will change as more statements are added, because for example new relationships are discovered, or statements are added which say that a previous statement is no longer true, or even that our algorithms for calculating relationships are improved.

Each relationship is made of the following attributes:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>relationship_type</td>
<td>the type of relationship, e.g. subsidiary, shareholding. A subsidiary relationship typically comes from statutory filings (e.g. SEC 10-K reports); shareholding information typically comes from company registers, and is more granular</td>
</tr>
<tr>
<td>relationship_properties</td>
<td>any additional properties of the relationship, e.g. ownership percentage</td>
</tr>
<tr>
<td>confidence</td>
<td>a computed confidence in the relationship based on the underlying statements</td>
</tr>
<tr>
<td>parent_name</td>
<td>the name of the parent entity</td>
</tr>
<tr>
<td>parent_type</td>
<td>the type of entity the parent is</td>
</tr>
<tr>
<td>parent_opencorporates_url</td>
<td>the OpenCorporates URL of the parent entity</td>
</tr>
<tr>
<td>child_name</td>
<td>the name of the child entity</td>
</tr>
<tr>
<td>child_type</td>
<td>the type of entity the child is</td>
</tr>
</tbody>
</table>
The response is not paginated and can return no results (if we have no known relationships) or hundreds of relationships. It currently returns just a depth of 1 (i.e. just immediate parents and children). We are looking at adding greater levels of depth. Note the related entities may be companies, or placeholders. Placeholders are objects that we haven't yet matched to a company, either because we don't have the companies in that jurisdiction, or because we can't confidently match it to a company. To allow the network to be 'walked', you can also make network requests for placeholders, e.g. A Company controls B Placeholder controls C Company. Note that some networks are circular (A controls B, which controls C, which controls A), and you should allow for this when making repeated network calls so that you do not get stuck in an endless loop.

Filter options:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>confidence</td>
<td>a computed confidence in the relationship based on the underlying statements (between 0 and 100). Only relationships above this confidence will be returned (the default value here is 60). You can also pass in 'all' here to get all relationships, irrespective of the confidence in the data</td>
</tr>
<tr>
<td>dates</td>
<td>By default, the network data will be what we think to be true today (usually because the information on which it is based is recent enough to make that a reasonable assumption). However, if a date is passed in (in ISO 8601 format), it will be the network data we have that is believed to be current on that date. This is useful for data such as the US banks, for which we hold good data going back over 10 years.</td>
</tr>
<tr>
<td>ownership_percentage</td>
<td>By default the API will return every control relationship we know about where the percentage control is greater or equal to 2%. However, this can be overridden by passing a numeric value here, for example 50 or even 0 (for every relationship, regardless of percentage). Note that this will not affect the results for control relationships where we do not know the percentage (e.g. many subsidiary relationships)</td>
</tr>
</tbody>
</table>

Example:

https://api.opencorporates.com/v0.4/companies/gb/00102498/network
https://api.opencorporates.com/v0.4/companies/gb/00102498/network?confidence=80

3.1.3.5 companies/jurisdiction_code/:company_number/statements

This returns the statements associated with each company. A statement is a purported 'statement of fact' from a source (a public record or a user). For example, subsidiary statement may have been parsed from a filing at the US Securities And Exchange Commission, or a user may have made a statement that one company is a parent of another.

Example:

https://api.opencorporates.com/v0.4/companies/gb/00102498/statements
https://api.opencorporates.com/v0.4/companies/gb/00102498/statements?page=2

3.1.3.6 companies/jurisdiction_code/:company_number/data

[Note the 'data' type is being deprecated in favour of statements, which has a number of benefits, including improved provenance, and, shortly, improved searchability] This returns the data held for the given company. In
OpenCorporates a datum is a piece of information derived from a public record, or contributed by a user. Examples are a website, sales tax number, address, or entry in an official register (e.g. the UK Charity Register or US SEC register). Each datum will be returned with the OpenCorporates id, the OpenCorporates URL for the datum, timestamps indicating when they were created/updated and the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>the id given to the filing by the company registry</td>
</tr>
<tr>
<td>title</td>
<td>this is the title of the datum as displayed on OpenCorporates. It is computed internally depending on the data_type</td>
</tr>
<tr>
<td>description</td>
<td>the given description of the datum as displayed on OpenCorporates. It is computed internally depending on the data_type. Together with the title this makes it easy to display a useful summary of the datum</td>
</tr>
<tr>
<td>data_type</td>
<td>this is a string value denoting the type of data, e.g. 'CompanyAddress', 'OfficialRegisterEntry'</td>
</tr>
</tbody>
</table>

Example:

https://api.opencorporates.com/v0.4/companies/gb/00102498/data

3.1.3.7 officers/search

This returns a collection of officers whose name matches the given search term (submitted as :q in the query parameters). It's important to note that the search is deliberately quite loose, requiring the returned officers to have all the searched-for words (in any order), but allowing other words to be present too (so 'John Smith' is the same as 'Smith John').

As discussed in the general characteristics of the API, search is case-insensitive. Other modifiers applicable to searches (changing order using ElasticSearch, facets, etc., are applicable to this call):

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jurisdiction_code</td>
<td>Term filter. The officers searched for can be restricted to a given jurisdiction by passing a jurisdiction_code query parameter.</td>
</tr>
<tr>
<td>date_of_birth</td>
<td>Date filter. The companies searched for can be restricted by passing in an exact date of birth or date span (note we currently only have date of birth for UK company directors).</td>
</tr>
<tr>
<td>position</td>
<td>Term filter. The officers searched for can be restricted by passing a position (e.g. 'director', 'ceo', 'agent').</td>
</tr>
<tr>
<td>address</td>
<td>Term filter. The officers searched for can be restricted by passing in an address or parts of an address. This performs a similar search to the basic companies search, so searching for '52 London' will search for address with 52 and London in them, e.g. '52 Acacia Ave, London, NW1', or '52 London Road, St Albans'</td>
</tr>
<tr>
<td>inactive</td>
<td>Filter by inactive status (boolean). If true is supplied it will restrict to inactive officers (either where they are no longer directors of an active company, or where the company is inactive). If false is supplied it will exclude inactive officers. If no value is supplied it will not filter by inactive status.</td>
</tr>
</tbody>
</table>

Example:

https://api.opencorporates.com/v0.4/officers/search?q=john+smith
3.1.3.8 officers/:id

This returns information on a particular officer (a director or an agent for a company). Each officer will be returned with the name of the officer, the OpenCorporates id and url for the officer, the company it relates to, and one or more of the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>the position held (e.g. director, secretary, CEO)</td>
</tr>
<tr>
<td>uid</td>
<td>the id given to the officer by the company registry</td>
</tr>
<tr>
<td>start_date</td>
<td>this is date on which the officership started</td>
</tr>
<tr>
<td>end_date</td>
<td>this is date on which the officership ended</td>
</tr>
<tr>
<td>address</td>
<td>the given address of the officer, if known (only for users with API key)</td>
</tr>
<tr>
<td>date_of_birth</td>
<td>the date of birth of the officer, if known (only for users with API key)</td>
</tr>
</tbody>
</table>

3.1.3.9 corporate_groupings/:name

This returns information on a given CorporateGrouping. A CorporateGrouping is a user-curated collection of companies that belong to some human-understand concept of a corporation (which maps to the Wikipedia article about that corporation). See (https://blog.opencorporates.com/2011/06/01/introducing-corporategroupings-where-fuzzy-concepts-meet-legal-entities/). The name of the CorporateGrouping is case-insensitive. Included in the response will be core information for the CorporateGrouping (:name, :wikipedia_url, :companies_count), together an array of membership of the CorporateGrouping (each membership will contain a company and an OpenCorporates Source Object), and an array of the users that are curating this CorporateGrouping.

Example:

https://api.opencorporates.com/v0.4/corporate_groupings/capita

3.1.3.10 corporate_groupings/search

This returns a collection of corporate_groupings whose name matches the given search term (submitted as :q in the query parameters). The search is case-insensitive, and is a stem-search, that is it searches for corporate_groupings whose name begins with the given characters.

Example:

https://api.opencorporates.com/v0.4/corporate_groupings/search?q=bar

https://api.opencorporates.com/v0.4/corporate_groupings/search?q=ba&per_page=50&page=2

3.1.3.11 filings/:id

This returns information on a statutory filing for a company. Each filing will be returned with the date of filing, the OpenCorporates id and url for the filing, the company it relates to, and one or many of the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>uid</td>
<td>the id given to the filing by the company registry</td>
</tr>
<tr>
<td>title</td>
<td>this is either the given title of the filing, or the filing_type, or a generic field containing a human-friendly version of the date. This means that every filing can be relied upon to have a filing_title</td>
</tr>
<tr>
<td>description</td>
<td>the given description of the filing</td>
</tr>
<tr>
<td>url</td>
<td>the url of the filing. This may be a url of an HTML page, or of the document itself, for those more enlightened jurisdictions which make this available without charge</td>
</tr>
<tr>
<td>filing_type_name</td>
<td>a human-readable name of the type of filing. This might have been computed from the filing_code</td>
</tr>
<tr>
<td>filing_code</td>
<td>the code given by the company registry to this type of filing</td>
</tr>
</tbody>
</table>

3.1.3.12 data/id

This returns information on a given datum. In OpenCorporates a datum is a piece of information derived from a public record, or contributed by a user. Examples are a website, sales tax number, address, or entry in an official register (e.g. the UK Charity Register or US SEC register). Note that a datum can relate to more than one company (e.g. an official notice of a merger between two companies), and so an array of companies is returned, each including the company name, jurisdiction_code, company_number and OpenCorporates url.

The datum will also return the datatype, title, and description, as described in get companies/jurisdictioncode/:companynumber/data above. In addition the detailed attributes of the datum are returned as 'attributes' – these vary depending on the datatype.

The source of the data will be returned as an OpenCorporates source object

Example:

https://api.opencorporates.com/v0.4/data/2601371

3.1.3.13 statements/gazette_notices/search

This returns a collection of gazette notices that match the given search term (submitted as :q in the query parameters). It's important to note that the search is deliberately quite loose but requires the returned gazette notices to have all the searched-for words (in any order).

As discussed in the general characteristics of the API, search is case-insensitive. Other modifiers applicable to searches (changing order using ElasticSearch, facets, etc., are applicable to this call):

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jurisdiction_code</td>
<td>Term filter</td>
</tr>
<tr>
<td>publication_title</td>
<td>Term filter</td>
</tr>
<tr>
<td>normalised_classification_level_1</td>
<td>Term filter</td>
</tr>
</tbody>
</table>

Example:

https://api.opencorporates.com/v0.4/statements/gazette_notices/search?q=bar
3.1.3.14 statements/control_statements/search

This returns a collection of control statements that match the given search terms.

As discussed in the general characteristics of the API, search is case-insensitive. Other modifiers applicable to searches (changing order using ElasticSearch, facets, etc., are applicable to this call):

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_ultimate_beneficial_owner</td>
<td>whether the controlling entities are the &quot;ultimate beneficial owners&quot; of the controlled entity</td>
</tr>
<tr>
<td>has_controlling_entities</td>
<td>whether any controlling entities are present in the statement</td>
</tr>
<tr>
<td>control_mechanisms_mechanism_type Term filter</td>
<td>the types of control (e.g. share ownership, voting rights). See the schema for the range of values</td>
</tr>
<tr>
<td>controlling_entities_entity_type Term filter</td>
<td>the types of controlling entities (e.g. company, person). See the schema for the range of values</td>
</tr>
<tr>
<td>created_at Date filter</td>
<td>when the control statement was created in the OpenCorporates system</td>
</tr>
<tr>
<td>controlled_entity_name Term filter</td>
<td>the name of the controlled entity</td>
</tr>
<tr>
<td>controlling_entities_name Term filter</td>
<td>the name of the controlling entities</td>
</tr>
</tbody>
</table>

Example:
https://api.opencorporates.com/v0.4/statements/control_statements/search

3.1.3.15 statements/trademark_registrations/search

Returns all trademark registrations matching the given search term.

Parameters:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>The search term, matching the trademark mark text.</td>
</tr>
<tr>
<td>order</td>
<td>created_at (default)</td>
</tr>
<tr>
<td></td>
<td>The date the trademark registration was created in OpenCorporates mark_text</td>
</tr>
<tr>
<td></td>
<td>The trademark mark text</td>
</tr>
<tr>
<td></td>
<td>registration_date</td>
</tr>
<tr>
<td></td>
<td>The date the trademark was registered</td>
</tr>
<tr>
<td></td>
<td>expiry_date</td>
</tr>
<tr>
<td></td>
<td>The date the trademark expired, or is due to expire</td>
</tr>
<tr>
<td></td>
<td>score</td>
</tr>
</tbody>
</table>
Example:

https://api.opencorperatorates.com/v0.4/statements/trademark_registrations/search?q=bar&order=expiry_date

### 3.1.3.16 statements/:id

This returns a 'statement' in the OpenCorporates system. A statement is a purported 'statement of fact' from a source (a public record or a user). For example, subsidiary statement may have been parsed from a filing at the US Securities And Exchange Commission, or a user may have made a statement that one company is a parent of another.

All statements have the following attributes:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datatype</td>
<td>the type of statement, e.g. subsidiary, shareholding, branchrelationship</td>
</tr>
<tr>
<td>properties</td>
<td>the core of what is being said, for example for a licence, the regulator that has issued the licence, the type of licence (e.g. 'Bank') etc</td>
</tr>
<tr>
<td>provenances</td>
<td>one or more provenances for the statement. A provenance is made up of a datetime, a source_url, an actor which created the statement, and a confidence that the statement is true</td>
</tr>
</tbody>
</table>

Example:

https://api.opencorperatorates.com/v0.4/statements/11499887

### 3.1.3.17 placeholders/:id

A placeholder is what we call something we believe is probably a company. We use placeholders as the intermediate link between statements and companies, or, if we can't match the statement to a company (for example because for example we don't have companies in the given jurisdiction) as, well, placeholders for when we can match to a company.

Each placeholder will be returned with the source and one or many of the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>the name of the entity</td>
</tr>
<tr>
<td>jurisdiction_string</td>
<td>A plain text representation for the jurisdiction in which the placeholder is believed to be incorporated - this may be the name of a jurisdiction, of a country, or possibly an ISO 3166-2 code.</td>
</tr>
<tr>
<td>jurisdiction_code</td>
<td>the code for the jurisdiction in which the placeholder is believed to be incorporated (see <a href="#get-companies/:id">GET company</a>). This is inferred by OpenCorporates from the jurisdiction_string.</td>
</tr>
<tr>
<td>identifier</td>
<td>an identifier that is associated with the placeholder, for example an SEC CIK code.</td>
</tr>
<tr>
<td>opencorporates_url</td>
<td>the url of the placeholder.</td>
</tr>
</tbody>
</table>
3.1.3.18 placeholders/:id/network

This returns the immediate ‘computed corporate network’ for the given placeholder as a set of control relationships (i.e. one company is thought to control or influence another company). This is the same data you can see on a company’s network page on the main OpenCorporates site.

A control relationship is a relationship of control from the subject entity to another entity (a ‘child’ relationship), or from another entity to the subject entity (a ‘parent’ relationship). These relationships are computed automatically from a number of relationship statements associated with each entity, and the network will change as more statements are added, because for example new relationships are discovered, or statements are added which say that a previous statement is no longer true (see statements for more details), or even that our algorithms for calculating relationships are improved. Each relationship is made of the following attributes:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>relationship_type</td>
<td>the type of relationship, e.g. subsidiary, shareholding. A subsidiary relationship typically comes from statutory filings (e.g. SEC 10-K reports); shareholding information typically comes from company registers, and is more granular</td>
</tr>
<tr>
<td>relationship_properties</td>
<td>any additional properties of the relationship, e.g. control percentage</td>
</tr>
<tr>
<td>confidence</td>
<td>a computed confidence in the relationship based on the underlying statements</td>
</tr>
<tr>
<td>parent_name</td>
<td>the name of the parent entity</td>
</tr>
<tr>
<td>parent_type</td>
<td>the type of entity the parent is</td>
</tr>
<tr>
<td>parent_opencorporates_url</td>
<td>the OpenCorporates URL of the parent entity</td>
</tr>
<tr>
<td>child_name</td>
<td>the name of the child entity</td>
</tr>
<tr>
<td>child_type</td>
<td>the type of entity the child is</td>
</tr>
<tr>
<td>child_opencorporates_url</td>
<td>the OpenCorporates URL of the child entity</td>
</tr>
</tbody>
</table>

The response is not paginated and can return no results (if we have no known relationships) or hundreds of relationships. It currently returns just a depth of 1 (i.e. just immediate parents and children). We are looking at adding greater levels of depth. Note the related entities may be companies, or placeholders. Placeholders are objects that we haven’t yet matched to a company, either because we don’t have the companies in that jurisdiction, or because we can’t confidently match it to a company. Note that some networks are circular (A controls B, which controls C, which controls A), and you should allow for this when making repeated network calls so that you do not get stuck in an endless loop.

Filter options:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>confidence</td>
<td>a computed confidence in the relationship based on the underlying statements (between 0 and 100). Only relationships above this confidence will be returned (the default value here is</td>
</tr>
</tbody>
</table>
60). You can also pass in 'all' here to get all relationships, irrespective of the confidence in the data.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dates</strong></td>
<td>By default, the network data will be what we think to be true today (usually because the information on which it is based is recent enough to make that a reasonable assumption). However, if a date is passed in (in ISO 8601 format), it will be the network data we have that is believed to be current on that date. This is useful for data such as the US banks, for which we hold good data going back over 10 years.</td>
</tr>
<tr>
<td><strong>ownership_percentage</strong></td>
<td>By default the API will return every control relationship we know about where the percentage control is greater or equal to 2%. However, this can be overriden by passing a numeric value here, for example 50 or even 0 (for every relationship, regardless of percentage). Note that this will not affect the results for control relationships where we do not know the percentage (e.g. many subsidiary relationships).</td>
</tr>
</tbody>
</table>

Example:

```
https://api.opencorporates.com/v0.4/placeholders/645258/network
https://api.opencorporates.com/v0.4/placeholders/645258/network?confidence=80
```

### 3.1.19 placeholders/id/statements

**Description:** This returns the statements associated with a given placeholder. A statement is a purported 'statement of fact' from a source (a public record or a user). For example, subsidiary statement may have been parsed from a filing at the US Securities And Exchange Commission, or a user may have made a statement that one company is a parent of another. For more details, see the statements section below.

Example:

```
https://api.opencorporates.com/v0.4/placeholders/645258/statements
https://api.opencorporates.com/v0.4/placeholders/645258/statements?page=2
```

### 3.1.20 jurisdictions

**Description:** This returns the list of all the jurisdictions we know about (not all of which we have companies for), as well as the jurisdiction_code for the jurisdiction.

Example:

```
https://api.opencorporates.com/v0.4/jurisdictions
```

### 3.1.21 jurisdictions/match

**Description:** This matches the name of a jurisdiction to the jurisdiction. It tries to be highly accurate but accepts a wide variety of formats. For example, "GB", "G.B.", "United Kingdom", "Great Britain" all return the jurisdiction for the UK. "Delaware, USA", "Delaware (USA)", "DELAWARE", "DE (US)" all return the jurisdiction of the US state of Delaware.

You can also supply an optional :related_jurisdiction_code to help the matcher disambiguate between say "Georgia" the US State and Georgia, the Eastern European country, or between "PA" (the US state of Pennsylvania) and PA (for Panama).

Example:

```
https://api.opencorporates.com/v0.4/jurisdictions/match?q=The+Netherlands
https://api.opencorporates.com/v0.4/jurisdictions/match?q=PA&related_jurisdiction_code=us
```
3.1.3.22 industry_codes

From v0.4 OpenCorporates has moved to a new way of representing industry codes (previously we only catered for UK SIC codes), and we can now handle a wide variety of different industry codes, including US NAICS codes and EU NACE codes (and their derivatives). Where a company register makes available the industry codes, we now store that code, together with the code scheme which it belongs to, e.g. For this Belgian company, the industry code consists of the code scheme (in this case be_nace_2008, which represents the NACE-BEL 2008 code scheme) and the code 66191 (which in NACE-BEL 2008 is the code for 'Agenten en makelaars in bankdiensten'). This can be represented as a uid (in this case 'be_nace_2008-66191') to make searching by industry codes consistent and straightforward.

We also map different code schemes to each other, and for those code schemes that can be mapped (such as ones derived from the EU NACE 2 scheme) you can also use the mapped schemes. So, in the above example you can search companies by the industry code uid of 'eu_nace_2-66191' as well as 'be_nace_2008-66191'. You can get information on individual industry codes and their mapped equivalents at the industry_codes/:code_scheme_id/:code call.

This call returns all the code schemes we know about, including links to the official pages for the code scheme. See industry_codes/:code_scheme_id to get a list of the codes associated with each code scheme.

Example:
https://api.opencorporates.com/v0.4/industry_codes

3.1.3.23 industry_codes/:code_scheme_id

Description: This call returns further details about the code_scheme, together with the list of industry codes associated with it

Example:
https://api.opencorporates.com/v0.4/industry_codes/be_nace_2008

3.1.3.24 industry_codes/:code_scheme_id/:code

Description: This call returns the details of a specific code.

Example:
https://api.opencorporates.com/v0.4/industry_codes/be_nace_2008/66191

3.1.4 API accounts, authentication and authorisation

No registration is required to use the OpenCorporates API, but this comes with strong usage limits: up to 500 API requests per month. Therefore, registering for an API account is recommended. There are two types of accounts:

- Free API accounts for open data projects or products, i.e. the product or database in which the data is incorporated is also released under an open licence (specifically share-alike attribution, with attribution to OpenCorporates as detailed at the OpenCorporates licence page).
- Paid-for API Accounts, which remove the OpenCorporates share-alike restrictions (on the rights acquired by OpenCorporates in assembling and cleaning up the data, particularly database rights).

Authentication in the API is done through a simple API token, which should be submitted with each request in the query parameters. This token is available from the account page on OpenCorporates. For example, the data from a company using the API key would be obtained with the following call:

https://api.opencorporates.com/companies/gb/00102498?api_token=yourapitokengoeshere
The status of an account can be obtained by calling the method [https://OpenCorporates.com/users/account](https://OpenCorporates.com/users/account), returning the following values:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plan</td>
<td>the type of plan.</td>
</tr>
<tr>
<td>expiry_date</td>
<td>when the api_plan runs out. If on a regular payment plan, this will be the date of the expiry of the current payment period.</td>
</tr>
<tr>
<td>status</td>
<td>status of the API account.</td>
</tr>
<tr>
<td>usage</td>
<td>this has two subvalues - today and this_month.</td>
</tr>
<tr>
<td>calls_remaining</td>
<td>this has two subvalues - today and this_month.</td>
</tr>
</tbody>
</table>

Example:

[https://api.opencorporates.com/v0.4/account_status?api_token=yourapitokengoeshere](https://api.opencorporates.com/v0.4/account_status?api_token=yourapitokengoeshere)
3.2 Open Refine Reconciliation API

3.2.1 General description

The Open Refine Reconciliation API allows Open Refine users to match company names to legal corporate entities. This is especially useful when you have an existing spreadsheet or dataset featuring lots of companies. Matching (or reconciling) to legal entities allows you to get more information about the companies (for example the registered address or statutory filings), and makes it easier to match with other datasets or exchange with other organisations.

3.2.2 Endpoint

The URL (endpoint) of the OpenCorporates reconciliation service is https://opencorporates.com/reconcile.

Open Refine expects the API to return a matching score with each possible result. This is computed by OpenCorporates searching all the companies matching the name of the company, doing a certain amount of normalising of the company name, and restricting to jurisdiction if requested. Then, it is compared the possible results against the given term, and calculated the score based on the similarity. Then, the score is adjusted, adjusting it down if the company is inactive, and also if it's a foreign branch of a company. This ensures that home companies, and current companies score more highly.

Restricting to jurisdictions. Many company names are quite common across different locations, and there is the additional problem of foreign branches, which are usually named the same as the home company. To avoid false matches, it's a good idea to limit matches to a specific jurisdiction. This is very easy to do with the Open Refine Reconciliation API, as it provides a specific reconciliation URL (or endpoints) for each jurisdiction. You just add the jurisdiction_code for the jurisdiction to the end of the normal reconciliation URL, e.g. https://opencorporates.com/reconcile/es for Spain, instead of https://opencorporates.com/reconcile.

Additional attributes. You can currently submit two additional parameters with each company – the jurisdiction code and a relevant date. The jurisdiction code allows the search to be restricted to a specific jurisdiction which may vary from company to company (e.g. an EU dataset). A submitted date changes the scoring behaviour, meaning that a company will score higher if it was active at that date, and score significantly lower if it didn't yet exist, or was inactive at that date. This is useful if you are matching data from the past, so that you want to match the companies which were relevant to the date of the datapoint.

3.2.3 API accounts, authentication and authorisation

The reconciliation API usage does not require any type of user registration either, and indeed there is no option for user registration for the usage of this functionality.

The reconciliation is under a share-alike attribution Open Database Licence. This means data can be reused, even commercially, provided you release the resultant data under the same share-alike attribution licence.
3.3 OpenOpps API

3.3.1 General description

The OpenOpps API provides access to tender and contract data from a range of European government bodies. The data is formatted according to the Open Contracting Data Standard (OCDS), a data format that is focused on enabling the disclosure of data and documents at all stages of the contracting process. OCDS was initially release in November 2014 and is being adopted by the Open Contracting Partnership.

The OCDS standard was created mostly to support organizations to increase contracting transparency and allow deeper analysis of contracting data by a wide range of users, which is clearly inline with some of the general objectives of the TheyBuyForYou project.

The contents of this section are based on the current online documentation available at the OpenOpps API as well as in the OCDS website, especially in what respects to the data model. A webinar on the usage of the OpenOpps API is also available at:


(access restricted to TheyBuyForYou consortium members only). The objective of this section is to provide a general overview of the main data models used by the API as well as the main types of resources and calls that are made available by it.

Among the main data offered by the API for a given tender, we can find information about the parties involved in the contracting process, the planning about the process (including goals, budgets, projects to which the contracting process relates to), core information about the tender, information on the awards made as part of the contracting process, and details on the implementation of each contract towards completion. More details are provided in section Error! Reference source not found.. As an example, next we show an excerpt of the response to the following API call https://openopps.com/api/tbfy/ocds.

```json
{
  "count": 1622,
  "next": "https://openopps.com/api/tbfy/ocds/?page=2",
  "previous": null,
  "results": [
    {
      "ocid": "ocds-8c46vo-0102-CSL_2018_A1RpoDy4es",
      "countryname": "France",
      "language": "fr",
      "source": "td_achatpublic_fr",
      "releasedate": "2018-05-31T00:00:00Z",
      "date_created": "2018-06-01T01:34:54.430619",
      "json": {
        "uri": "https://openopps.com/tenders/ocds-8c46vo-0102-CSL_2018_A1RpoDy4es/?format=json",
        "license": "https://opendatacommons.org/licenses/odbl/",
        "releases": [
          {
            "id": "CSL_2018_A1RpoDy4es",
            "tag": [
              "tender"
            ],
            "date": "2018-05-31T00:00:00+00:00",
            "ocid": "ocds-8c46vo-0102-CSL_2018_A1RpoDy4es",
            "publishedDate": "2018-05-31T00:00:00+00:00",
            "uri": "https://openopps.com/tenders/ocds-8c46vo-0102-CSL_2018_A1RpoDy4es/?format=json",
            "license": "https://opendatacommons.org/licenses/odbl/"
          }
        ]
      }
    }
  ]
}
```

6 https://openopps.com/api/tbfy/
The endpoint for the API is openopps.com/api/, and it is particularly deployed for the TheyBuyForYour at openopps.com/api/tbfy. An account is necessary in order to make use of the API, which needs to be requested to the OpenOpps support team. Once an account is available, the API can be authenticated using JSON Web Tokens (JWT). Finally, all API calls return JSON.

We will now focus on some of the main characteristics of the API:

**Identifiers.** Resources are identified using an OpenOpps URI, which is composed by a base URI and the Open Contracting ID (OCID). For example, the following URI provides access to the HTML representation of the tender with OCID ocds-0c46vo-0009-DN337739-1: `https://openopps.com/tenders/ocds-0c46vo-0009-DN337739-1/`. This same tender may be obtained in JSON format using the following URL: `https://openopps.com/tenders/ocds-0c46vo-0009-DN337739-1/?format=json`. However, no content negotiation is provided in these URLs.
**Versioning.** The API does not currently support versioning.

**Pagination.** Those calls returning a number of results (e.g., OCDS searches) return a paginated response. By default, 10 objects are returned (although this can be further specified with the parameter page_size), together with a link to the next page. The page parameter allows establishing the page number that we may want to return (e.g., page=4 to access page number 4).

**Provenance.** The information about every tender is obtained from a source that is specified inside the core JSON object using the parameter “source” (e.g., “source”: “td_due_north_newsite”).

**Filters.** Different types of filters can be applied as parameters to the search related method calls in the API: OCID, source, country name, release date (which can be also specified as a range), end date (which can be also specified as a range). Filters are case sensitive.

**Search results.** All method calls that return search results can be ordered by the following parameters: source, publication date, OCID and creation date.

### 3.3.2 Data model

In this section we provide a brief description of the main elements in the data model used by the OpenOpps API. As aforementioned, this data model is based on the OCDS common data model, but without having all the
richness of such data model. The graphical representation provided in
Figure 3 only provides a representation to those classes for which there is information available in the current OpenOpps API.
Figure 3. Overview of the OpenOpps (a subset of OCDS) data model
3.3.3 Method calls

In this subsection we provide details of the methods implemented in the OpenOpps API, related to the previous data model. A complete summary of these methods is provided in Annex I. All of the methods use the HTTP verb GET (POST is used for API authentication mechanisms, so it is explained in section 3.3.4).

3.3.3.1 tbfy/ocds

This returns a list of tenders from the OpenOpps database. As discussed in the section on general characteristics of the API, this call can be filtered using several parameters, and the search results can be ordered according to different parameters as well.

Through the data offered by the API, it is possible to obtain information related to the date, buyer, tender, status, documents and, even, a short description of the tender. Among the most relevant fields we can find:

<table>
<thead>
<tr>
<th>OCID</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ocid</td>
<td>Open contracting identifier</td>
</tr>
<tr>
<td>countryname</td>
<td>Name of the country</td>
</tr>
<tr>
<td>language</td>
<td>Language of the country</td>
</tr>
<tr>
<td>source</td>
<td>Source of the information</td>
</tr>
<tr>
<td>releasedate</td>
<td>Date of the tender release</td>
</tr>
<tr>
<td>date_created</td>
<td>Date of tender creation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JSON</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>Uri of the API</td>
</tr>
<tr>
<td>license</td>
<td>Type of license</td>
</tr>
<tr>
<td>publishedDate</td>
<td>Date of publication</td>
</tr>
<tr>
<td>publicationPolicy</td>
<td>Policy of the publication</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLISHER</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>uid</td>
<td>Name of the buyer</td>
</tr>
<tr>
<td>uri</td>
<td>Publisher uri</td>
</tr>
<tr>
<td>name</td>
<td>Publisher name</td>
</tr>
<tr>
<td>scheme</td>
<td>Type of scheme used</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RELEASE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Name of the buyer</td>
</tr>
<tr>
<td>tag</td>
<td>Type of release</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>date</td>
<td>Release date</td>
</tr>
<tr>
<td>ocid</td>
<td>Open contracting identifier of the release</td>
</tr>
<tr>
<td>language</td>
<td>Release language</td>
</tr>
<tr>
<td>initiationType</td>
<td></td>
</tr>
</tbody>
</table>

**BUYER**

| name        | Name of the buyer                |

**TENDER**

<table>
<thead>
<tr>
<th>id</th>
<th>Identifier of the tender</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>Title of the tender</td>
</tr>
<tr>
<td>status</td>
<td>Status of the tender (active or not)</td>
</tr>
<tr>
<td>description</td>
<td>Description of the tender</td>
</tr>
</tbody>
</table>

**ADDRESS**

| countryname | Country of the buyer             |

**TENDER PERIOD**

<table>
<thead>
<tr>
<th>endDate</th>
<th>endDate of the contract</th>
</tr>
</thead>
<tbody>
<tr>
<td>startDate</td>
<td>startDate of the contract</td>
</tr>
</tbody>
</table>

**DOCUMENT**

<table>
<thead>
<tr>
<th>id</th>
<th>Identifier of the tender url</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>Address of the tender url</td>
</tr>
<tr>
<td>format</td>
<td>Format of the document</td>
</tr>
<tr>
<td>language</td>
<td>Language of the document</td>
</tr>
<tr>
<td>documentType</td>
<td></td>
</tr>
</tbody>
</table>

Example:

https://openopps.com/api/tbfy/ocds/
3.3.3.2 tenders/:ocid

This returns a specific tender identified by its OCID. The information provided for a tender is the same as discussed above for the general search term.

3.3.4 API accounts, authentication and authorisation

As discussed in the introduction, the OpenOpps API needs to be authenticated using JSON Web Tokens (JWT). Tokens must be acquired by making a POST request to /api/api-token-auth/ containing the username and password previously provided by OpenOpps. The headers and data payloads for this POST call are provided below:

Headers: Content-Type: application/json
Data: {"username": "user", "password": "pass"}

Result:

```
{
"token":
"eyJ0eXAiOiJKV2ZiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoxOTQxLCJlbWFpbCI6ImZ5ZWRyb0BmaS51cG0uZXMiLCJ1c2VybmFtZSI6ImZ5ZWRyb0BmaS51cG0uZXMiLCJleHAiOjE1Mjc4NDg4NzR9.diSYbXm3e1iw8W3t43s0MSUXBFy0UlwxNCzd1Jb16G"
}
```

As a result of this request, a token is generated, which can be used for a limited amount of time. This token needs to be used in the headers of all the API GET requests to authenticate and receive authorisation to use the API, as shown in the following figure:

![Figure 4. API GET request with authorisation](image)

3.4 Zaragoza API and SPARQL-based access

3.4.1 General Description

The open data policies of the city council of Zaragoza, which have been implemented for more than 10 years, make the city operate an open-data-by-default policy for all the public sector information that the city council is handling. Among many other data sources, this is also the case for the type of data that is of interest for TheyBuyForYou.

The contents of this section are based on the current offering of open data that the city council of Zaragoza provides in its open data portal, API and SPARQL endpoint on the areas of public procurement (tenders) and invoices. The objective of this section is to provide a general overview of the main data models used in both cases, as well as the main types of resources and calls that are made available on the API and the Zaragoza SPARQL endpoint.
Public procurement data (tenders) is made available at the open data portal in the dataset with name “Perfil del Contratante” ([https://www.zaragoza.es/sede/servicio/catalogo/147](https://www.zaragoza.es/sede/servicio/catalogo/147)). By browsing the metadata from this dataset, it can be observed that this data is made available via the Zaragoza SPARQL endpoint ([https://www.zaragoza.es/sede/portal/datos-abiertos/servicio/sparql](https://www.zaragoza.es/sede/portal/datos-abiertos/servicio/sparql)). This RDF data is organised according to the Public Procurement Ontology (PPROC). As an example, next we show the JSON-LD code that corresponds to contract 1380962-2017.

```json
{
    "@context": {
        "contzar": "http://www.zaragoza.es/api/recurso/sector-publico/contrato/",
        "orgzar": "http://www.zaragoza.es/api/recurso/sector-publico/organismo/",
        "xsd": "http://www.w3.org/2001/XMLSchema#",
        "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
        "pc": "http://purl.org/procurement/public-contracts#",
        "cpv": "http://purl.org/cpv/2008/",
        "pproc": "http://contsem.unizar.es/def/sector-publico/pproc#",
        "org": "http://www.w3.org/ns/org#",
        "s": "http://schema.org/",
        "dcterms": "http://purl.org/dc/terms/",
        "gr": "http://purl.org/goodrelations/v1#",
        "pproc:estimatedDuration": {
            "@type": "xsd:duration"
        },
        "pproc:urgencyType": {
            "@type": "@id"
        },
        "pproc:procedureType": {
            "@type": "@id"
        },
        "pproc:mainObject": {
            "@type": "@id"
        },
        "gr:hasCurrencyValue": {
            "@type": "xsd:decimal"
        },
        "gr:valueAddedTaxIncluded": {
            "@type": "xsd:boolean"
        },
        "pproc:priceReviewAllowable": {
            "@type": "xsd:boolean"
        },
        "pproc:prorogationsExpected": {
            "@type": "xsd:nonNegativeInteger"
        },
        "pproc:provisionalFinancialGuarantee": {
            "@type": "xsd:string"
        },
        "pproc:finalFinancialGuarantee": {
            "@type": "xsd:float"
        },
        "pproc:finalFinancialGuaranteeDuration": {
            "@type": "xsd:duration"
        },
        "pproc:tenderDossierStartDate": {
            "@type": "xsd:date"
        },
        "pproc:tenderDeadline": {
            "@type": "xsd:dateTime"
        }
    }
}
```
"pproc:noticeDate": {
    "@type": "xsd:date"
},
"pproc:frameworkAgreementDerivativeContract": {
    "@type": "@id"
},
"pproc:tenderSubmissionLocation": {
    "@type": "@id"
},
"pproc:awardDate": {
    "@type": "xsd:date"
},
"pproc:formalizedDate": {
    "@type": "xsd:date"
},
"@id": "contzar:1380962-2017",
"@type": ["pproc:Contract", "pproc:ServicesContract", "pproc:MultiannualContract",
"pproc:ContractWithoutLots"],
"dcterm:title": "Contratación del servicio de bar en el cdm santa isabel",
"dcterm:identifier": "1380962-2017",
"pproc:managingDepartment": {
    "@id": "orgzar:serviciodeinstalacionesdeportivas",
    "@type": "org:Organization",
    "dcterms:title": "Servicio de Instalaciones Deportivas"
},
"pc:contractingAuthority": {
    "@id": "orgzar:1",
    "@type": "org:Organization",
    "dcterms:title": "AYUNTAMIENTO DE ZARAGOZA"
},
"pproc:contractingBody": {
    "@id": "orgzar:CDVD",
    "@type": "org:Organization",
    "dcterms:title": "Concejal Delegado de Vivienda y Deporte del Ayuntamiento de Zaragoza"
},
"pproc:contractTemporalConditions": {
    "@id": "contzar:1380962-2017/ContractTemporalConditions",
    "@type": "pproc:ContractTemporalConditions",
    "pproc:estimatedDuration": "P48M"
},
"pproc:contractProcedureSpecifications": {
    "@id": "contzar:1380962-2017/ContractProcedureSpecifications",
    "@type": "pproc:ContractProcedureSpecifications",
    "pproc:urgencyType": "pproc:Regular",
    "pproc:procedureType": "pproc:RegularOpen",
    "pproc:notice": [{
        "@id": "contzar:1380962-2017/ContractNotice/anuncioBOA",
        "@type": "pproc:ContractNotice",
        "pproc:noticeDate": "2018-02-20",
        "pproc:noticeSite": "BOA",
    }, {
        "@id": "contzar:1380962-2017/ContractNotice/anuncioPerfil",
        "@type": "pproc:ContractNotice",
        "pproc:noticeDate": "2018-02-20",
        "pproc:noticeSite": "BOA",
    }]}
"pproc:noticeSite": "Perfil del contratante",
}],
"pproc:contractAdditionalObligations": {
"@id": "contzar:1380962-2017/ContractAdditionalObligations",
"@type": "pproc:ContractAdditionalObligations",
"pproc:provisionalFinancialGuarantee": "0",
"pproc:finalFinancialGuarantee": "5",
"pproc:finalFinancialGuaranteeDuration": "P24M",
"pproc:advertisementAmount": "Gastos de anuncios: Los gastos derivados de la inserción de anuncios en boletines y cualesquiera otras publicaciones serán de cuenta del adjudicatario"
},
"pproc:tenderDossierStartDate": "2018-02-06",
"pproc:tenderDeadline": "2018-03-07T00:00:00",
"pproc:tenderInformationProvider": [{
"@id": "contzar:1380962-2017/TenderInformationProvider/1",
"@type": "pproc:InformationProvider",
}, {
"@id": "contzar:1380962-2017/TenderInformationProvider/2",
"@type": "pproc:InformationProvider",
"s:location": {
"@id": "orgzar:26043",
"@type": "s:Place",
"s:name": "Ayuntamiento de Zaragoza, Edificio Seminario, Planta Baja, Servicio de Contratación",
"s:address": {
"@id": "orgzar:26043/PostalAddress",
"@type": "s:PostalAddress",
"s:streetAddress": "Vía Hispanidad nº20",
"s:postalCode": "50009",
"s:addressLocality": "Zaragoza",
"s:addressCountry": "Spain"
}
}
}],
"pproc:tenderSubmissionLocation": "orgzar:26043"
},
"pproc:contractObject": {
"@id": "contzar:1380962-2017/ContractObject",
"@type": "pproc:ContractObject",
"pproc:mainObject": "cpv:code-55410000",
"pproc:provision": {
"@id": "contzar:1380962-2017/Object",
"@type": "gr:Offering",
"s:name": "CONTRATACIÓN DEL SERVICIO DE BAR EN EL CDM SANTA ISABEL",
"dcterms:title": "CONTRATACIÓN DEL SERVICIO DE BAR EN EL CDM SANTA ISABEL"
},
"pproc:contractEconomicConditions": {
"@id": "contzar:1380962-2017/ContractEconomicConditions",
"@type": "pproc:ContractEconomicConditions",
"pproc:budgetPrice": [{
"@id": "contzar:1380962-2017/priceNoVAT",
"@type": "pproc:BundlePriceSpecification",
"gr:hasCurrencyValue": "13200.00",
"pproc:provision"}]
},
"pproc:tenderDossierStartDate": "2018-02-06",
"pproc:tenderDeadline": "2018-03-07T00:00:00",
"pproc:tenderInformationProvider": [{
"@id": "contzar:1380962-2017/TenderInformationProvider/1",
"@type": "pproc:InformationProvider",
}, {
"@id": "contzar:1380962-2017/TenderInformationProvider/2",
"@type": "pproc:InformationProvider",
"s:location": {
"@id": "orgzar:26043",
"@type": "s:Place",
"s:name": "Ayuntamiento de Zaragoza, Edificio Seminario, Planta Baja, Servicio de Contratación",
"s:address": {
"@id": "orgzar:26043/PostalAddress",
"@type": "s:PostalAddress",
"s:streetAddress": "Vía Hispanidad nº20",
"s:postalCode": "50009",
"s:addressLocality": "Zaragoza",
"s:addressCountry": "Spain"
}
}
}],
"pproc:tenderSubmissionLocation": "orgzar:26043"
},
"pproc:contractObject": {
"@id": "contzar:1380962-2017/ContractObject",
"@type": "pproc:ContractObject",
"pproc:mainObject": "cpv:code-55410000",
"pproc:provision": {
"@id": "contzar:1380962-2017/Object",
"@type": "gr:Offering",
"s:name": "CONTRATACIÓN DEL SERVICIO DE BAR EN EL CDM SANTA ISABEL",
"dcterms:title": "CONTRATACIÓN DEL SERVICIO DE BAR EN EL CDM SANTA ISABEL"
},
"pproc:contractEconomicConditions": {
"@id": "contzar:1380962-2017/ContractEconomicConditions",
"@type": "pproc:ContractEconomicConditions",
"pproc:budgetPrice": [{
"@id": "contzar:1380962-2017/priceNoVAT",
"@type": "pproc:BundlePriceSpecification",
"gr:hasCurrencyValue": "13200.00",
"pproc:provision"}]
},
"pproc:tenderDossierStartDate": "2018-02-06",
"pproc:tenderDeadline": "2018-03-07T00:00:00",
"pproc:tenderInformationProvider": [{
"@id": "contzar:1380962-2017/TenderInformationProvider/1",
"@type": "pproc:InformationProvider",
}, {
"@id": "contzar:1380962-2017/TenderInformationProvider/2",
"@type": "pproc:InformationProvider",
"s:location": {
"@id": "orgzar:26043",
"@type": "s:Place",
"s:name": "Ayuntamiento de Zaragoza, Edificio Seminario, Planta Baja, Servicio de Contratación",
"s:address": {
"@id": "orgzar:26043/PostalAddress",
"@type": "s:PostalAddress",
"s:streetAddress": "Vía Hispanidad nº20",
"s:postalCode": "50009",
"s:addressLocality": "Zaragoza",
"s:addressCountry": "Spain"
}
}
}],
"pproc:tenderSubmissionLocation": "orgzar:26043"
"gr:valueAddedTaxIncluded": "false",
"gr:hasCurrency": "EUR"
},
  
  
  
  "@id": "contzar:1380962-2017/priceWithVAT",
  "@type": "pproc:BundlePriceSpecification",
  "gr:hasCurrencyValue": "13200.00",
  "gr:valueAddedTaxIncluded": "true",
  "gr:hasCurrency": "EUR"
}]

},

"pproc:tenderRequirements": {
  
  "@id": "contzar:1380962-2017/TenderRequirements",
  "@type": "pproc:TenderRequirements",
  "pproc:tenderSubmissionSupportedLanguage": "es"
},

"pc:tender": [{
  
  "@id": "contzar:1380962-2017/Tender1",
  "@type": "pc:Tender",
  "pc:supplier": {
    "@id": "contzar:1380962-2017/Tender1/Supplier",
    "@type": "org:Organization",
    "s:name": "BRYAN ALAVA BRITO",
    "org:identifier": "04738212M",
    "org:classification": "kosOrg:NIF"
  }
},

  
  
  
  },

  "@id": "contzar:1380962-2017/Tender2",
  "@type": "pc:Tender",
  "pc:supplier": {
    "@id": "contzar:1380962-2017/Tender2/Supplier",
    "@type": "org:Organization",
    "s:name": "CRITIAN GUTIERREZ CAIZA",
    "org:identifier": "26279419X",
    "org:classification": "kosOrg:NIF"
  }
},

  
  
  
  },

  "@id": "contzar:1380962-2017/Tender3",
  "@type": "pc:Tender",
  "pc:supplier": {
    "@id": "contzar:1380962-2017/Tender3/Supplier",
    "@type": "org:Organization",
    "s:name": "MARTHA CAIZA GUAMAN",
    "org:identifier": "26059730V",
    "org:classification": "kosOrg:NIF"
  }
},

  
  
  
  },

  "@id": "contzar:1380962-2017/Tender4",
  "@type": "pc:Tender",
  "pc:supplier": {
    "@id": "contzar:1380962-2017/Tender4/Supplier",
    "@type": "org:Organization",
    "s:name": "FABRICIO ALAVA BRITO",
    "org:identifier": "04738212M",
    "org:classification": "kosOrg:NIF"
  }
},

  
  
  
  },

  "@id": "contzar:1380962-2017/Tender5",
  "@type": "pc:Tender",
  "pc:supplier": {

"@id": "contzar:1380962-2017/Tender5/Supplier",
"@type": "org:Organization",
"s:name": "CRISTINA PEMAN FERNANDEZ",
"org:identifier": "291221867",
"org:classification": "kosOrg:NIF"
}

}, {
"@id": "contzar:1380962-2017/Tender6",
"@type": "pc:Tender",
"pc:supplier": {
  "@id": "contzar:1380962-2017/Tender6/Supplier",
  "@type": "org:Organization",
  "s:name": "HOSTELEROS EBRO ZARAGOZA 2008, S.L.",
  "org:identifier": "899185316",
  "org:classification": "kosOrg:NIF"
}

}, {
"@id": "contzar:1380962-2017/Tender7",
"@type": "pc:Tender",
"pc:supplier": {
  "@id": "contzar:1380962-2017/Tender7/Supplier",
  "@type": "org:Organization",
  "s:name": "DANIEL BALLESTEROS JORDAN",
  "org:identifier": "25204813X",
  "org:classification": "kosOrg:NIF"
}

}, {
"@id": "contzar:1380962-2017/Tender8",
"@type": "pc:Tender",
"pc:supplier": {
  "@id": "contzar:1380962-2017/Tender8/Supplier",
  "@type": "org:Organization",
  "s:name": "PAULINA ORTIZ BRITO",
  "org:identifier": "73055785B",
  "org:classification": "kosOrg:NIF"
}

}, {
"@id": "contzar:1380962-2017/Tender9",
"@type": ["pc:Tender", "pproc:AwardedTender", "pproc:FormalizedTender"],
"pproc:awardDate": "2018-05-15",
"pc:supplier": {
  "@id": "contzar:1380962-2017/Tender9/Supplier",
  "@type": "org:Organization",
  "s:name": "WILMER OSSA BUITRAGO",
  "org:identifier": "05005241G",
  "org:classification": "kosOrg:NIF"
},
"pc:offeredPrice": [{
  "@id": "contzar:1380962-2017/Tender9/Item/PriceNoVAT",
  "@type": "pproc:BundlePriceSpecification",
  "gr:hasCurrencyValue": "28800.00",
  "gr:valueAddedTaxIncluded": "false",
  "gr:hasCurrency": "EUR"
}, {
  "@id": "contzar:1380962-2017/Tender9/Item/PriceWithVAT",
  "@type": "pproc:BundlePriceSpecification",
  "gr:hasCurrencyValue": "28800.00",
  "gr:valueAddedTaxIncluded": "true",
  "gr:hasCurrency": "EUR"
}]}
Data about invoices is available on the Zaragoza open data API (https://www.zaragoza.es/docs-api_sede/), more specifically at https://www.zaragoza.es/docs-api_sede/#/Ayuntamiento_Facturas. As an example, next we show the response to https://www.zaragoza.es/sede/servicio/factura/24.

```json
{
  id: "24",
  title: "GASES PARA EL LABORATORIO DE LA PLANTA POTABILIZADORA",
  entidad: {
    id: "23780",
    title: "SERVICIO DE EXPLOTACIÓN DEL AGUA POTABLE",
    url: "http://www.zaragoza.es/api/recurso/sector-publico/organismo/23780"
  },
  tercero: {
    id: 173,
    title: "AL-AIR-LIQUIDE-SA",
    cif: "A28016814"
  },
  facturaElectronica: "N",
  amount: 346.15,
  ejercicio: 2012,
  issued: "2012-01-01T00:00:00Z",
  payment: "2012-05-11T00:00:00Z",
  status: "Pagada"
}
```

The Zaragoza API currently supports both http and https. Besides, the API can be used without or with API keys (the latter recommended for 3rd party developers), with no usage limits in any of the cases. Finally, the API returns several types of formats, depending on the dataset (JSON, CSV, TTL, JSON-LD, GeoJSON, etc.).

We will now focus on some of the main characteristics of the API:

**Identifiers.** Zaragoza follows Linked Data principles in the generation of identifiers for the resources that are made available in the API (as well as those used in the triple store that stores RDF and provides the SPARQL endpoint). Identifiers are usually composed of a base URI (https://www.zaragoza.es/sede/<<domain>>/<<subdomain>>/<<id>>). This follows the Spanish NTI (“Norma Técnica de Interoperabilidad”) guidelines. 

**Versioning.** The API does not currently support versioning. The same applies to the RDF knowledge graph.

**Pagination.** Those calls returning a number of results (e.g., invoice searches) return a paginated response. By default, 50 objects are returned (although this can be further specified with the parameter rows). The start parameter allows establishing the page number that we may want to return (e.g., ?start=4 to access page number 4).

**Provenance.** All information comes from the city council. Therefore, no specific provenance is recorded. In any case, the metadata for each of the accessed data sources specifies the actual source of the whole dataset.

**Filters.** Filters can be applied to any of the records returned in the JSON objects returned by the API. Besides, there is the possibility of evaluating more complex expressions in the FIQL query language (parameter q), as well as specifying the columns/attributes that will be provided as a result (parameter fl).
3.4.2 Data model

In this section we provide a brief description of the main elements in the data model used by the Zaragoza API for invoices, and by the SPARQL endpoint for public contracts. The data model for the invoices is ad-hoc for the city council, based on the data that it has available for them, and it is in the process of being homogenised in the context of an interest group of cities in Spain, so it may evolve in the near future. As aforementioned, the data model for public contracting data is PPROC, and is graphically represented in Figure 5. A larger description of this ontology is provided in deliverable D5.1.

![Graphical representation of the data model for invoices](http://contsem.unizar.es/def/sector-publico/pproc)

The graphical representation provided in Figure 6 describes the data model of the API for invoices.
3.4.3 Method calls and predefined SPARQL queries

In this subsection we provide details of some of the predefined SPARQL queries for the public contracting data published in the Zaragoza SPARQL endpoint, as well as of the method calls in the Zaragoza open data API for the data related to invoices. A complete summary of these queries and methods is provided in Annex I. The API methods use the HTTP verb GET.

3.4.3.1 SPARQL queries for public contracting data

As any SPARQL endpoint, the Zaragoza SPARQL endpoint allows any type of SPARQL query to be evaluated on it, following the recommendations provided in the SPARQL protocol. For the convenience of reusers, a range of SPARQL queries have been generated as examples of the type of queries that can be evaluated in the SPARQL endpoint. These queries are used, for instance, to obtain indicators that are used for transparency purposes. They are maintained in https://github.com/pproc/pproc-sparql and some additional ones are published at https://www.zaragoza.es/ciudad/gestionmunicipal/contratos/enlace/risp/consultas-perfil-contratante.htm

- Contracts by managing department
- Organisations that have worked with Zaragoza on year XXX
- Organisations that have applied for tenders on year XXX
- Average time needed to award a tender
- Steps that a tender has gone through
- Number of formalised contracts in a date range
- Total cost of contracts formalised for organisation XXX
- Tenders in which organisation XXX has participated
- Total amount of money in contracts by managing department
- Total amount and number of contracts per type of contract

### 3.4.3.2 Invoices search: servicio/factura

This returns a list of invoices from the Zaragoza databases. It is possible to obtain information related to the subject of the invoice, whether it is an electronic invoice or not, the amount, the year, when it was issued, whether it has been received, paid or is in the process of being paid, and the payment date, if applicable. It is also possible to obtain the organisation ( “tercero” in Spanish) that the invoice has to be paid to and the managing department ( “entidad” ).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>invoice’s identifier</td>
</tr>
<tr>
<td>title</td>
<td>invoice’s title</td>
</tr>
<tr>
<td>facturaElectronica</td>
<td>describes if it is an electronic invoicing (“S/N” in Spanish, “Y/N” in English)</td>
</tr>
<tr>
<td>amount</td>
<td>total amount to pay</td>
</tr>
<tr>
<td>ejercicio</td>
<td>tax year</td>
</tr>
<tr>
<td>issued</td>
<td>date of payment issuance</td>
</tr>
<tr>
<td>payment</td>
<td>effective payment date</td>
</tr>
<tr>
<td>status</td>
<td>describes the status of the invoice</td>
</tr>
</tbody>
</table>

### 3.4.3.3 Invoices search: servicio/factura/:id

This returns the specific data for a single invoice with the provided id:

```json
https://www.zaragoza.es/sede/servicio/factura/22?r=html
```

```json
{
    "id": "22",
    "title": "MENSAJERIA",
    "entidad": {
        "id": "20232",
        "title": "S. SOCIALES COMUNITARIOS",
        "uri": "http://www.zaragoza.es/api/recurso/sector-publico/organismo/20232"
    },
    "tercero": {
        "id": 10332,
        "title": "GRUPO-LA-VELOZ-SOC-COOPERATIVA",
        "cif": "F50633478"
    }
}
```
3.4.4 API accounts, authentication and authorisation

As discussed in the introduction to this section, the Zaragoza open data API offers the possibility of registering, which is recommended for third party reusers. However, there are no differences in terms of usage limits or authorisation to access other types of data that would not be otherwise available without the corresponding tokens.

In case of making use of authentication mechanisms, access control to the API methods is done by adding two parameters to the headers of the request:

- clientId: identifier of the client making the request.
- HmacSHA1 hash that is generated with the request data.

When the request is received by the server, the HmacSHA1 is regenerated and double checked. If the check is successful, there is a check on the permissions associated to the client, so as to see whether it is authorised. There is also support for Oauth2, using external authentication mechanisms.

With respect to the SPARQL endpoint, its usage does not require any type of user registration either, and indeed there is no option for user registration for the usage of this functionality.
4 Conclusions and future work

As discussed in the introduction, the main objective of this document was to set the basis principles for the architecture of the TheyBuyForYou platform, which can be seen as a collection of REST APIs that will be provided for those willing to make use of the data that will be stored, maintained and published in the TheyBuyForYou knowledge graph. This is a first attempt at providing such an architecture at an early stage of the project, when most of the tools that will make use of the API as well as the business cases have only started to be developed or designed.

Following our agile methodology for software development, and our principles of “release early”, as described in deliverable D5.1, it is expected that the architecture will be evolving over the course of the project, so as to give support to those needs that will be arising from the user stories derived from the needs of the online tools or mostly from the business cases.

It is also especially important to note that the current API catalogue is aiming at providing an overview of the current offering already provided by some of the most prominent data sources that we will be using in the context of the project. Therefore, we have focused on describing how these APIs look like currently, and not yet on identifying their gaps or trying to find how to homogenise them in order to provide, for instance, the same architecture or a compatible architecture for authentication and authorisation, for search and filtering, for ordering, for paging, etc. We have tried to provide a homogeneous description of all of them, which would allow to make some comparisons among them, while at the same time trying to keep the descriptions simple, so that they can be followed easily not only by developers, but also by those working on the online tools and on the business cases.

The next steps, which will be consolidated in the following API and platform deliverable to be delivered on month 15, will be focused on presenting how the TheyBuyForYou API will have evolved beyond the individual APIs that are presented here and available nowadays, providing a consistent view over the knowledge graph. It will also present how these APIs are providing support to the user stories that will have been the subject of work until its delivery, something that will be exemplified with running code and examples that will be also accessible from the project website, as showcases of the work that has been done and the opportunities that the TheyBuyForYou knowledge graph offers.

All developments will continue to be regularly updated in the GitHub repositories of the GitHub organisation that represents the project: http://github.com/tbfy.
## ANNEX I. TheyBuyForYou API cheatsheet for developers

<table>
<thead>
<tr>
<th>Method</th>
<th>Endpoint</th>
<th>Description</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>companies/:jurisdiction_code/:company_number</td>
<td>Core data for the given company</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>companies/:jurisdiction_code/:company_number/data</td>
<td>Data held for the given company</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>companies/:jurisdiction_code/:company_number/filings</td>
<td>Statutory filings for the given company</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>companies/:jurisdiction_code/:company_number/network</td>
<td>Immediate 'computed corporate network' for the given company</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>companies/:jurisdiction_code/:company_number/statements</td>
<td>Statements associated with each company</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>companies/search</td>
<td>Collection of companies whose name matches the given search term</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>corporate_groupings/name</td>
<td>Information on a given CorporateGrouping</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>corporate_groupings/search</td>
<td>Collection of corporate_groupings whose name matches the given search term</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>data/:id</td>
<td>Information on a given datum</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>filings/:id</td>
<td>Information on a statutory filing for the a company</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>industry_codes</td>
<td>Code schemes we know about</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>industry_codes/:code_scheme_id</td>
<td>Further details about the code_scheme</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>industry_codes/:code_scheme_id/:code</td>
<td>Details of a specific code</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>jurisdictions</td>
<td>List of all the jurisdictions we know about</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>jurisdictions/match</td>
<td>Name of a jurisdiction to the jurisdiction</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>officers/search</td>
<td>Collection of officers whose name matches the given search term</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>officers/:id</td>
<td>Information on a particular officer</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>placeholders/:id</td>
<td>Returns placeholders, placeholder is what we call something we believe is probably a company</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>placeholders/:id/network</td>
<td>Immediate 'computed corporate network' for the given company</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>Method</td>
<td>Endpoint</td>
<td>Description</td>
<td>Source</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>GET</td>
<td>placeholders/:id/statements</td>
<td>statements associated with a given placeholder</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>statements/gazette_notices/search</td>
<td>collection of gazette notices that match the given search term</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>statements/control_statements/search</td>
<td>collection of control statements that match the given search terms</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>statements/trademark_registrations/search</td>
<td>trademark registrations matching the given search term</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>statements/:id</td>
<td>'statement' in the OpenCorporates system</td>
<td>OpenCorporates</td>
</tr>
<tr>
<td>GET</td>
<td>tbfy/ocds/</td>
<td>information related to the date, buyer, tender, status, documents and, even, a short description of the tender</td>
<td>OpenOpps</td>
</tr>
<tr>
<td>GET</td>
<td>tenders/ocid</td>
<td>returns a specific tender identified by its OCID</td>
<td>OpenOpps</td>
</tr>
<tr>
<td>GET</td>
<td>servicio/factura</td>
<td>Invoices list</td>
<td>Zaragoza</td>
</tr>
<tr>
<td>GET</td>
<td>servicio/factura/:id</td>
<td>Invoices details</td>
<td>Zaragoza</td>
</tr>
</tbody>
</table>