European
Commission

SC353DI07171

D04.01: Persistent URIs for NUTS

Document Metadata

Property	Value
Date	2017-09-21
Status	Accepted
Version	1.00
Authors	Makx Dekkers – AMI Consult Ana Fernandez de Soria Risco – PwC EU Services Florian Barthélemy - PwC EU Services Alexandru Droscariu – PwC EU Services
Reviewed by	Nikolaos Loutas – PwC EU Services Susanne Wigard – European Commission, ISA² Programme
Approved by	Susanne Wigard – European Commission, ISA² Programme

This study was prepared for the ISA² Programme by:

PwC EU Services

Disclaimer:

The views expressed in this report are purely those of the authors and may not, in any circumstances, be interpreted as stating an official position of the European Commission.

The European Commission does not guarantee the accuracy of the information included in this study, nor does it accept any responsibility for any use thereof.

Reference herein to any specific products, specifications, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favouring by the European Commission.

All care has been taken by the author to ensure that s/he has obtained, where necessary, permission to use any parts of manuscripts including illustrations, maps, and graphs, on which intellectual property rights already exist from the titular holder(s) of such rights or from her/his or their legal representative.

Table of Contents

1 Introduction

In the context of the implementation of the persistent URI policy for EU institutions[1], the current document describes the result of an analysis of the Nomenclature of Territorial Units for Statistics (NUTS)[2] and proposes an RDF model, and the approach to provide persistent URIs for NUTS codes. Besides the details of the NUTS RDF model, the document proposes a set of URI patterns, and a way to preserve links between different NUTS versions. It concludes with a description of an implemented pilot showcasing the publication of NUTS as linked data.

The publication of NUTS codes as linked data could help users/Eurostat to exploit information which originally is encoded only by postcodes or local administrative units. Other users could track changes over time using links between NUTS versions, and users of e-procurement or e-tendering systems could link their data to persistent NUTS codes to enhance data discovery.

[1] https://joinup.ec.europa.eu/community/semic/news/eu-institutions-define-common-persistent-uri-service

[2] http://ec.europa.eu/eurostat/web/nuts/overview

2 OBJECTIVES AND SCOPE

The objectives of the report, in alignment with the SEMIC Action, were focusing on the implementation of Persistent URIs inside EU institutions, in this case Eurostat and the Publications Office of the EU. Through this implementation, the ISA² Programme intends to:

1. Propose persistent URIs for NUTS codes and an RDF NUTS model to showcase the practical use of persistent URIs. The persistent URIs would enable the NUTS maintenance team to ensure to all the NUTS re-users a consistent access to the online information available for each code. This information would be accessible in RDF format, allowing SPARQL queries on its triple store to retrieve specific information contained in the RDF NUTs model.
2. Guide the implementation of persistent URIs for other collections. From this report, the NUTS maintenance team should understand how to assign persistent URIs to the future versions of the NUTS code and how to update the current RDF model with the new version of the codes.

In the scope of this report:

1. Provisioning of persistent NUTS URIs.
2. A pilot project to demo the use of persistent NUTS URIs. This section of the report should clearly explain how an application was built based on the RDF NUTS model and the persistent URIs developed and how it can be used.

3 Approach

A collaborative approach was implemented during the whole process in order to achieve the objectives defined. The main contributors to this activity were Eurostat and the Publications Office of the European Union, as owners and maintainers of the NUTS codes and ISA^2/PwC, as the main responsible for the activity. A small review group was formed by several stakeholders of the activity and supported by PwC under the SEMIC project. The Figure 1 Main stakeholders of the activity presents the two main groups of stakeholders for this activity and their role. It is further detailed in the description of the steps below.

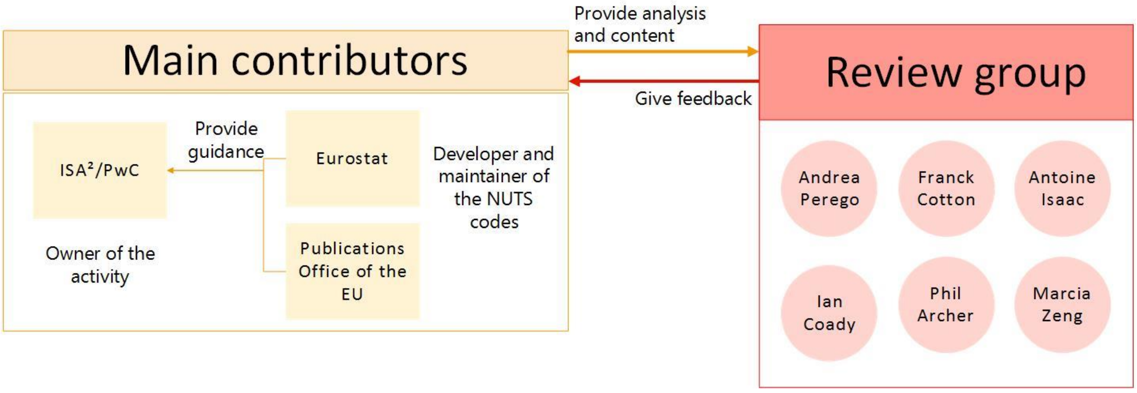


Figure 1 Main stakeholders of the activity

The approach to draft the current report follows the steps listed below:

1. Analyse documentation on NUTS, and draft initial proposal for governance and management of NUTS URIs (PwC). The initial analysis and description of a first proposal took place in February. The initial proposal contained a description of how NUTS should be modelled in RDF, how to deal with NUTS URIs governance and change management, and a pilot demo to show the use of NUTS URIs. Specific questions were already addressed to some of the contributors (Eurostat) to confirm the analysis of the documentation.
2. Establish contact with reviewers of the task group and ask to provide feedback on the results of analysis and proposal (OP, Eurostat, and PwC).
3. Review proposal and provide feedback (reviewers). A small group of reviewers was formed and multiple issues on the proposal were submitted between 21 March 2017 and 18 April 2017.
4. Refine proposal based on the received feedback (PwC). The different comments received from reviewers were analysed and the issues identified were presented during two different calls (5 April and 7 May 2017) with the group of reviewers. The group decided for each issue if it should be accepted, rejected, closed or opened in the context of this work.
5. Build a pilot, to demo the use of NUTS as linked data (PwC).

4 NUTS LEGAL BASIS

The Nomenclature of Territorial Units for Statistics (NUTS) acquired a legal status with the European Commission Regulation No 1059/2003[3]. This regulation sets out the rules for the NUTS and for future amendments of the classification. The rules for future amendments ensure the consistency of the NUTS through time. The NUTS classification divides the economic territory of the Member States into territorial units and assign to each territorial unit a specific code and name.

As anticipated by the regulation, several amendments were integrated into the initial regulation[4], for example when new countries joined the European Union in 2005 and 2008.

The latest version of the NUTS levels 1, 2 and 3 breakdown will be effective as from 1 January 2018, following the Regulation (EC) No 2066/2016 amending the initial regulation of 2003.

The latest effective version of the NUTS applicable in the European Union was released before 9 December 2013 and the Regulation No 1319/2013 which provided the NUTS levels 1, 2 and 3 breakdown effective as of 1 January 2015[5]. The different NUTS versions can be found on the Eurostat website: http://ec.europa.eu/eurostat/web/nuts/history.

The European Institutions have different roles in the development and monitoring of the NUTS. First, Eurostat is responsible of the development and maintenance of the different versions of the NUTS classification. The role of the European Commission is to propose the changes as amendments to the Regulation No 1059/2003. And finally, the European Parliament scrutinises the propositions from the Commission and stresses certain aspects of the NUTS development requiring a specific attention such as "the treatment of smaller administrative units" or the most pertinent level "to identify the area in which an integrated policy for territorial development might best be implemented"[6].

[3] http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:32003R1059
[4] http://ec.europa.eu/eurostat/web/nuts/legislation
[5] http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:32013R1319
[6] http://www.europarl.europa.eu/atyourservice/en/displayFtu.html?ftuId=FTU_5.1.6.html

5 Modelling NUTS in RDF

5.1 Modelling considerations

The following sections introduce the entities that are modelled in this specification. The formal definitions of the entities as classes with their properties are detailed in sections 5.2 through 5.7.

5.1.1 NUTS code

The entity *NUTS Code* is a description of the characteristics of a code, including its notation, its label, its level, its status and relationships with other codes and with NUTS Versions.

It is modelled as a *Concept* as defined in SKOS (class *skos:Concept*): an idea, a notion or unit of thought.

NUTS Codes are not versioned because their meaning is stable – a code is withdrawn if fundamental aspects change, e.g. when the region that it represents changes.

As NUTS Codes are not versioned and NUTS Versions (see below) are versioned, a NUTS Code may be a member of more than one NUTS Version.

Changes in label are modelled using two labels: the 'preferred label' (property *skos:prefLabel*) and the 'alternative label' (property *skos:altLabel*). The preferred label is the label that is/was associated with the Code in the most recent NUTS version of which it is/was a member. All other labels that were associated with the Code in the past are included as *skos:altLabel*.

At this time (2017), neither the latest nor earlier versions of NUTS have parallel labels in multiple languages. For multilingual countries, a single label may contain names in more than one language, e.g. "Région de Bruxelles-Capitale/Brussels Hoofdstedelijk Gewest". However, in the current specification, the preferred label is repeatable to allow for future use of parallel, multilingual labels.

Post Codes and Local Administrative Units are made available by Eurostat for download from the NUTS website[7] as Excel files linked from the NUTS-3 codes. These files contain only the Post Codes or LAU codes that correspond to that particular NUTS-3 code.

Relationships between codes are modelled as 'replacement', 'merging' or 'splitting':

- Replacement is the situation when the Code is changed but the geospatial characteristics remain the same.
- Merging is the situation when two Codes that represent smaller regions A and B are replaced by one Code that represents the region C that combines the regions A and B: C=A+B.
- Splitting is the situation that a Code that represents a region G is replaced by two or more Codes, e.g. D and E, that represent regions that together are equal to region G: D=G-E and E=G-D.

Codes on a NUTS level with a lower level number are 'broader' than NUTS Codes with a higher level number: a Code in NUTS-2 is 'broader' than a Code in NUTS-3, because

[7] http://ec.europa.eu/eurostat/web/nuts. Post Codes: http://ec.europa.eu/eurostat/web/nuts/correspondence-tables/postcodes-and-nuts; Local Administrative Units: http://ec.europa.eu/eurostat/web/nuts/local-administrative-units

the Code on NUTS-2 represents a larger area that includes the areas represented by the NUTS-3 Code. Conversely, the Code in NUTS-3 is 'narrower' than the Code in NUTS-2. Following this approach, NUTS-2 Code FRC1 (France—Bourgogne) is 'broader' than NUTS-3 Code FRC11 (France—Côte-d'Or) as the area represented by FRC11 is part of the area represented by FRC1.

5.1.2 Geometry

The entity *Geometry* is a description of the geospatial characteristics of the region represented by the NUTS Code, including its type (Region, Line or Label Point), the generalisation which is the scale of the data and the projection as expressed with a code from the EPSG Geodetic Parameter Registry[8].

This entity is modelled as a *Geometry* as defined in the Core Location Vocabulary (class *locn:Geometry*): a location identified as a point, line, polygon, etc. expressed using coordinates in some coordinate reference system. It is also modelled as a *Dataset* as defined in DCAT (class *dcat:Dataset*): a collection of data, published or curated by a single agent, and available for access or download in one or more formats.

The modelling as DCAT *Dataset* allows to link to the data file modelled as a DCAT *Distribution*.

5.1.3 Distribution

The entity *Distribution* is a description of the data that is associated with a *Geometry*, including the URL from where the file containing the data can be downloaded, the format of the file and the licence under which the data is available.

It is modelled as a *Distribution* as defined in DCAT (class *dcat:Distribution*): a specific available form of a dataset.

5.1.4 NUTS version

The entity *NUTS Version* is a description of a published version of NUTS, including the date of its publication, its name and description, and a link to the Regulation that governs its publication.

It is modelled as a *Concept Scheme* as defined in SKOS (class *skos:ConceptScheme*): an aggregation of one or more concepts.

*NUTS Version*s are versioned with the year of publication as the distinguishing characteristic. Links to previous and next versions are provided if such versions exist using properties *dct:replaces* and *dct:isReplacedBy*.

The relationship between a *NUTS Code* and a *NUTS Version* is included in the description of the *NUTS Code*. The full set of *NUTS Code*s that are members of a particular *NUTS Version* can be generated by selecting all *NUTS Code*s that link to that *NUTS Version*.

5.1.5 Regulation

The entity *Regulation* is a description of the legal act that governs the publication of a particular *NUTS Version*.

[8] http://www.epsg-registry.org/

It is modelled as a *Legal Resource* as defined in the ontology of the European Legislation Identifier. The link between the NUTS Version and the Regulation is expressed using the property *dct:isRequiredBy*.

This specification does not specify the elements of the description of the *Regulation*. Such a description can be based on the elements in the ELI ontology and other vocabularies.

5.2 Namespaces

The following namespaces are used in the remainder of this section.

Prefix	Reference	Namespace URI
dcat	Data Catalog Vocabulary. https://www.w3.org/TR/vocab-dcat/	http://www.w3.org/ns/dcat#
dct	DCMI Metadata Terms. http://dublincore.org/documents/dcmi-terms/	http://purl.org/dc/terms/
eli	European Legislation Identifier. http://publications.europa.eu/mdr/eli/	http://data.europa.eu/eli/ontology#
locn	ISA Programme Location Core Vocabulary. https://www.w3.org/ns/locn	http://www.w3.org/ns/locn#
nutsdef	Model elements for the NUTS Linked Open Data model defined in this specification (Reference to vocabulary to be determined)	To be determined, e.g. http://data.europa.eu/<xyz>/; <xyz> to be assigned by the URI Committee
skos	SKOS Simple Knowledge Organization System. https://www.w3.org/2004/02/skos/	http://www.w3.org/2004/02/skos/core#
xsd	XML Schema	http://www.w3.org/2001/XMLSchema#

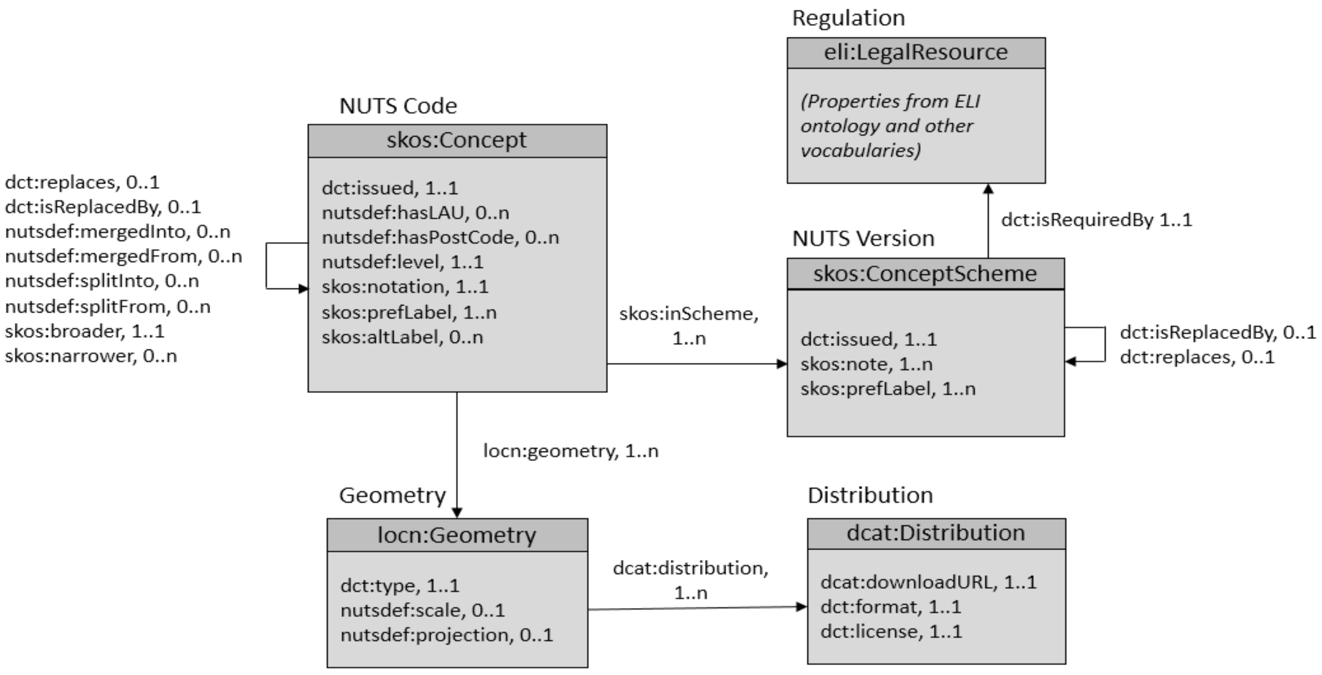
Note that the URI for the elements in the NUTS model are not in the same namespace as the instance data for the entities.

5.3 Classes

Label	Definition	Class URI
NUTS code	An idea, a notion or unit of thought. https://www.w3.org/TR/skos-reference/#concepts	skos:Concept
NUTS Version	An aggregation of one or more SKOS concepts https://www.w3.org/TR/skos-reference/#schemes	skos:ConceptScheme

Label	Definition	Class URI
Distribution	A specific available form of a dataset. https://www.w3.org/TR/vocab-dcat/#class-distribution	dcat:Distribution
Geometry	A location identified as a point, line, polygon, etc. expressed using coordinates in some coordinate reference system. https://www.w3.org/ns/locn#locn:Geometry This class is at the same time modelled as a Dataset to enable the data to be linked as a Distribution: A collection of data, published or curated by a single agent, and available for access or download in one or more formats. https://www.w3.org/TR/vocab-dcat/#class-dataset	locn:Geometry, dcat:Dataset
Regulation	A legal act or any component of a legal act, like an article. In: http://publications.europa.eu/mdr/eli/documentation/ELI_Ontology-v1.1.pdf	eli:LegalResource

5.4 Conceptual model diagram



5.5 Data types

Table for types of Literals:

Label	Definition	URI
NUTS code notation	A string of characters conforming to the pattern: Two uppercase alphabetic characters from the set of country codes defined at http://ec.europa.eu/eurostat/statistics-explained/index.php/Glossary:Country_codes, followed by zero, one, two or three alphanumeric characters from the set A-Z, 0-9.	nutsdef:notation
Date	An interval of exactly one day in length	xsd:date
String	A sequence of characters	xsd:string

5.6 Properties per class

5.6.1 NUTS code

Label	Definition	Range	Card.
dct:issued	Date of formal issuance (e.g., publication) of the resource. http://dublincore.org/documents/dcmi-terms/#terms-issued. This is the issue date of the first NUTS Version in which the described NUTS Code appeared.	rdfs:Literal, datatype xsd:date	1..1
dct:isReplacedBy	A related resource that supplants, displaces, or supersedes the described resource. http://dublincore.org/documents/dcmi-terms/#terms-isReplacedBy. This refers to a new NUTS Code that replaces the described NUTS Code.	rdfs:Resource	0..1
dct:replaces	A related resource that is supplanted, displaced, or superseded by the described resource. http://dublincore.org/documents/dcmi-terms/#terms-replaces. This refers to a NUTS Code that is replaced by the described NUTS Code.	rdfs:Resource	0..1

Label	Definition	Range	Card.
locn:geometry	Associates any resource with the corresponding geometry. https://www.w3.org/ns/locn#locn:geometry. This refers to the description of the geometry associated with the described NUTS Code.	locn:Geometry	1..n
nutsdef:hasLAU	A document that contains a list of the Local Administrative Units corresponding to the described NUTS code.	foaf:Document	0..1
nutsdef:hasPostCode	A document that contains a list of the Post Codes corresponding to the described NUTS code.	foaf:Document	0..1
nutsdef:level	A value that indicates the level of the Code; permissible values are 0, 1, 2 or 3.	rdfs:Literal, datatype xsd:integer (0|1|2|3)	1..1
nutsdef:mergedFrom	This refers to a NUTS Code in a preceding NUTS Version that represented a smaller area that is part of the area represented by the described NUTS Code. Subproperty of dct:replaces.	skos:Concept	0..n
nutsdef:mergedInto	This refers to a NUTS Code in a subsequent NUTS Version that represents an area that includes the area represented by the described NUTS Code. Subproperty of dct:isReplacedBy.	skos:Concept	0..1
nutsdef:splitFrom	This refers to a NUTS Code in a preceding NUTS Version that represented a larger area that includes the area represented by the described NUTS Code. Subproperty of dct:replaces.	skos:Concept	0..1
nutsdef:splitInto	This refers to a NUTS Code in a subsequent NUTS Version that represents an area that is part of the area represented by the described NUTS Code. Subproperty of dct:isReplacedBy.	skos:Concept	0..n

Label	Definition	Range	Card.
skos:broader	A broader concept. https://www.w3.org/TR/skos-reference/#semantic-relations This refers to a NUTS Code on a higher NUTS level than the described NUTS Code.	skos:Concept	0..1
skos:inScheme	A Concept Scheme of which the described Concept is a member. https://www.w3.org/TR/skos-reference/#schemes This is the NUTS Version of which the described NUTS Code is a member.	skos:ConceptScheme	1..n
skos:narrower	A narrower concept. https://www.w3.org/TR/skos-reference/#semantic-relations This is a NUTS Code on a lower NUTS level than the described NUTS Code.	skos:Concept	0..n
skos:notation	A string of characters used to uniquely identify a concept within the scope of a given concept scheme, and not normally recognizable as a word or sequence of words in any natural language. https://www.w3.org/TR/skos-reference/#notations This is the character string for the described NUTS Code, e.g. "UKC11".	rdfs:Literal, datatype nutsdef:notation	1..1
skos:prefLabel	A preferred lexical label for the described Resource. https://www.w3.org/TR/skos-reference/#labels This is the label for the described NUTS Code. It is repeatable to allow for labels in multiple languages.	rdfs:Literal, datatype xsd:string plus language tag	1..n

Label	Definition	Range	Card.
skos:altLabel	An alternative lexical label for the described Resource. https://www.w3.org/TR/skos-reference/#labels This is an alternative label for the described NUTS Code. It is used to record previous labels in cases where a code was relabelled.	rdfs:Literal, datatype xsd:string plus language tag	0..n

5.6.2 Geometry

Label	Definition	Range	Card.
dcat:distribution	Connects a dataset to its available distributions. https://www.w3.org/TR/vocab-dcat/#Property:dataset_distribution This refers to a description of a file that contains the data for the Geometry.	dcat:Distribution	1..n
dct:type	The nature or genre of the resource. http://dublincore.org/documents/dcmi-terms/#terms-type This is the kind of the Geometry. The values must be taken from a controlled vocabulary, see section 5.7.1.	skos:Concept	1..1
nutsdef:scale	The scale or generalisation of the Geometry.	rdfs:Literal, datatype integer	0..1
nutsdef:projection	The projection of the Geometry expressed as a code from the EPSG Geodetic Parameter Registry, http://www.epsg-registry.org/	rdfs:Literal, datatype xsd:string	0..1

5.6.3 Distribution

Label	Definition	Range	Card.
dcat:downloadURL	A file that contains the distribution of the dataset in a given format. https://www.w3.org/TR/vocab-dcat/#Property:distribution_downloadurl This refers to the location where the data can be downloaded.	rdfs:Resource	1..1
dct:format	The file format, physical medium, or dimensions of the resource. http://dublincore.org/documents/dcmi-terms/#terms-format The values must be taken from a controlled vocabulary, see section 5.7.2.	dct:MediaTypeOrExtent	1..1
dct:license	A legal document giving official permission to do something with the resource. http://dublincore.org/documents/dcmi-terms/#terms-license The values must be taken from a controlled vocabulary, see section 5.7.3.	dct:LicenseDocument	0..1

5.6.4 NUTS version

Label	Definition	Range	Card.
dct:isReplacedBy	A related resource that supplants, displaces, or supersedes the described resource. http://dublincore.org/documents/dcmi-terms/#terms-isReplacedBy This refers to a newer NUTS version.	rdf:Resource	0..1
dct:isRequiredBy	A related resource that requires the described resource to support its function, delivery, or coherence. http://dublincore.org/documents/dcmi-terms/#terms-isRequiredBy This refers to the Regulation that forms the legal basis for the described NUTS version.	rdf:Resource	1..1

Label	Definition	Range	Card.
dct:issued	Date of formal issuance (e.g., publication) of the resource. http://dublincore.org/documents/dcmi-terms/#terms-issued This is the issue date of the described NUTS Version.	rdfs:Literal, datatype xsd:date	1..1
dct:replaces	A related resource that is supplanted, displaced, or superseded by the described resource. http://dublincore.org/documents/dcmi-terms/#terms-replaces This refers to the previous NUTS version.	rdf:Resource	0..1
skos:note	General documentation. https://www.w3.org/TR/2009/NOTE-skos-primer-20090818/#secdocumentation This gives a description of the NUTS version. It is repeatable to allow for descriptions in multiple languages.	rdfs:Literal, datatype xsd:string with language tag	1..n
skos:prefLabel	A preferred lexical label for the described Resource. https://www.w3.org/TR/skos-reference/#labels This is the name of the NUTS Version, e.g. "NUTS 2016". It is repeatable to allow for labels in multiple languages.	rdfs:Literal, datatype xsd:string with language tag	1..n

5.6.5 Regulation

The Regulation under which the NUTS Version is issued is identified with a European Legislation Identifier (ELI) and can be described by properties defined in the ELI ontology (see http://publications.europa.eu/mdr/eli/) and properties from other vocabularies.

5.7 Controlled vocabularies

5.7.1 Type of Geometry

For the type of Geometry, three values are relevant: Region, Line and Label point. URIs for these three types must be minted as part of a controlled vocabulary.

5.7.2 Format of Distribution

For the format of Distributions, the MDR File type NAL must be used. See: http://publications.europa.eu/mdr/authority/file-type/. The formats GeoJSON and TopoJSON should be added to that NAL.

5.7.3 Licence of Distribution

For the licence of Distributions, the MDR Licence NAL must be used. See: http://publications.europa.eu/mdr/authority/licence/.

5.8 Versioning

The proposed approach is to version the NUTS as a whole, not to version the individual codes.

NUTS codes are not versioned because semantic changes, e.g. major changes in the territory that the code refers to, lead to creation of a new code and retirement of the old code (e.g. EL2 was replaced by EL6 between NUTS2010 and NUTS2013). Codes that remain the same across NUTS versions can simply have two occurrences of *skos:inScheme* pointing to two *Concept Schemes*.

The only change that occurs, albeit sporadically, is a change in the label associated with a code while the code remains unchanged. An example is the change for code PL2 that had label MAKROREGION POŁUDNIOWY in NUTS2016 and REGION POŁUDNIOWY in NUTS2013.

As it is not necessary to track which label was associated with the code at a specific time, the current label can be expressed with *skos:prefLabel* and previous labels with *skos:altLabel*. This enables finding the correct code when searching on any of the labels that were associated with the code at any time.

5.9 URI patterns

Entity	URI pattern	Example URI
NUTS Version	http://data.europa.eu/nuts/scheme/<year>	http://data.europa.eu/nuts/scheme/2016
Code	http://data.europa.eu/nuts/code/<XYnnn>	http://data.europa.eu/nuts/code/ES521
Geometry	http://data.europa.eu/nuts/geometry/<XYnnn>-<type>-<scale>-<projection> with <type> one of (region, line, labelpoint) and <scale> expressed as 1m, 3m etc.	http://data.europa.eu/nuts/geometry/ES521-region-1m-EPSG:4258 http://data.europa.eu/nuts/geometry/ES521-labelpoint
Distribution	http://data.europa.eu/nuts/distribution/<XYnnn>-<type>-<scale>-<projection>/<format> with <format> one of (shape, geojson, topojson, pbf, wkt)	http://data.europa.eu/nuts/distribution/ES521-region-1m-EPSG:4258/geojson

Note that the URI for the elements in the NUTS model are not in the same namespace as the instance data for the entities. For the model namespace, a separate URI collection ID will be requested.

6 Pilot to demo the use of NUTS URIs

This chapter describes the pilot project. The output of the project is a web application[9] which shows on a map the evolution of NUTS codes between 2010, 2013 and 2016. The codes are linked to each other by the changes (split, merged or replaced).

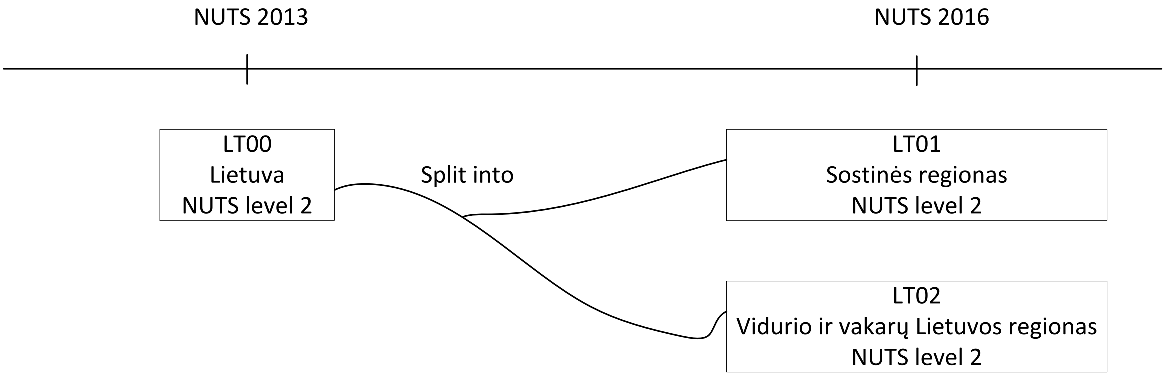


Figure 2 – Links between codes of different versions of NUTS

Figure 2 displays an example of a NUTS 2013 code split into two NUTS codes in 2016. The subsections below explain the technical aspects of the implementation of the pilot and the final results.

6.1 Technical documentation

This section provides technical details such as the programming language used, type of database used, where to download or access the tool, etc.

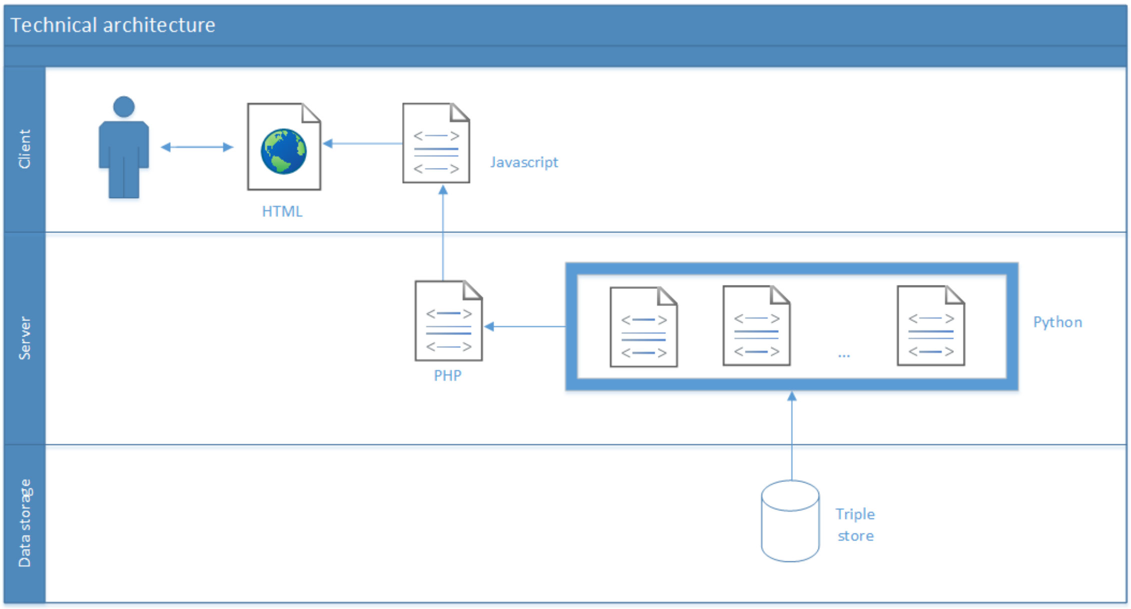


Figure 3 - Technical architecture

[9] http://52.50.205.146:8890/NUTS_visualisation/map.html

Figure 3 depicts a high-level overview of the pilot and the technologies used:

- HTML, to implement the user interface of the pilot;
- Javascript, to update the visualisation;
- PHP, to execute the python scripts;
- Python, to query the triple store in order to get information using SPARQL; and
- Triple store where the NUTS information is stored according to the data model implemented (see section 5).

Client

The user accesses the map visualisation through an **HTML** website, which is deployed in a commercial cloud server. The HTML uses **javascript** to execute the actions, which are mainly drawing the map and updating the regions according to the year and NUTS level selected.

Server

The javascript needs to retrieve stored information from the database. To do so, the pilot executes **python** scripts which contain the necessary queries. Since the javascript is on the client side and the database on the server, there is a **PHP** file in between the javascript and python files. Depending on the input arguments, the PHP file executes the corresponding python script to get the data.

Data storage

The python scripts query the triple store using the SPARQL query language. The database that the pilot uses is a Virtuoso triple store and the graph "http://52.50.205.146/NUTS-codes". The update of the map and evolution of codes is checked in the database via the available SPARQL endpoint:

http://52.50.205.146/sparql

Input files

The pilot uses the NUTS geometries that are published in the official Eurostat website[10], i.e. it downloads the geometries published by Eurostat and stores them locally to draw the map. The geometries are available in different formats, of which the pilot uses GeoJSON (60M). The geometries are stored in a javascript file per NUTS level and year.

In addition, the pilot takes as input the list of NUTS codes and changes from 2010-2013 and 2013-2016 that are published in the Eurostat NUTS history website[11]. The spreadsheets are downloaded from the Eurostat website and transformed into NUTS RDF (see section 5). The transformation is performed by using Google Refine and python scripts created to automatically generate the triples. This procedure is explained in section 6.3 in more detail.

It is important to take into account that the geometries for 2016 NUTS codes are not published yet. However, the spreadsheet with the NUTS changes from 2013 to 2016

[10] http://ec.europa.eu/eurostat/statistical-atlas/gis/viewer/distribution/v1/

[11] http://ec.europa.eu/eurostat/web/nuts/history

is available. Therefore, the pilot 2016 map is drawn using the 2013 geometries, but it reflects the changes from 2013 to 2016.

Structure of the source code

The source code can be found on GitHub:

https://github.com/SEMICeu/NUTS_pilot

The structure of the source code consists of:

- datasets – Folder with the files storing the NUTS geometries, which are transformed into JSON variables. The folder contains a file with a JSON variable per NUTS year and level (e.g. "nuts0_2010_data" contains the NUTS geometries of the NUTS level 0 in 2010);
- scripts – Folder with the scripts to be called and executed from the main page and css configuration. It consists of:
  - map.css – CSS file with the website layout;
  - draw_map.js – javascript file with the main functions, i.e. the map drawing and update of colours;
  - queries.js – javascript file with the functions to get the information from the triple store;
  - sliders.js – javascript file with the functions needed to visualise and update the map information when the year slider is changed;
  - languages.js – javascript file to support multilingualism. For the moment, it only contains information for the display of information in English;
  - callScript.php – PHP file that executes the python scripts;
  - code_changed.py – python script to query the database in order to get the list of NUTS URIs that changed from one year to other (split, merged and replaced relations). It stores the changes in a JSON file called 'data_year.txt' (one per year);
  - discontinued.py – python script to get the codes that are discontinued from 2013 to 2016. This script is only needed because we do not have the geometries corresponding to 2016 and thus, the 2013 geometries have to be linked with the 2016 codes;
  - list_changes.py – python script to check the changes of the NUTS code from a year, linked to the type of change and value, i.e. URI of the previous NUTS code. It creates a file 'list_changes_year.txt' with the result from the query in JSON;
  - list_props.py – python script to store all the properties and its values for all the NUTS code of a year in the file 'props_year.txt'. This script is created to optimise the user experience. It should be executed if there are changes in the triple store or if new geometries are included;
  - merged.py – python script that stores the merges from a year to the next year (e.g. from 2013 to 2016) in the file 'changes_merged_year.txt'. This script is created to optimise the user experience;
  - replaced.py – python script similar to the merges, but for replacements from a year to the next;
  - split.py – python script similar to the merges, but for splits from a year to the next;
- scripts_spreadsheet – Folder with the scripts needed to generate the triples of new codes from the changes spreadsheet. It consists of:
  - NUTS.py – script to generate the changes triples, looking for keywords like "replaces", "split", etc.;
  - RDF Skeleton – skeleton to get the triples from the spreadsheet with the basic information using Google Refine;

- map.html – Main page of the pilot.

6.2 Deployment

To deploy the pilot locally or in other server, the following list of prerequisites need to be fulfilled:

- For the client side, the pilot HTML needs to be available via a public service (we used xampp in the local implementation). Currently, the pilot is deployed in an Amazon server;
- As technologies, the server needs to have installed python and php;
- For the data storage, the pilot needs to have installed a triple store. Currently, it uses Virtuoso.

To deploy it locally (i.e. having a webserver running on localhost) or in another server, the following steps need to be taken:

1. Download the code from GitHub[12];
2. Extract the code under the public directory on the server; and
3. Execute the scripts to generate the supportive txt files with the JSON variables: code_changed.py, discontinued_2016.py, list_changes.py, list_props.py, merged.py, replaced.py and split.py. All the scripts, except the "discontinued" one, have to be executed as many times as there are years that are to be displayed in the pilot (e.g. having as argument 2010, 2013 and 2016 respectively).

6.3 Adaptability

The pilot can be adapted or extended by adding more geometries, like the ones from 2016, or storing more changes from a year to another received in a spreadsheet.

To add new geometries in GeoJSON, the following steps are needed:

1. Download the file in GeoJSON[13];
2. Create a new file under *datasets* with the declaration of the variable and having as value the GeoJSON data;
3. Include the file in the HTML head;
4. Modify the function "get_selected_NUTS" under "draw_map.js", adding a case for the new year; and
5. Add a new value in the slider. To do so, add the year in the variable "quartersList" in the file "sliders.js".

Note that it is possible that the NUTS geometries contain countries that are not part of Europe, such as Turkey, Norway, Switzerland or Montenegro. In case that you do not want to show them in the visualisation, you need to remove the codes and related coordinates from the countries.

[12] https://github.com/SEMICeu/NUTS_pilot

[13] http://ec.europa.eu/eurostat/statistical-atlas/gis/viewer/distribution/v1/

If new geometries are added, the triple store should be updated with the NUTS codes and the corresponding changes. These changes are published in a spreadsheet[14] by Eurostat. To include them in the triple store, the following steps are needed:

1. Download the changes spreadsheet;
2. Use Google Refine (or any other tool to transform from CSV to RDF) to generate the NUTS codes and the basic properties (label, notation, version…):
    a. Import the spreadsheet;
    b. Remove the first row;
    c. Apply the RDF Skeleton that is included in the GitHub repository, under the 'scripts_spreadsheet' folder;
    d. Review the years that are specified in the RDF Skeleton and edit if needed;
    e. Export the file in RDF;
3. Execute the "NUTS.py" script to generate the changes triples (replaced, merged and split). To do so, it is important to pass the following arguments to the script when executing it (use the order specified below):
    a. Number of lines that there are in the spreadsheet;
    b. Name of the spreadsheet, i.e. the input file to process;
4. Check the changes that are not transformed with the script and create the triples manually. These exceptions are documented in the "NUTS.py" script under the 'scripts_spreadsheet' folder;
5. Consolidate the triples from the three steps performed before into a single file;
6. Upload the consolidated file in the triple store.

6.4 Results

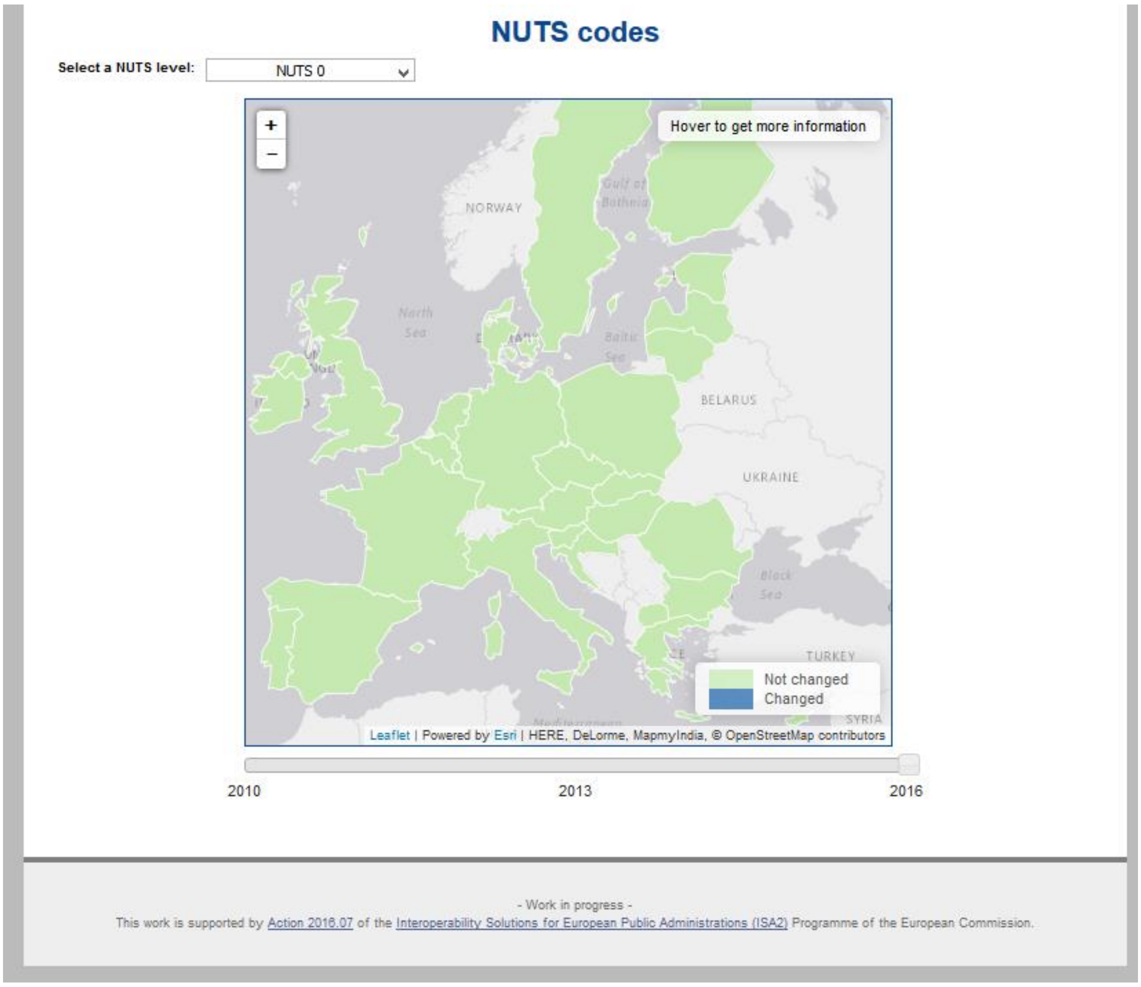
The pilot is publicly available through the following link:
http://52.50.205.146:8890/NUTS_visualisation/map.html

3 main aspects should be noted:

- Map visualisation;
- Selector of NUTS level; and
- Slider to select the year.

The map is the most important part of the visualisation. It has the NUTS regions of the selected year, drawn according to the selected NUTS level.

[14] http://ec.europa.eu/eurostat/web/nuts/history



- Work in progress -

This work is supported by Action 2016.07 of the Interoperability Solutions for European Public Administrations (ISA2) Programme of the European Commission.

Figure 4 – Map with regions of NUTS 2016, level 0

For instance, Figure 4 shows the NUTS regions of 2016, at level 0, i.e. countries.

The top right of the map is updated with the NUTS notation and label of the code that the mouse is hovering over. Finally, a pop-up window appears if the user clicks on a region. It contains the information of the selected code and information about its evolution, in case that it changed.

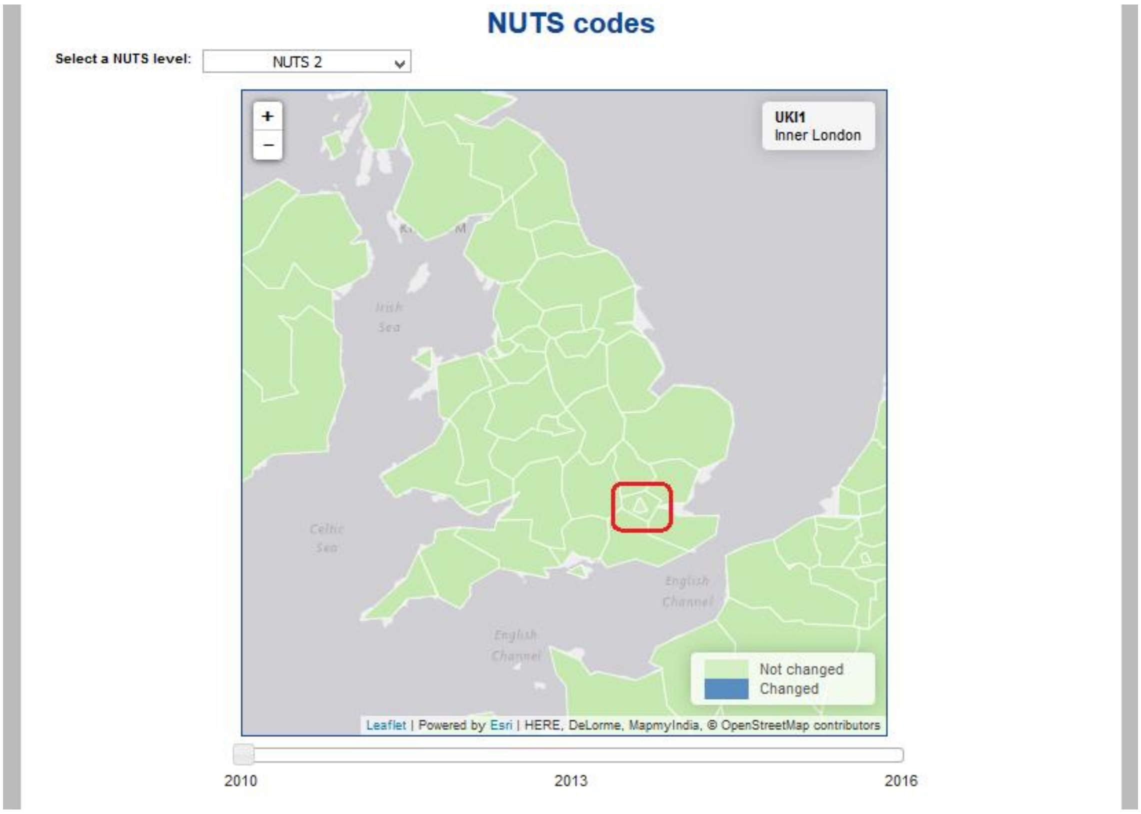
The regions' colours are two:

- Green, meaning that the NUTS code of that region did not change from the previous year to the selected one; or
- Blue, i.e. the NUTS code changed from the previous year to the selected one.

The evolution of NUTS codes is indicated in a changes spreadsheet (see section 6.1, input files), transformed into RDF and stored in a triple store. The changes can be of three types:

- Split from, i.e. a code in the current year was split from a region from the previous year;
- Merged from, i.e. a code in the current year was merged from two or more codes from the previous year; and
- Replaces, i.e. the code from the current year replaces a code from the previous year.

A NUTS code has changed if it is related to another code by, at least, one of the relations mentioned above. For an example, see Figure 5.



NUTS codes

Select a NUTS level: NUTS 2

UKI1
Inner London

Figure 5 – Region UK1 in 2010

The code UKI1, for example, of the NUTS level 2 is a region called Inner London in 2010. However, it is split into two regions, Inner London – West and Inner London – East in 2013 (see Figure 6).

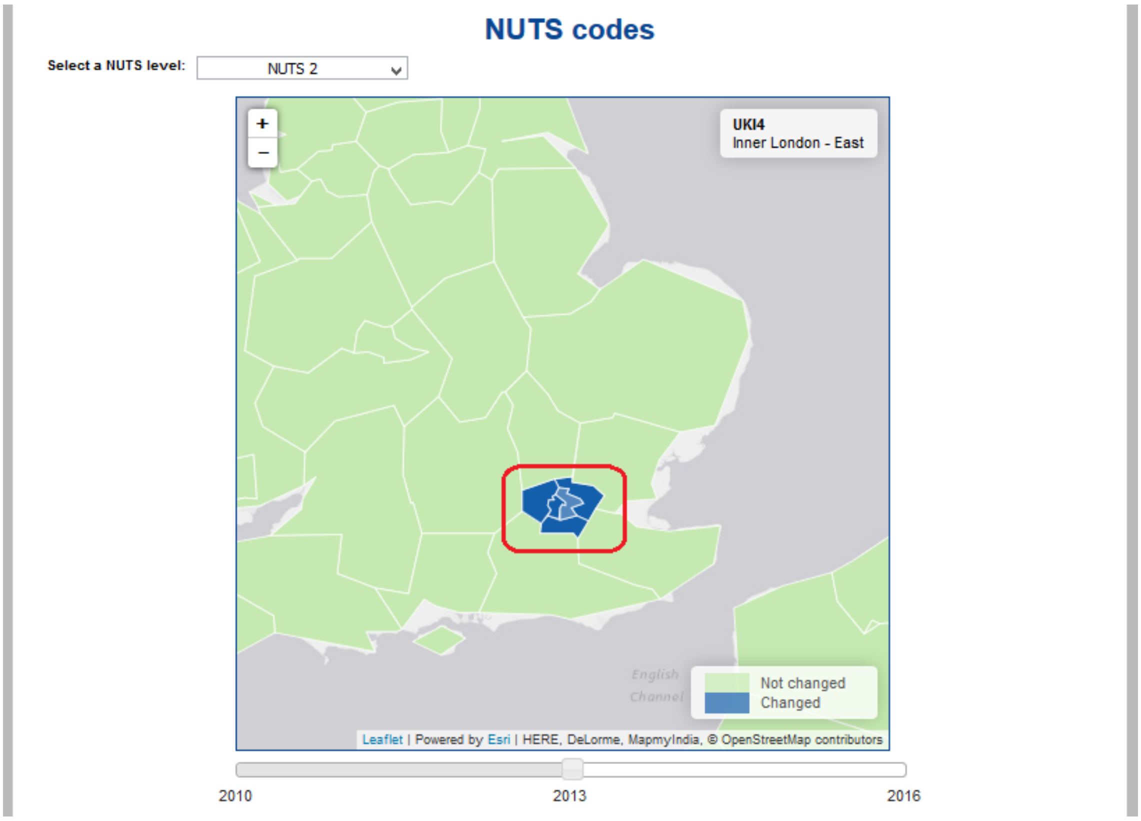
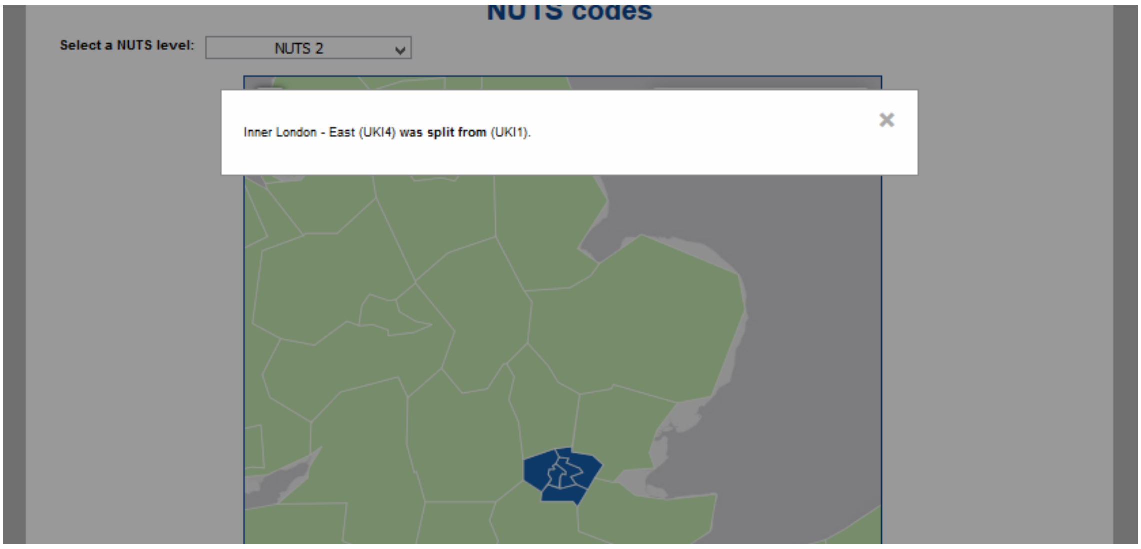


Figure 6 – Region UKI3 and UKI4 in 2013

In addition, a pop-up window is displayed when clicking on any region, with further details about the changes. For example, the change visualised in the figure above is explained in a pop-up when clicking UKI4 (see Figure 7).



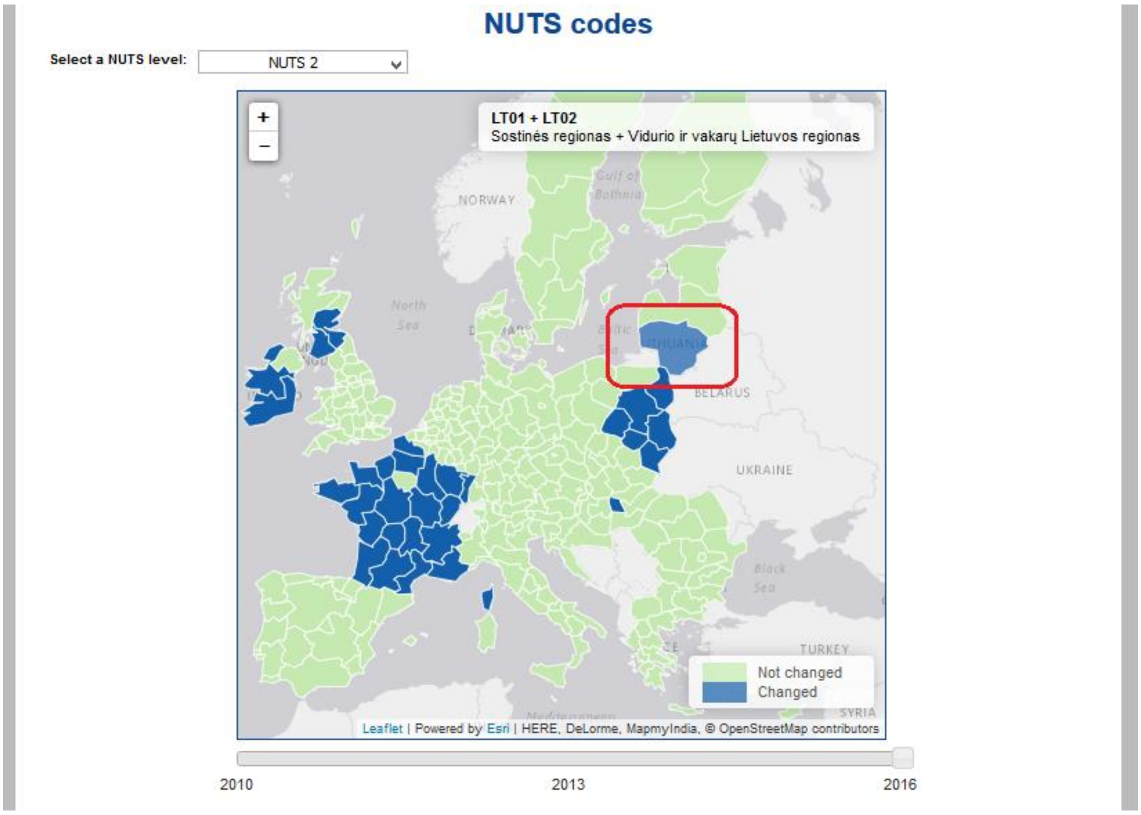
Inner London - East (UKI4) was split from (UKI1).

Figure 7 – Changes of UKI4

Finally, it is important to remark that the geometries for 2016 were not published when the pilot was implemented. Therefore, the geometries shown in 2016 are the same as the 2013.

The regions are coloured by checking the evolution of codes from the previous year to the current one (from 2013 to 2016), contrary to how it is usually done, i.e. from the current year to the previous one. This implies that there are some regions that exist but there is no link between the old and the new code and thus, the code is shown as discontinued from 2013 to 2016.

There are also split regions of which we can reflect the change in the pop-up, but not in the drawn map.



NUTS codes

Select a NUTS level: NUTS 2

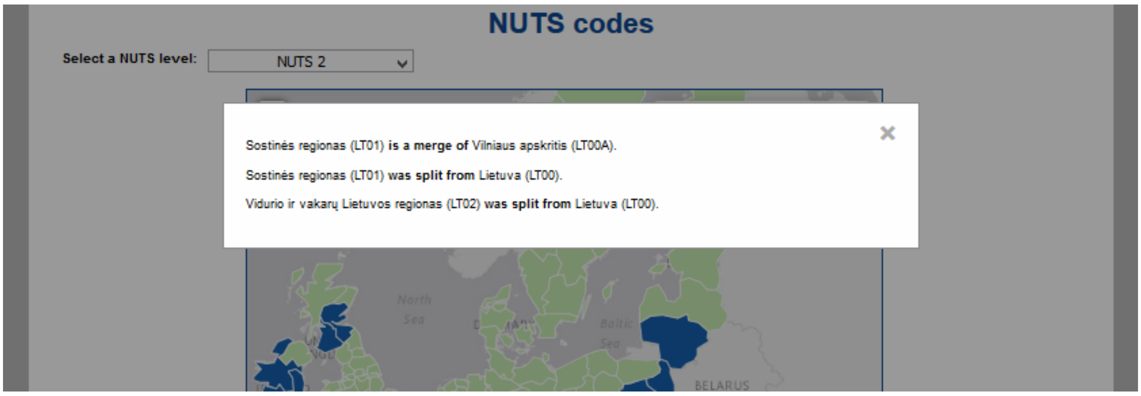
LT01 + LT02
Sostinės regionas + Vidurio ir vakarų Lietuvos regionas

🟩 Not changed
🟦 Changed

Leaflet | Powered by Esri | HERE, DeLorme, MapmyIndia, © OpenStreetMap contributors

Figure 8 – Split region in 2016

For instance, Figure 9 shows a NUTS code level 2 that in 2013 was only one region (Lietuva) and in 2016 was split into two regions ("Sostinės regionas" and "Vidurio ir vakarų Lietuvos regionas"). This is shown in the information box on the top right when hovering on the region, and in the pop-up displayed when clicking on the changed code (see Figure 9).



Sostinės regionas (LT01) **is a merge of** Vilniaus apskritis (LT00A).

Sostinės regionas (LT01) **was split from** Lietuva (LT00).

Vidurio ir vakarų Lietuvos regionas (LT02) **was split from** Lietuva (LT00).

Figure 9 – Pop-up window of split regions

7 CONCLUSIONS AND RECOMMENDATIONS

The current document proposes an RDF model for NUTS, and the approach to provide persistent URIs for NUTS codes, including URI patterns and versioning. It ends with the description of a web application which shows how the publication of NUTS as linked data could be used to visualise on a map changes between NUTS versions.

Eurostat data services could provide more opportunities to make use of the NUTS persistent URIs. More insight could be gained by linking data sets. For example, a map of physicians contrasted with death by disease (circulatory systems) could reveal that a large number of doctors does not mean better treatment, and eventually, linking also healthcare expenditure data would be possible.

To make the NUTS persistent URIs publicly available to users, all NUTS versions should be linked and published in a registry. A namespace should be registered under http://data.europa.eu/ to ensure long term persistence. It is recommended to publish under the same namespace the classes, properties and relationships of the NUTS RDF model, and also the NUTS codes.

The management of the persistent NUTS URIs should be integrated within the existing NUTS management processes as much as possible. In addition, people having roles and responsibilities for NUTS management processes should also be in charge for the management of the persistent NUTS URIs.