Tombolo Digital Connector Manual

Exported from the Tombolo GitHub Wiki on 2017-12-04 15:45

The Tombolo Digital Connector is software for interconnecting data-sets and urban models. Its aim is to advance the technology that can help policy makers, planners, designers and developers make better, evidence-based decisions, tailored to the cities they work in.

See also general FAQ

Table of Contents

The below you can find the suggested reading order of this Wiki. On the right hand side you can find an alphabetical list of pages.

Features

The Digital Connector main features are:

- Importers for various open source data sources and transforming them into a centralised data format.
- Importers for proprietary data that has been transformed according to the centralised data format.
- Model recipe language for creating urban models by joining datasets.
- Aggregators and de-aggregators for joining spatial data of different spatial granularities.
- Export data and model output in CSV or GeoJson format to use further in tools such as GIS systems, Jupyter Notebooks, etc.

The main advantages of using the Digital Connector are:

- No need to parse and understand the file format of numerous open data sources.
- Simple aggregation and de-aggregation of different spatial granularities.
- Reusability of model recipes.

- Simplified data gathering from complex APIs, such as Open Street Map.

Using the Digital Connector

In a nutshell, using the Digital Connector involves four steps:

1. After ensuring that you have installed the pre-requisites outlined here, install the Digital Connector by cloning the GitHub repository.
2. Write a recipe file that describes the desired data output from the Digital Connector. This recipe includes which data-sources to use and how to mix the data together.
3. Run the export. In this step the Digital Connector will connect to the relevant data sources, download the necessary data, join the data as per the user recipe and export the data in the requested output format.
4. Work with the data in tools such as QGIS, Jupyter Notebooks, R, etc.

Example: Correlation between deprivation and childhood obesity

Rosie wants to study the correlation between deprivation and childhood obesity. She browses the importable data and finds that MSOA level childhood obesity is available from Public Health England and LSOA level deprivation score is available from DCLG. She writes in her recipe file that the output should be a CSV file with three columns.

- The first column is the MSOA identifier.
- The second column contains the percentage of obese children in that MSOA.
- The third column contains the median LSOA level deprivation value for the MSOA.

After exporting the data, Rosie loads it to a Jupyter Notebook where she uses R to calculate correlation coefficients and render a scatter-plot.

Example: Cycling and pollution

Thomas wants to visualise the relation between bicycle friendly boroughs in London and air quality. He browses the importable data and finds traffic counts from Department for Transport and air quality measurements from the London Air Quality Network. He writes a recipe saying that he wants the the shape of each borough as a feature. He specifies that each borough should have three attributes:

- a name attribute;
- an attribute showing cycle traffic as a fraction of total traffic;
- and an attribute quantifying the air quality.

After exporting the data, Thomas opens the file in QGIS where he visualises it.

For a detailed use case see Use Case on Cycling and Air Quality.

Example: Active Transport Index

As part of the Tombolo project we have built an application called City Index Explorer. The application functions as a demonstrator of the possibilities for combining various urban data sources into an urban model.

One of the indices we have developed for demonstration is the Active Transport Index. The index combines various data sources to assign a score to each LSOA representing the use and potential for active transport.

For details, see Use Case on Active Transport Index

System Architecture

Figure 1 shows a high level system architecture for the Tombolo Digital Connector. The figure shows the main system components and the interlinking between them. We also show the links to external data and systems.

- At the core of the digital connector is the **Local Datastore** where all incoming data is stored in a centralised data format.
- **Importers** are responsible for interpret the incoming data and code it in the centralised data format. We have both built-in importers for **publicly available datasources** as well as support for users to write their own importers for their **proprietary data**. If the proprietary data is available simple shape-files or simple csv files, it may be importable using a generic data importer without any additional implementation.
- **Exporters** are used to export data from the system. The output data can simply be the data values as they originated in the input data, or as a modelled combinations of various data-sources.
- For the modelling part we both have pre-defined **built-in model recipes** as well as support for the user to specify and export their own **proprietary model recipes**.

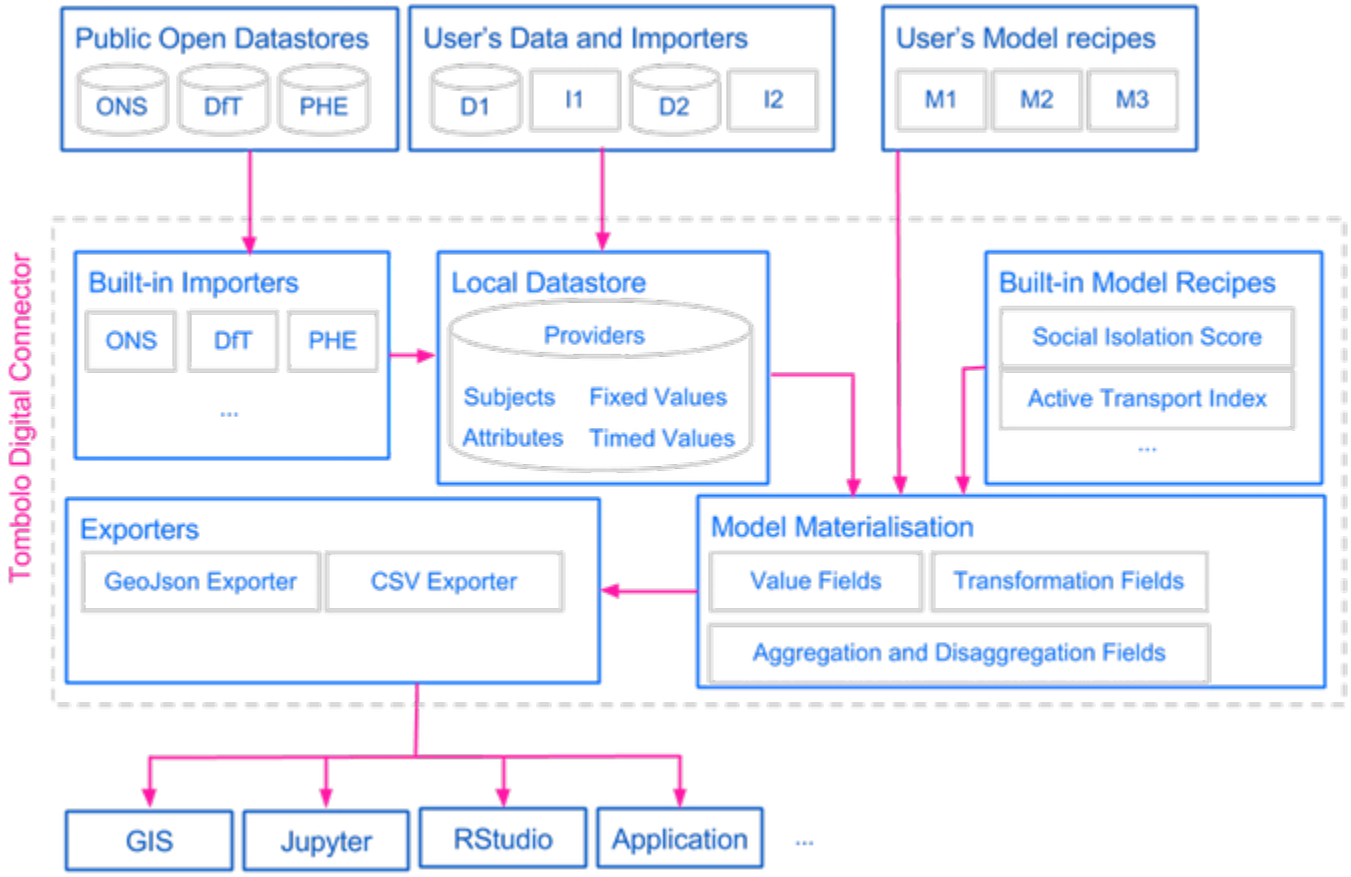

Figure 1: *Overview of the Tombolo Digital Connector components and data workflow.*

Local Datastore

At the core of the Digital Connector there is a centralised data-format for urban data. We refer to the physical implementation of the data-format as the Local Datastore. In the current implementation the data is stored in a PostGIS database instance. Figure 1 shows the Entity Relationship Diagram for the Local Datastore.

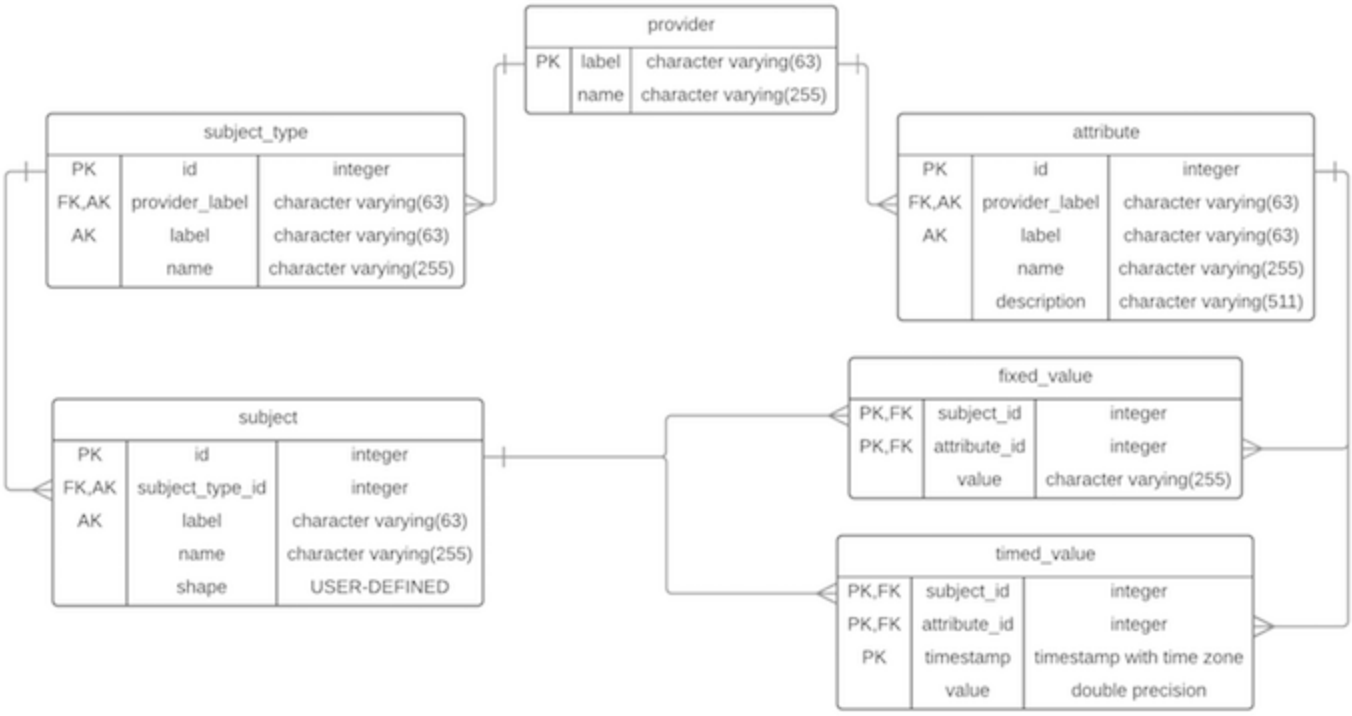

Figure 1: *ERD for the Tombolo Digital Connector centralised data-format (Local Datastore).*

In short, the datastore contains a collection of **subjects** (e.g. traffic-counters, road segments or neighbourhoods) and **attributes** (e.g. bicycle-count, population, etc.). For each subject and each attribute, there is associated a time-series of zero or more values. For attributes that do not change over time we also have the notion of fixed values (e.g., road names).

Below we describe the centralised data format in more detail. Database field names are written in italics and primary keys as boldface.

Provider

Provider is a data object representing sources of data. A provider could be a governmental organisation, such as ONS; or a private entity supplying a publicly available data source, such as Twitter.

A provider consists of:

- ***label:*** a unique label for the provider
- *name:* a name of the provider

Subject Type

Subject Type is the type of subjects imported into the system (see description of subjects below). Subject

types can be various such as traffic counter, IoT sensor, street segment, building, lsoa, msoa, local authority, etc.

A subject type consists of:

- ***provider:*** a provider of the subject type. I.e., the organisation responsible for defining and providing digital version of the subject.
- ***label:*** a label for the subject type. The label is unique for each provider.
- *name*: a human readable name of the subject type.

Subject

Subject is a data object representing any type of subject or entity, which will in most cases refer to geographic objects, although we could support other subject types such as individuals or businesses. We support a both physical geographic subjects, such as street segments and buildings, as well as logical geographic subjects such as a geo-coded tweet or image.

A subject consists of:

- ***subject type:*** type of the subject (See description above).
- ***label:*** a unique label for the subject. The labels are unique within a subject type.
- *name:* a human readable name of the subject.
- *shape:* the geometric shape of the subject, in case it has one.

Attribute

Attribute is a data object representing anything that could be measured or calculated, such as population density (for an LSOA), CO2 concentration (for an air quality sensor), obesity rate (for a local authority), etc.

An attribute consists of:

- ***provider:*** refers to the organisation or source of the data values for this attribute.
- ***label:*** a label of the attribute, unique for each provider.
- *name:* a human readable name of the attribute (e.g. Population density, Obesity rate, CO2 concentration, etc.)
- *description:* a further description of the attribute, e.g. describing the methodology used to generate the attribute values.

Fixed Value

Fixed Value is a fixed attribute that can be assigned to a Subject. Examples can be a the value of a road-type Attribute assigned to a road Subject.

A fixed value consists of:

- ***subject:*** a foreign key reference to a Subject.
- ***attribute:*** a foreign key reference to an Attribute.
- *value:* the actual value of the attribute for the subject, most often a string.

Timed Value

Timed Value is a data object representing the value of an attribute for a certain subject, taken at a certain time point.

A timed value consists of:

- ***subject:*** a foreign key reference to a Subject.
- ***attribute:*** a foreign key reference to an Attribute.
- ***timestamp:*** time point to which the value refers. (E.g. 2015-03-04T10:33:44Z)
- *value:* the actual value for the attribute, subject and timestamp, most often a numeric value.

Importers

In this section we describe the Tombolo Digital Connector importers. The role of importers is to connect to external or local data sources and import the appropriate data by reformatting it into the centralised data format.

Importers are of three types:

- **Subject Importers** import only Subjects. Examples include the LSOA and MSOA importers for the ONS Geographies Portal and the Health Organisation importer for NHS Choices data.
- **Timed Value Importers** import only Timed Values for an externally defined set of Subjects. Examples include the ONS Census importer that import census values for geographies such as LSOAs, MSOAs and Local Authorities.
- **Mixed Importers** import both Subjects and Timed Values. Examples include the Traffic Counts importer which imports both the locations of traffic counts as Subjects and the actual traffic counts as Timed Values.

We have built-in importers for a range of mostly public and open datasets. Users can use these importers directly to import the data they need in the processing. Additionally we have support for users to extend the Digital Connector with their own user defined custom importers for their local proprietary datasets. In the code base we have support tools such as Excel and Shapefile data extraction tools to make it easier to write custom importers.

Built-in importers

Below you can find a list of built-in importers that were available at the time of writing. For an up to date list look at the codebase.

- Department for Communities and Local Government (DCLG)
  - Indices of multiple deprivation (value importer)
- Department for Education (DfE)
  - List of all schools (subject importer)
- Department for Transport (DfT)
  - Accessibility of output areas (value importer)
  - Traffic counts (mixed importer)
- London Air Quality Network (LAQN)
  - Air quality data from Environmental Research Group Kings College London (mixed importer)
- London Datastore
  - Borough profiles (value importer)
  - Public Health Outcomes Framework (PHOF) indicators for London (value importer)
  - Walking and cycling information of London boroughs (value importer)

- NHS Choices
  - Health Organisations (subject importer)
- Office of National Statistics (ONS)
  - Output area geometries (subject importer)
  - Census data (value importer)
  - Claimants data (value importer)
  - Wages data (value importer)
- Open Street Map (OSM)
  - Cycling infrastructure (subjects importer)
  - Greenspace data (subjects importer)
  - Land-use data (subjects importer)
  - Road infrastructure (subjects importer)
- Public Health England (PHE)
  - Adult Obesity data (value importer)
  - Childhood Obesity data (value importer)
- Space Syntax
  - Open Map (mixed importer)
- Transport for London (TfL)
  - Transport stations importer (mixed importer)
- Twitter
  - Geo-coded tweet importer (subject importer)
- Generic Importers
  - Generic CSV Importer (value importer / subject importer / mixed importer)

In the development of importers we have taken a pragmatic approach where we implement importers on need-to-use basis where the city challenges have been in the driver seat. As the project progresses we will be constantly supporting more data sources.

Fields and Models

In this section we describe the Tombolo Digital Connector fields and how modelling can be seen as fields. As with importers we have a set of built-in fields, recipes and models, together with a field specification language where users can define their own custom fields and models.

See also FAQ about fields and model recipes

Value Fields

Value fields are the most basic fields. Their purpose is to being able to export existing raw imported data values or to feed them to one of the nested fields introduced later on in this section.

Currently we have implemented the following value fields:

- **Fixed Annotation Field:** Returns a fixed value for annotation purposes.
- **Fixed Value Field:** Returns the Fixed Value of a specified Attribute for a given Subject.
- **Latest Value Field:** Returns the latest Timed Value for a particular Attribute on the given Subject.
- **Subject Latitude Field:** Returns the latitude of the centroid of the Subject.
- **Subject Longitude Field:** Returns the longitude of the centroid of the Subject.
- **Subject Name Field:** Returns the name of the Subject.
- **Values By Time Field:** Returns all Timed Values on an Attribute for a given Subject.

For an up-to-date list, see here: value fields

Transformation Fields

Transformation fields take as input a set of one or more fields and produce a new field by transforming the values of the input fields.

- **Arithmetic Field:** Takes as input an operation, and two fields. It returns for a given Subject the value resulting from applying the operation on the two field values. Supported operations are addition, subtraction, multiplication and division.
- **Percentiles Field:** Field that returns for a subject and an attribute the percentile in which its value falls, compared to all the values for the same attribute for a set of subjects. Percentiles can be calculated either over the output subjects or any other specified set of subjects. E.g. we could output the quartiles value for the population density of all LSOAs in Leeds where the quartiles boundaries are calculated using population density values for all LSOAs in England.
- **Field Value Sum Field:** Takes a list of fields as input and returns a field consisting of the sum of the input fields (To be replaced by generalised Arithmetic Field).
- **Fraction Of Total Field:** For a subject, returns the sum of its Timed Values for a list of dividend attributes divided by a divisor attribute (To be replaced by generalised Arithmetic Field).

For an up-to-date list, see here: transformation fields

Aggregation and Disaggregation Fields

The aggregation and disaggregation fields are types of transformation fields where the transformation is not only a combination of other fields but also either aggregates values from fine granularities to a coarse granularity or de-aggregates values from coarse granularities to finer granularities.

- **Geographic Aggregation Field:** A field that calculates the value of a field for every other Subject within its geometry and performs some aggregation function on the result. Currently we support sum and mean value aggregations. E.g. a cycle-traffic value for a local authority can be calculated by averaging the cycle-counts from all traffic counters located within that local authority.
- **Map To Containing Subject Field:** This field will find a subject of a given type that contains the provided subject, and then associate a field value with that new subject. For example, if the containing Subject Type is a 'City' and it is given a subject representing a building, it will assign the containing city's field value to the building subject.
- **Map To Nearest Subject Field:** A field that finds the nearest subject of a given Subject Type and then evaluate the field specification with that new subject. For example, if the nearest Subject Type is 'Street' and it is given a subject representing a building, it will evaluate the field specification with a subject representing the Street that building is on.

For an up-to-date list, see here: aggregation fields

Modelling Fields and Built-in Models

Modelling Fields and Built-in models are an important part of the Tombolo Digital Connector. It allows users to share definitions of custom fields and models. They are also used to encode built-in models that have been developed within the Tombolo project to address the City Challenges. Basic Modelling Field: A field that takes as input a specification (recipe) of a potentially complex field and returns a value that is calculated according to the specification. The recipe/specification can also be seen as a model.

At the time of writing we include the following Built-in models:

- **Active Transport Index:** A modelling field combining traffic counts from Department for Transport, cycling infrastructure from Open Street Map and travel to work information from the UK Census.
- **Social Isolation Score:** A modelling field for applying the Age UK model described in the city challenges description below.

These models are them selves combinations of sub models, e.g. for calculating the fraction of households renting in an LSOA. For browsing the available built-in models see modelling fields

Exporters

Since the main goal of the Tombolo Digital Connector is to connect urban data and urban models, there is a large set of urban analytics and model building that takes place outside of the connector. To allow for connections with external systems the Digital Connector provides support for exporting data and model output. Currently there are two data formats supported.

- **GeoJson** is one of the most common data format for geographic data. It allows for easy integration between the Tombolo Digital Connector and Geographic Information Systems such as QGIS.
- **CSV** is one of the most common data format for relational data. It allows for easy integration between the Connector and various data processing and analytics tools.

The workflow of exporting data is core functionality of the current state of the Tombolo Digital Connector. The user creates a recipe file where they describe the output data they would like to get. The recipe file consists of four parts:

- **Subjects:** The user can specify the set of subjects for which data and models are to be exported. As an example, subjects can be all spatial network segments for a specific geographic area, all LSOAs within a certain geographic area, etc.
- **Data-sources:** A list of data-sources needed to be imported in order to export the data. As an example, data-sources can be the Space Syntax Open Space Map (SSx OSM) for the Royal Borough of Greenwich, traffic counts for London from Department for Transport (DFT), etc.
- **Fields:** A list of fields that are to be returned for each subject. As an example, for a set of spatial network segments the user could specify to export the connectivity of each segment according to SSx OSM, the nearest DfT traffic counts for that segment (if available) and a deprivation value for that segment disaggregated from the LSOA level deprivation scores from Department for Communities and Local Government (DCLG). In case a field is a transformation or a modelling field, the needed computation is performed at the time of exporting.
- **Exporter:** The name of the exporter to be used. E.g. GeoJson or CSV.

Note that in the case of built-in model fields, the user does not need to specify the data-sources since they are already included in the built-in model recipe.

Recipe Language

The core means of using the Tombolo Digital Connector is to create a data export recipes, describing the data or models that we want to get out of the system, without describing how to generate the data. The recipe language is expressed in the well-known JSON format.

Data export recipe

A data export recipe consists of two parts:

- **Dataset**: is a description of the data to be exported. The format of the dataset recipe is explained in the Dataset recipe section below.
- **Exporter**: is the canonical name of the Java class to be used to export the data. At the time of writing there are two types of exporters, one for CSV output and one for GeoJson.

Example data export specification for GeoJson output:

```
{
  "dataset" : INSERT-YOUR-DATASET-RECIPE,
  "exporter" : "uk.org.tombolo.exporter.GeoJsonExporter"
}
```

Dataset recipe

A dataset recipe is composed of three parts:

- **Subjects**: Specifies the Subjects for which we return data in the final dataset. I.e., return all LSOAs in Milton Keynes, return all traffic counters in Greenwich, or return all local authorities in England. See further details in the Subject recipe section.
- **Datasources**: A list of data-sources needed to be imported in order to export the data. As an example, data-sources can be the Space Syntax Open Space Map (SSx OSM) for the Royal Borough of Greenwich, traffic counts for London from Department for Transport (DFT), etc. See further details in the Datasource recipe section.
- **Fields**: A list of Fields that are to be returned for each subject. As an example, for a set of spatial network segments the user could specify to export the connectivity of each segment according to SSx OSM, the nearest DfT traffic counts for that segment (if available) and a deprivation value for that segment disaggregated from the LSOA level deprivation scores from Department for Communities and Local Government (DCLG). In case a field is a transformation or a modelling field, the needed computation is performed at the time of exporting. See further details in the Field recipe section.

Example dataset specification skeleton:

```
{
  "subjects" : [INSERT-SUBJECT-RECIPE],
  "datasources" : [INSERT-DATASOURCE-RECIPE],
  "fields" : [INSERT-FIELD-RECIPE]
}
```

Subject recipe

A subject recipe is has two mandatory parts:

- **Provider**: specifies the provider of the subject subject to be returned.
- **Subject type**: specifies the type of the subject to be returned. At the time of writing we support various types, such as, local authority, msoa, lsoa, trafficCounter, SSxNode (a node in a Space Syntax graph), gpSurteries, etc.

Example of a simple subject specification where we return all traffic counters that have been imported.

```
{
  "provider" : "uk.gov.dft",
  "subjectType" : "trafficCounter"
}
```

Additionally a subject recipe can have two types of so-called match rules where we can restrict further the set of Subjects returned:

- **match-rule**: is a filter where we can restrict the returned subjects based on name or label. The match-rule is further composed of:
  - **attribute**: is the attribute on which we would like to filter. E.g. name or label.
  - **pattern**: is a string pattern that the specified attribute should match. We use the SQL like syntax using % as the wildcard. E.g. for filtering all strings that start with the string 'Leeds' we use the pattern 'Leeds%'.
- **geo-match-rule**: is a filter where we can restrict the returned subjects based on geographic constraints.

Example where we return all LSOAs whose name starts with the string 'Leeds':

```
{
  "provider" : "uk.gov.ons",
  "subjectType" : "lsoa",
  "matchRule": {
    "attribute": "name",
    "pattern": "Leeds%"
  }
}
```

Example where we return all London boroughs based on filtering for the label prefix 'E090':

```
{
  "provider" : "uk.gov.ons",
  "subjectType" : "localAuthority",
  "matchRule" : {
    "attribute": "label",
    "pattern": "E090%"
  }
}
```

Example where we return all traffic-counters in Greenwich and Islington:

```
{
  "provider" : "uk.gov.dft",
  "subjectType" : "trafficCounter",
  "geoMatchRule" : {
    "geoRelation" : "within",
    "subjects" : [
      {
        "provider" : "uk.gov.ons",
        "subjectType" : "localAuthority",
        "matchRule" : {
          "attribute" : "name",
          "pattern" : "Greenwich"
        }
      },
      {
```

```
        "provider" : "uk.gov.ons",
        "subjectType" : "localAuthority",
        "matchRule" : {
          "attribute" : "name",
          "pattern" : "Islington"
        }
      }
    ]
  }
}
```

Datasource recipe

The datasource recipe consists of two fields:

- **importer-class**: Which Java class is used to import the data.
- **datasource-id**: The identifier of the dataset to be imported. This value is unique within each importer class.

Example for LSOA geographies

```
{
  "importerClass": "uk.org.tombolo.importer.ons.OaImporter",
  "datasourceId": "lsoa"
}
```

Examples for importing the ONS Census dataset for overcrowding:

```
{
  "importerClass": "uk.org.tombolo.importer.ons.CensusImporter",
  "datasourceId": "QS408EW"
}
```

Datasource geography scope

Some data providers support customisable downloading of their data based on geographic area. E.g. Department of Transport provides traffic counts for individual regions or local authorities.

- **geographyScope**: is a list of geographic areas to import. For the supported values see the code for the corresponding importer.

Example for importing DfT traffic-counts for London:

```
{
  "importerClass" : "uk.org.tombolo.importer.dft.TrafficCountImporter",
  "datasourceId" : "trafficCounts",
  "geographyScope" : ["London"]
}
```

Datasource temporal scope

Some datasources support configurable downloads for some time periods.

- **temporalScope**: is a list of temporal references for which data can be imported. For the supported

values see the code for the corresponding importer.

Field recipe

The field recipe contains a list of fields that you want to output. Fields can be as simple as the latest value for a certain attribute, such as population density, or a potentially complex model, such as one for estimating the risk of social isolation among the elderly. All fields have two default fields (sic):

- **fieldClass**: is the Java class used to calculate the field value
- **label**: the label that the recipe creator chooses to associate with the field

In addition each field has additional fields (sic) depending on the type of the field. For example a latest-value-field you need to specify the attribute for which you want the latest value. E.g.,

"field": { "fieldClass": "uk.org.tombolo.field.value.LatestValueField", "label": "Count of cars and taxis", "attribute": { "provider": "uk.gov.dft", "label": "CountCarsTaxis" } }

As another example the arithmetic-field takes as an input the operation you want to apply and two fields

{ "fieldClass": "uk.org.tombolo.field.transformation.ArithmeticField", "label": "An example of adding two fields", "operation": "add", "field1": INSERT-A-FIELD-RECIPE, "field2": INSERT-A-FIELD-RECIPE }

Further examples can be found in the Fields and Models description page.

Notes

- Introduce two types of scope.
  - Subject scope (optional)
  - Normalisation scope (optional)

HowTo: Importer Implementation

This questionnaire is aimed at evaluating the effort required to implement a Tombolo importer for a data source. Before answering this questionnaire, please familiarise yourself with the Tombolo data model.

Describe your importer

- What are the subjects related to this dataset?
- Are the subjects imported by this datasource or do you link to subjects in an external datasource?
- What are the fixed value attributes for the subjects?
- What are the timed value attributes for the subjects?