



Collaborative Project

## GeoKnow – Making the Web an Exploratory Place for Geospatial Knowledge

Project Number: 318159

Start Date of Project: 01/12/2012

Duration: 36 months

### Deliverable 2.7.1: geodata.gov.gr Geospatial Data as Linked Data

Dissemination Level	Public
Due Date of Deliverable	Month 18, 31/05/2014
Actual Submission Date	Month 19, 04/06/2014
Work Package	WP 2, Semantics-Based Geospatial Information Management
Task	T2.7
Type	Prototype
Approval Status	Final
Version	1.0
Number of Pages	65
Filename	D2.7.1_Geodata.gov.gr_Geospatial_Data_as_Linked_Data.pdf

**Abstract:** This deliverable presents a methodology and a suite of tools for automatically exposing INSPIRE data and metadata on the Semantic Web through GeoSPARQL endpoints, as well as for discovering INSPIRE data through a SPARQL endpoint by repurposing existing INSPIRE catalogue services (CSWs). Further, it reports how such transformations were applied to a real-world open data catalogue and SDI with geographic data for Greece (geodata.gov.gr).

The information in this document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided "as is" without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/ her sole risk and liability.



## History

Version	Date	Reason	Revised by
0.0	30/04/2014	First version and section outline	Kostas Patroumpas
0.1	02/05/2014	First draft with methodology, platform design, and INSPIRE data transformations	Kostas Patroumpas
0.2	06/05/2014	Additions on INSPIRE specifications, data alignment, CSW & metadata handling, and implementation of web interface	Michail Alexakis, Nikos Georgomanolis, Thodoris Stratiotis
0.3	10/05/2014	Added text on Motivation & Perspectives	Spiros Athanasiou
0.4	12/05/2014	First complete (draft) version	Kostas Patroumpas
0.5	20/05/2014	Included Appendix with technical details and query examples	Nikos Georgomanolis, Kostas Patroumpas
0.6	22/05/2014	Restructuring, various edits and improvements throughout the document	Spiros Athanasiou, Kostas Patroumpas
0.7	23/05/2014	Interim version prepared for the project's peer-review	Kostas Patroumpas
0.8	04/06/2014	Improvements based on internal review comments	Spiros Athanasiou, Kostas Patroumpas
0.9	04/06/2014	Proof reading, formatting improvements	Kostas Patroumpas
1.0	04/06/2014	Final version	Kostas Patroumpas

## Author List

Organisation	Name	Contact Information
ATHENA	Michail Alexakis	alexakis@imis.athena-innovation.gr
ATHENA	Spiros Athanasiou	spathan@imis.athena-innovation.gr
ATHENA	Nikos Georgomanolis	ngeorgomanolis@imis.athena-innovation.gr
ATHENA	Kostas Patroumpas	kpatro@imis.athena-innovation.gr
ATHENA	Thodoris Stratiotis	stratiot@imis.athena-innovation.gr

# Executive Summary

The INSPIRE Directive by the European Commission sets the legal and technical foundations towards interoperable Spatial Data Infrastructures (SDIs) across Europe. EU member states are already providing such services for several geospatial data themes (e.g., transportation networks, administrative units). Unfortunately, the INSPIRE ecosystem is currently disjoint from the Semantic Web, without any means to repurpose existing SDIs as high-quality data sources, and thus multiply their value through interlinking, reasoning and inferencing.

In this Deliverable, we present a methodology that can assist stakeholders into automatically exposing INSPIRE SDIs on the Semantic Web. First, we developed methods for discovering INSPIRE data through a SPARQL endpoint, by repurposing the existing INSPIRE catalogue services. Further, we implemented a suite of tools for automatically transforming INSPIRE data and metadata into spatial linked data as RDF triples. The compiled geographic and thematic information can then be loaded into semantic repositories and queried through SPARQL statements or interlinked with other data. Our open-source solutions essentially repurpose existing INSPIRE SDIs, so as to promote uptake and facilitate their reuse in practice.

Our open-source software tools, either enhanced (TripleGeo) or developed (TripleGeo-CSW) specifically for this task, are publicly available for use by any interested party. In addition, we have set up a web interface with several SPARQL endpoints, available at

<http://geodata.gov.gr/sparql/>

This service allows users to retrieve data and metadata for Greece for several thematic categories (e.g., administrative units, protected sites), and also discover available datasets across Europe according to user-specified criteria.

The layout of this document is as follows:

Section 1 sets the framework of our approach with respect to exposing SDIs on the Semantic Web.

In Section 2, we discuss the challenges for exposing INSPIRE as linked data as well as some recent approaches.

In Section 3, we analyse how INSPIRE-compliant metadata on spatial data can be translated into RDF.

In Section 4, we present a middleware that can be used to discover INSPIRE metadata from catalogue services in the Web through a virtual SPARQL endpoint.

The transformation process and RDF modelling issues on INSPIRE data are examined in Section 5.

Section 6 reports our experience from applying such transformations to a real-world SDI with publicly available data for Greece in order to expose its contents through (Geo)SPARQL endpoints.

Section 7 offers conclusions and future extensions both from a research and a practical point of view.

Finally, in the Appendix, we include implementation details, excerpts from the source code used in the transformations, instructions for users, various indicative queries, as well as a few details about the datasets utilized in our use case scenario.

## Abbreviations and Acronyms

<b>CSW</b>	Catalogue Services for the Web
<b>CRS</b>	Coordinate Reference System
<b>DBMS</b>	DataBase Management System
<b>DCAT</b>	Data Catalogue Vocabulary
<b>EU</b>	European Union
<b>GML</b>	Geography Markup Language
<b>GIS</b>	Geographical Information Systems
<b>ICT</b>	Information and Communications Technology
<b>INSPIRE</b>	INfrastructure for SPatial InfoRmation in Europe
<b>ISO</b>	International Organization for Standardization
<b>JRC</b>	Joint Research Centre of the European Commission
<b>KML</b>	Keyhole Markup Language
<b>LOD</b>	Linked Open Data
<b>MBR</b>	Minimum Bounding Rectangle
<b>OGC</b>	Open Geospatial Consortium ( <a href="http://www.opengeospatial.org/">http://www.opengeospatial.org/</a> )
<b>OSM</b>	OpenStreetMap ( <a href="http://www.openstreetmap.org/">http://www.openstreetmap.org/</a> )
<b>OWL</b>	OWL 2 Web Ontology Language
<b>RDF</b>	Resource Description Framework
<b>RDFS</b>	Resource Description Framework Schema
<b>SDI</b>	Spatial Data Infrastructure
<b>SPARQL</b>	SPARQL Protocol and RDF Query Language
<b>SQL</b>	Structured Query Language
<b>SRID</b>	Spatial Reference system IDentifier
<b>SKOS</b>	W3C Simple Knowledge Organization System
<b>UML</b>	Unified Modeling Language ( <a href="http://www.uml.org/">http://www.uml.org/</a> )
<b>URI</b>	Uniform Resource Identifier
<b>URL</b>	Uniform Resource Locator
<b>WFS</b>	Web Feature Service Interface (OGC Standard)
<b>WKT</b>	Well Known Text (as defined by ISO 19125)
<b>WMS</b>	Web Map Service (OGC Standard)
<b>WGS84</b>	World Geodetic System (EPSG:4326)
<b>W3C</b>	World Wide Web Consortium ( <a href="http://www.w3.org/">http://www.w3.org/</a> )
<b>XSL</b>	EXtensible Stylesheet Language
<b>XML</b>	Extensible Markup Language

# Table of Contents

<b>1. INTRODUCTION</b>	<b>7</b>
1.1 OBJECTIVES OF THIS TASK	8
1.2 RELATION TO THE LINKED DATA LIFECYCLE	8
<b>2. BACKGROUND</b>	<b>10</b>
2.1 THE INSPIRE DIRECTIVE	10
2.2 INSPIRE SDIS AS LINKED DATA SOURCES	11
<b>3. TRANSFORMING INSPIRE METADATA INTO RDF</b>	<b>12</b>
<b>4. DISCOVERING INSPIRE THROUGH CATALOGUE SERVICES</b>	<b>15</b>
4.1 OGC CATALOGUE SERVICES FOR THE WEB (CSW)	15
4.2 IMPLEMENTATION OF THE TRIPLEGEO-CSW MIDDLEWARE	15
<b>5. TRANSFORMING INSPIRE DATA INTO RDF</b>	<b>19</b>
5.1 SPECIFICATION OF URIS	19
5.2 ASSIGNING INSPIRE IDENTIFIERS	20
5.3 GEOMETRY SERIALIZATIONS	21
5.4 CONVERTING THEMATIC ATTRIBUTES	21
<b>6. EXPOSING INSPIRE SDI AS RDF: A USE CASE IN GREECE</b>	<b>24</b>
6.1 GEODATA.GOV.GR: A SDI FOR GREECE	24
6.2 DATA ALIGNMENT TO INSPIRE ANNEX I	25
6.3 TRANSFORMING FEATURES FROM GEODATA.GOV.GR INTO RDF	26
6.4 ENABLING QUERIES THROUGH (GEO)SPARQL ENDPOINTS	27
<b>7. CONCLUSIONS AND PERSPECTIVES</b>	<b>30</b>
<b>8. REFERENCES</b>	<b>31</b>
<b>9. APPENDIX</b>	<b>35</b>
9.1 TRANSFORMING INSPIRE DATA AND METADATA INTO RDF	35
9.1.1 Data Alignment to INSPIRE schema	35
9.1.2 Transforming INSPIRE Metadata into RDF	36
9.1.2.1 Execution of the XSLT Transformation	41
9.1.3 Implementation of the TripleGeo-CSW Middleware	42
9.1.3.1 Parsing of SPARQL Queries	42
9.1.3.2 Formulating CSW Requests	44
9.1.3.3 Executing the TripleGeo-CSW Middleware	45
9.1.4 Transforming INSPIRE Data into RDF	45
9.1.4.1 Execution of the XSLT Transformation	54
9.2 (GEO)SPARQL ENDPOINTS FOR INSPIRE-ALIGNED DATA AND METADATA	55

9.2.1	Implementation of SPARQL Endpoints .....	55
9.2.2	Using (Geo)SPARQL Endpoints for INSPIRE Metadata .....	56
9.2.3	Using (Geo)SPARQL Endpoints for INSPIRE Data .....	58
9.2.4	Using a Virtual SPARQL Endpoint for Discovering INSPIRE CSWs .....	63
9.3	LICENSING .....	64

## List of Figures

Figure 1: The Linked Data Lifecycle.....	9
Figure 2: Excerpt of XSL script for transforming INSPIRE metadata elements into RDF.....	13
Figure 3: Flow diagram for processing SPARQL queries in the TripleGeo-CSW middleware..	16
Figure 4: Example GeoSPARQL query against INSPIRE metadata.....	17
Figure 5: An example GetRecords request submitted to INSPIRE-compliant CSWs.....	18
Figure 6: A geographic name (in Greek) as expressed in INSPIRE-compliant RDF/XML. ....	22
Figure 7: Excerpt of HALE alignment to INSPIRE for geometries of Greek administrative units. ....	26
Figure 8: Web interface for submitting (Geo)SPARQL queries against five backend engines.	28
Figure 9: Processing (Geo)SPARQL queries through the web interface. ....	29
Figure 10: XSL stylesheet Metadata2RDF.xsl for transforming INSPIRE metadata into RDF. ....	41
Figure 11: XSL stylesheet Inspire_Main.xsl for transforming INSPIRE data into RDF. ....	48
Figure 12: XSL stylesheet GML2WKT.xsl for transforming geometries from GML into WKT. ....	54

## List of Tables

Table 1: RDF mappings for indicative INSPIRE metadata elements.....	12
Table 2: Datasets from geodata.gov.gr exposed as INSPIRE-aligned RDF data.....	24
Table 3: Target (Geo)SPARQL endpoints set up for the use case scenario.....	27
Table 4: HALE alignments to INSPIRE themes available for datasets from geodata.gov.gr. ....	35
Table 5: Mapping pairs of elements to particular values. ....	43
Table 6: Elements supported in a CSW GetRecords request. ....	44
Table 7: XSL stylesheets developed for INSPIRE Data Themes of Annex I.....	45
Table 8: Auxiliary XSL stylesheets.....	46

# 1. Introduction

Multiple products, systems, and organizations are dependent on immediate access to accurate and official geospatial information. This need is addressed mainly through *Spatial Data Infrastructures* (SDIs). The term SDI [GSDI] was coined in 1993 to denote a framework of technologies, policies, and institutional arrangements that together facilitate the creation, exchange, and use of geospatial data and related information resources across an information-sharing community. SDIs are essentially interoperability frameworks, and as such, research and development in this field is closely tied to standardization activities lead by international bodies, namely the ISO/TC 211 [ISO211], OGC [OGC], and W3C [W3C]. The necessity for uniformity and standardization among the industry, practitioners and academia, was expressed more than 20 years ago in a United Nations World Summit on Sustainable Development. Therefore, SDIs have a simple, yet challenging goal: *make geospatial data easier to discover and use*. SDIs are mature systems, developed after decades of research and development, based on common standards, and with de facto support across GIS platforms and services.

In EU, the INSPIRE Directive 2007/2/EC [INSPIRE-Directive] standardizes SDIs across member states, on a legislative, institutional and technical level. As such, INSPIRE (*IN*frastructure for *SP*atial *Info*Rmation in *EU*rope) will enable the discovery, download, and visualization of geospatial information across the EU in a common, cross-boundary manner. Several member states already offer INSPIRE-compliant metadata, data, and services. According to the INSPIRE roadmap, harmonization and full compliance of all metadata, data and services, is built on specific deadlines and milestones [INSPIRE-Roadmap]. A pan-European SDI is slowly formed based on member state SDIs. Thus, in the following years, every aspect of geospatial information will be available through the INSPIRE network of services, following specific standards for geospatial information.

SDIs in general, and INSPIRE-compliant SDIs in particular, will gradually expose through standardized services the complete wealth of official geospatial data produced, curated, and applied by the public sector. The GIS community, including researchers and professionals, are provided with significant opportunities to reuse this data, across diverse domains and applications. The emerging alignment of INSPIRE data licensing policies towards open knowledge, further broadens the potential for value added services. The potential volume, quality, and wealth of this body of geospatial knowledge can be a game-changer for industry, research and governance.

Unfortunately, the INSPIRE ecosystem is currently disjoint from the Semantic Web, without any means to introduce INSPIRE data as high-quality linked data sources. Such an omission is intentional and completely understandable. INSPIRE SDIs are based on slowly changing and widely established standards, adopted through careful consultation. These slow processes are required to establish a political and technological consensus given the extreme investment required to implement an SDI. Beyond the strict costs of new ICT systems, one should take into account the associated complexities and effort for an intertwined change across stakeholders, organizations, legal foundations, and the data life-cycle. As such, INSPIRE SDIs should be considered as almost diachronic, extremely slowly changing, but powerful resources for geospatial knowledge.

The fact that Semantic Web technologies and Linked Data are not integrated into INSPIRE SDIs is therefore simply a matter of unfortunate timing. The rise of Semantic Web followed the period when the INSPIRE technical specifications were prepared, discussed and finalized. Hence, establishing this "*missing link*" between INSPIRE and the Semantic Web is very challenging. We believe that it is possible to repurpose existing INSPIRE SDIs in the Semantic Web, without affecting their established operation (e.g., applications) and thus incurring additional costs. This will



be significant for the Semantic Web community, given the de facto access to high quality geospatial data, but also a testament for the sound decision making process of the INSPIRE stakeholders regarding forward-compatibility.

## 1.1 Objectives of this Task

In this deliverable, our goal is to devise effective methods for automatically exposing INSPIRE SDI metadata, data, and services as Linked Data according to Resource Description Framework [RDF], so as to maximize potential impact and practical applications. INSPIRE data semantics refer to particular domains (e.g., transport, addresses, and hydrology) and attempt to cover many alternative representations of facts in the real world (e.g., addresses could be specified either by a single locator or by combining a thoroughfare, a street number, and a postal code). However, INSPIRE data is also intended for use in other domains as well, e.g., a road network may be helpful in nature preservation studies. This potential is exactly one of the major benefits that come from publishing this geospatial information as linked data. Thanks to RDF vocabularies, such geo-semantic data could be interlinked with third-party features (e.g., traffic, population statistics) and enriched with value-added information. Moreover, as government agencies and public authorities are actually the owners of INSPIRE-aligned data, this content may serve as a common reference point across Europe for applications, studies, measurements, derived datasets, etc.

Still, only a small amount of such information has been published as *linked geospatial data* and associated with other resources in the Semantic Web. Apart from the inherent complexity of geospatial concepts, the lack of standardization made this task quite challenging. However, the recent OGC GeoSPARQL standard [OGC12] proposes structures for storing geometric RDF, querying them through a SPARQL extension [SPARQL] equipped with a variety of spatial operations [BK12], as well as a foundation to support spatial reasoning on LOD. We regard this development as a great opportunity to publish INSPIRE data encoded and queried through GeoSPARQL.

To the best of our knowledge, ours is the first attempt to build an abstraction layer on top of the INSPIRE infrastructure based on GeoSPARQL concepts, thus making INSPIRE contents accessible and discoverable as linked data. As we analyze next, we had to address several issues not strictly related to geometries, but ranging from property alignments to assignment of URIs and multi-lingual support. As a solid proof-of-concept, we apply this methodology on data and metadata from an SDI in Greece, and we set up endpoints for querying and discovering such features. What's more, we develop a platform that can act as a broker between a SPARQL endpoint and existing *Catalogue Services for the Web* (CSW), in order to enable on-the-fly discovery of third-party INSPIRE metadata via Semantic Web technologies.

Overall, we believe that our approach provides a technical roadmap towards bridging the gap between the INSPIRE and Linked Data ecosystems. It relies on existing INSPIRE SDIs, requiring no modifications for the SDI, and no extra work on behalf of the SDI owner. Even an external stakeholder could simply deploy our software and provide full GeoSPARQL access to INSPIRE data across the EU. Towards this goal, we provide all of our software with an open source license.

## 1.2 Relation to the Linked Data Lifecycle

GeoKnow covers the whole lifecycle of spatial data, as illustrated in Figure 1. In this particular task, we suggest methodologies and we develop tools that mostly relate to two particular phases of the Linked Data Lifecycle:

- *Search/Browsing/Exploration*: We offer a tool (TripleGeo-CSW) and a web interface (available through <http://geodata.gov.gr/sparql>), which allow users to explore the quantity and quality of INSPIRE-compliant datasets available from Catalogue Services across Europe. We have also set up several SPARQL endpoints with geospatial support,



where we have loaded several data and metadata from a real-world SDI in Greece. Indicative queries specified in GeoSPARQL are also available to users.

- *Extraction:* We provide a tool (TripleGeo) along with several scripts (XSL stylesheets, INSPIRE RDF mappings) for translating INSPIRE data and metadata into RDF representations. We also outline several recommendations for INSPIRE alignment and best practices for such transformations over a wide range of spatial datasets (INSPIRE Data Themes Annex I).

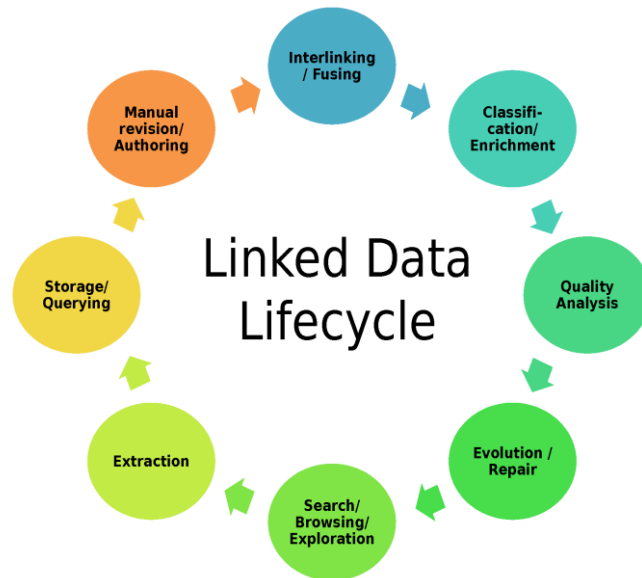


Figure 1: The Linked Data Lifecycle.

## 2. Background

### 2.1 The INSPIRE Directive

INSPIRE aims to set the framework for a European Union (EU) Spatial Data Infrastructure (SDI), so as to facilitate sharing of environmental spatial information between European public authorities under a unified technical and institutional basis. The respective **European Directive 2007/2/EC** [INSPIRE-Directive] was passed by the European Parliament on 14 March 2007 and came into force on 15 May 2007. According to the formal roadmap set by the European Commission for INSPIRE compliance [INSPIRE-Roadmap], its implementation is taking place in various stages and its full implementation is expected by 2020.

In order to support these goals, INSPIRE specifies several *Implementing Rules* [INSPIRE-Rules] to ensure that SDIs in EU member states are compatible and usable in a transboundary context. These foundations include technical interoperability standards for geospatial metadata, data and online services, as well as uniform legal rules for data sharing between public authorities. Having such a pan-European SDI infrastructure established, it is expected that public access to spatial data will be eased across Europe permitting data reuse. In addition, formulation, implementation, management and assessment of EU environmental policies will be significantly facilitated.

According to the Directive, public authorities and organizations should provide their spatial data according to the INSPIRE specifications. This includes catalogues of available resources using metadata, common access policies and standards, as well as network services for discovery, viewing, downloading, transformation, etc. These datasets relate to 34 themes [INSPIRE-themes] with a territorial dimension or environmental impact. Some themes are used to represent entities that describe the environment like lakes, land cover, or habitats. Other themes describe things that may have environment impact, like transport networks or buildings. Yet another category refers to entities that assist in environmental management, like administrative or statistical units. These data themes are organized into three stages according to Annexes to the Directive that specify the date of their release, with full harmonization and publishing required by 2020.

In this work, we deal with the first group of such themes, as specified in **Annex I**:

- **[RS] Coordinate reference systems:** Systems for uniquely referencing spatial information in space as a set of coordinates (x,y,z) and/or latitude, longitude and altitude, based on a geodetic datum.
- **[GG] Geographical grid systems:** Harmonised multi-resolution grid with a common point of origin and standardised location and size of grid cells.
- **[GN] Geographical names:** Names of areas, regions, localities, cities, suburbs, towns or settlements, or any geographical or topographical feature of public or historical interest.
- **[AU] Administrative units:** Units of administration, dividing areas where Member States have and/or exercise jurisdictional rights, for local, regional and national governance, separated by administrative boundaries.
- **[AD] Addresses:** Location of properties based on address identifiers, usually by road name, house number, postal code usually specified by road name, house number, and postal code.
- **[CP] Cadastral parcels:** Areas defined by cadastral registers or equivalent land parcel, under homogeneous real property rights and unique ownership.
- **[TN] Transport networks:** Road, rail, air and water transport networks and related infrastructure, including links between different networks.
- **[HY] Hydrography:** Rivers, lakes, lagoons, including marine areas and all other water bodies and items related to them, including river basins and sub-basins.

- **[PS] Protected sites:** Areas with certain protection targets, referring to conservation of nature, fishing or forest resources, cultural heritage objects or areas, etc.

Two other annexes specify more data themes for inclusion in INSPIRE, such as:

- *elevation, land cover, geology, orthoimagery* (**Annex II**),
- as well as *statistical units, soil, land use, natural risk zones, atmospheric conditions, habitats and biotopes*, and many more (**Annex III**).

Several member states have begun publishing harmonized datasets for Annex I, so we expect that transforming such data into RDF by the methodology proposed in this deliverable would be straightforward with minimal customizations (e.g., specifications for base URIs or language literals), as explained later on.

## 2.2 INSPIRE SDIs as Linked Data Sources

At present, no complete INSPIRE ontology in RDF/OWL exists. This reflects the difficulty of bridging the "*closed world*" assumption of UML models in INSPIRE with the "*open world*" view of RDF. However, this limitation refers not only to INSPIRE, since exposing geospatial information as open linked data is a relatively new research topic. In [AO13] a Reference Architecture for building interoperable, linked SDIs has been presented, which was the result of analyzing existing SDI architectures. Especially for INSPIRE SDIs, some prominent opportunities of utilizing linked open data have been highlighted by the Joint Research Center of European Commission in [SL10], along with the requirements for achieving it.

With regard to the particular task of exposing metadata as open linked metadata, the *crosswalking* approach is suggested in [RWB12],[LFN+10]. *Metadata crosswalking* implies mappings from popular geospatial metadata schemas to the Dublin Core vocabulary, addition of extra metadata elements and expressing the metadata terms as RDF. The authors in [RWB12] also suggest an alternative method for publishing geospatial metadata provisioned by custom catalogue services as open linked metadata. In this case RDF metadata terms are published directly from the UML representation of the underlying custom schemas.

Exposing geospatial data as linked data appears to be a more challenging task. However, there are notable studies devoted to this goal. In [TSS11] a general approach is given on translating INSPIRE-compliant GML data models as semantic OWL ontologies. It takes into account common characteristics of GML and RDF/OWL and INSPIRE UML notations to produce the conversion rules for general and specific-type elements. Furthermore, a method of translating semantic SPARQL queries to corresponding OGC WFS/Filter queries is described. This mapping makes possible to semantically wrap standard *INSPIRE download services* exposing them as SPARQL endpoints. In [VVS+10] a specific methodology was devised, aiming at analyzing heterogeneous Spanish geospatial and non-geospatial datasets for three INSPIRE themes (Administrative units, Hydrography, and Statistical units), and then exposing them through <http://geo.linkeddata.es/>. This methodology entails generation of an ontology model mixing a number of different existing ontologies and vocabularies, extraction of RDF data using a variety of software tools, and finally interlinking by matching similarity strings between URIs. Most recently, the approach in [BJQ14] focuses on deriving linked data from GML data and also examines whether more meaningful RDF can be created from GML, by reusing existing concepts from vocabularies.

In contrast to the aforementioned approaches, there has not been any attempt to expose INSPIRE metadata and data according to the GeoSPARQL specification, as we present next. Our suite of tools for the Semantic Web can not only be used by data owners in order to make their SDI contents accessible in RDF, but also for discovering third-party data via SPARQL requests against Catalogue Services in the Web.

### 3. Transforming INSPIRE Metadata into RDF

There have been two main (and often complementary) approaches to cataloguing linked metadata. Data Catalogue Vocabulary (DCAT) [DCAT] is an RDF vocabulary designed to facilitate interoperability between data catalogues published on the Web. The VoID Vocabulary (VoID) [VoID] makes use of an RDF Schema vocabulary to express metadata about RDF datasets, and aims at data discovery, cataloging and archiving. However, it should be noted that both approaches make extensive use of terms from other vocabularies, in particular Dublin Core [DC11].

Based on similar vocabularies, a few initiatives and case studies headed towards linked metadata on spatial datasets. Among them, the Mimas Linked Data Project for LandMap Spatial Discovery in the UK has made some preliminary work [Mimas], mostly by identifying vocabularies and defining RDF mappings for a subset of their datasets. An open source prototype for Data Catalogue Vocabulary services based on DCAT is being implemented in GeoNetwork, and would provide support to harvest, search and link catalogue contents with other interlinked resources [GeoNetDCAT]. Public authorities have also published spatial metadata and provide SPARQL endpoints, such as the municipality of Zaragoza in Spain [ZarSPARQL].

INSPIRE Implementing Rules [INSPIRE-Rules] describe a schema for INSPIRE-compliant metadata according to ISO standards. ISO-19115 [ISO19115] provides an extensive schema for describing geospatial information, applicable to single datasets, series of datasets or even individual geographic features and feature properties. It contains metadata elements for data regarding its identification, topic, quality, geographical and temporal extent, as well as points of contact with the responsible parties. In addition, ISO-19119 [ISO19119] defines a framework for developing services that can be used to access and process geospatial data. This framework supports access to different data sources through a generic, platform-neutral application interface. INSPIRE metadata should not violate these ISO standards, but since the latter require many more elements (e.g., points of contact, restrictions) these have to be provisioned as well. On the other hand, metadata published according to ISO-19115 core is not guaranteed to conform with the INSPIRE ontology, so an alignment is necessary.

INSPIRE metadata is being used extensively to describe datasets, dataset series, services and thematic layers across Europe. Creating metadata is facilitated by the online INSPIRE metadata editor [INSPIRE-editor], but this facility is not yet enriched with Semantic Web technologies that would enable: exposing through SPARQL endpoints, linking with other data (e.g., DBpedia [DBpedia], GeoNames [GeoNames]), reasoning, etc. The EU Open Data Portal [EU-open-data] has introduced a vocabulary for metadata using DCAT, and provides it both as worksheet specification and as an ontology. Although it has been aligned with a standardised metadata vocabulary (ADMS) created by the EU's Interoperability Solutions for European Public Administrations (ISA) Programme [ISA], this vocabulary is not yet aligned with INSPIRE, so the portal does not currently offer any spatial metadata.

**Table 1: RDF mappings for indicative INSPIRE metadata elements.**

INSPIRE metadata element	RDF mapping of attribute
Resource title	dct:title
Resource language	dct:language
Keyword	dcat:keyword

Geographic Bounding Box	dct:spatial
Responsible organization - Owner	dct:rightsHolder

An exploratory investigation is under way by the Joint Research Centre (JRC) of the European Commission towards using Semantic Web technologies for geospatial metadata [Per12]. Currently, a preliminary version of RDF mappings for INSPIRE metadata elements has been defined [Per14]. A few indicative examples of these RDF mappings to vocabularies such as DCAT, DCT, SKOS, vCard, etc., are shown in Table 1. In the upcoming years, the JRC plans a LOD-enabled INSPIRE prototype, by making a corpus of RDF-encoded metadata available through the INSPIRE Geoportal [INSPIRE-GeoPortal], and queryable via a SPARQL endpoint. Currently, this geoportal offers an editor, a validator, and a viewer, but does not support RDF features.

Although still informal and subject to further modifications by the JRC, their draft on possible RDF mappings [Per14] has offered us a valuable basis to develop a methodology for transforming INSPIRE metadata elements into RDF. Our goal is to provide open-source tools that facilitate such transformation in order to load the results into triple stores and make them accessible through SPARQL. We are mostly interested in exposing the spatial coverage of data, as well as the temporal range of their lifecycle (i.e., when data was published or modified). However, many more elements are important as well, such as descriptions (e.g., title, abstract, subject, keywords), content assessments (like quality, provenance, or conformity), as well as their legal status (owner, license, point of contact, etc.). As this metadata is exposed on the Semantic Web, it may be potentially interlinked with other features, such as terms in code lists or multilingual thesauri.

```
<dct:conformsTo>
  <dct:Standard>
    <dct:title xml:lang='en'>
      <xsl:value-of select='//gmd:report//gmd:title/gco:CharacterString' />
    </dct:title>
    <dct:issued rdf:datatype='http://www.w3.org/2001/XMLSchema#date'>
      <xsl:value-of select='//gmd:report//gco:Date' />
    </dct:issued>
  </dct:Standard>
</dct:conformsTo>
```

**Figure 2: Excerpt of XSL script for transforming INSPIRE metadata elements into RDF.**

In practice, we created an application profile for such metadata as a set of templates used in XSLT transformation. Once invoked, our custom XSL stylesheet accepts an XML file with INSPIRE-compliant metadata records and maps them into suitable RDF statements according to the mapping. The result is an RDF/XML representation of INSPIRE-aligned metadata records, which can be readily loaded into a triple store. The XSL stylesheet is generic, covers all elements, and can be reused against any metadata conforming to INSPIRE specifications. The excerpt shown in Figure 2 refers to handling of elements related with dataset conformity. The entire XSL stylesheet is listed in the Appendix and it is also available from [INSPIRE2RDF]. Regarding the geographical coverage, its bounding box can be suitably expressed either as a GeoSPARQL polygon or a 2-dimensional rectangle BOX2D, depending on the target RDF store and its geospatial support.

Our design adheres to reusing existing URIs as much as possible, especially in statements concerning spatial, temporal, and identification elements. However, blank nodes exist in the resulting triples, since the RDF mappings are based on the abstract schema of metadata elements.

These blank nodes are used as locally-scoped artifacts, which need not be explicitly labelled. As argued in [MAHP11], a Skolemization scheme could be applied to remedy this weakness. Nevertheless, provided that the user is aware of the underlying schema, no particular problems will arise in formulating SPARQL queries against such metadata, as we have verified in our use-case scenario (Section 6).



## 4. Discovering INSPIRE through Catalogue Services

### 4.1 OGC Catalogue Services for the Web (CSW)

Catalogue Services for the Web (CSW) is an OGC standard [CSW] that describes application profiles for publishing, accessing, and searching over collections of metadata on geospatial data, services, and related resources. This metadata must be encoded in XML and the schema of its records is usually conformant to more specific standards (like ISO 19139, Dublin Core, or INSPIRE).

Users may submit a number of different HTTP requests to a CSW and the response is encoded in an XML document as well. Among the most typical requests that must be supported by a CSW, are:

- `GetCapabilities` can be used to retrieve metadata describing the type of requests the CSW service can accept.
- A `DescribeRecord` request returns a description of the metadata records' model.
- A `GetRecords` request retrieves actual metadata records which satisfy certain criteria and filters specified by the request.
- Issuing `GetRecordsById` returns specific records matching certain identifiers given as parameters.

Several other requests are non-mandatory for CSW mechanisms. Among them:

- `GetDomain` returns the range of values of a metadata record field or a request parameter.
- `Transaction` can be used to create new metadata records, and edit or delete existing ones.
- `Harvest` pulls metadata from third-party sources to create new records or to update existing ones.

All aforementioned requests can be submitted either using GET or POST HTTP methods.

### 4.2 Implementation of the TripleGeo-CSW Middleware

CSW services for INSPIRE-compliant metadata have already been available in various European countries, even in non-EU member states like Norway. Our work is focused on exposing such CSWs through a *CSW-to-RDF middleware* (nicknamed **TripleGeo-CSW Middleware**), which acts as a virtual SPARQL endpoint and can discover metadata from existing catalogues. This middleware undertakes to request any related information from several CSWs, collect the partial results, and finally return any qualifying metadata as RDF triples.

Towards this goal, we have made use of metadata from a list of CSW services. These include:

1. The INSPIRE Discovery service in the Czech Republic (<http://micka.cenia.cz/metadata/csw/index.php>).
2. The Estonian National Geoportal (<http://inspire.maaamet.ee/geoportal/csw>).
3. The Irish Spatial Data Exchange (<http://catalogue.isde.ie/geonetwork/srv/en/csw>).
4. The Spanish National Geographic Institute (<http://www.ign.es/csw-inspire/srv/eng/csw>).
5. The Metadata Catalogue of the SDI for Spain (<http://www.idee.es/csw-inspire-idee/srv/eng/csw>).
6. The Discovery Service for the UK Location catalogue (<http://csw.data.gov.uk/geonetwork/srv/en/csw>).

## 7. The National CSW for Norway (<http://www.geonorge.no/geonetwork/srv/nor/csw-inspire>).

We stress that this is just an indicative list of currently operating CSWs. Of course, this list may be extended as more INSPIRE-compliant such services become available, without necessitating absolutely any change in our existing framework. It only requires adding any additional CSW into the list of such services, i.e., editing the respective file that is accessible by the middleware.

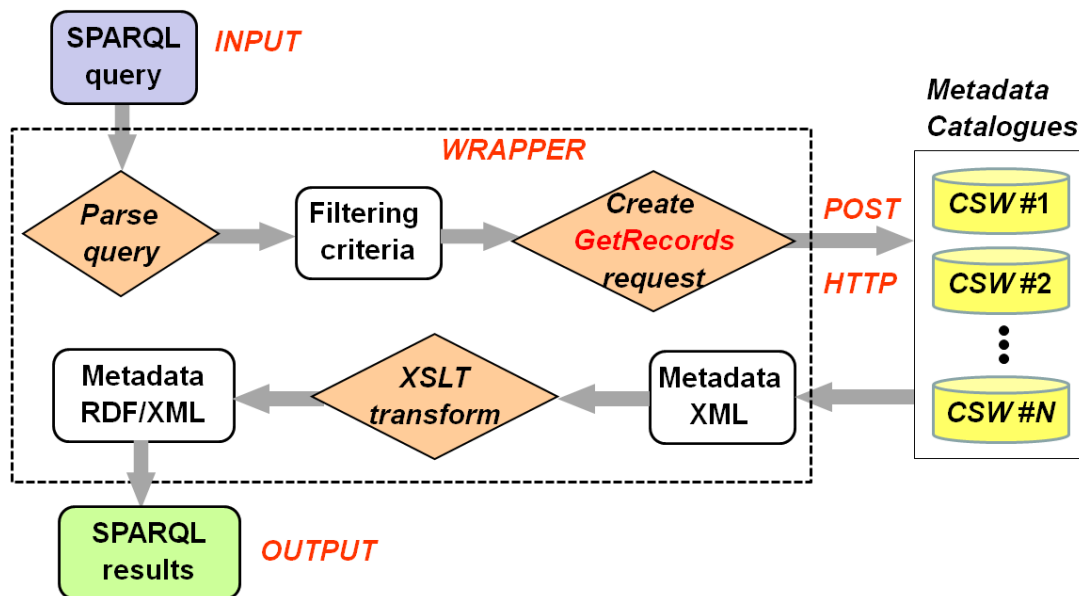


Figure 3: Flow diagram for processing SPARQL queries in the TripleGeo-CSW middleware.

The flow diagram of the process executed by this middleware is illustrated in Figure 3. Its input is a SPARQL query specifying one or more conditions according to the INSPIRE ontology with suitable DCAT mappings, as explained in Section 3. We have implemented a parser which identifies these conditions, including spatial predicates as documented next. The OGC standard defines a specific model for CSW requests (POST/GET HTTP protocols), which is quite complex. However, our major concern here is the CSW `<Filter>` element, which controls whether metadata should be retained according to specific criteria. Hence, the user-specified SPARQL conditions must be internally rewritten and then integrated into the `<Filter>` element of a `GetRecords` request for CSW. Thanks to the OGC standard for CSW, an identical such request will be submitted simultaneously via POST HTTP protocol against each of the available catalogues. As soon as each catalogue provides its response as a collection of qualifying metadata records in a separate XML file, the wrapper will merge their contents. With the same stylesheet described in Section 3, these metadata records (conforming to ISO 19115 specifications) are finally converted into XML/RDF triples and can be made available to the user. Of course, the final output will be INSPIRE-compliant metadata, since the XSLT transformation uses templates that map each metadata element into DCAT elements.

This open-source middleware is publicly available from [TripleGeo-CSW]. More technical details about its implementation can be found in the Appendix.

In our scenario, we assume that the user wishes to discover whether there are any available, updated spatial datasets for a given geographical area, under a certain category (e.g., transport, hydrography), or by specifying a keyword (e.g., "Architecture", "rail"). More specifically, the filtering criteria in these SPARQL queries may include one or more of the following:

- *Matching constant values.* This search involves metadata records with a given constant value or a string literal (e.g., "Architectural") as the object in the triple pattern.
- *Matching regular expressions (REGEX)* against string literals. Through FILTER conditions in SPARQL, the user can specify matchings of string values (e.g., "^environment\*") with keywords, titles, subjects, etc. present in the metadata records.
- *Date comparisons* make use of typical operators (>,<,<=,>=) and a constant date value, in order to identify datasets issued, modified or published before or after that particular date.
- *Spatial filtering.* INSPIRE-compliant metadata include the bounding box (BBOX) of the geographical extent for each dataset as an indication of its coverage area. Therefore, it makes much sense to allow users identify whether there are any available data in their region of interest, also specified as a box. Currently, we support typical GeoSPARQL topological predicates like sfWithin(), sfContains(), sfIntersects(), sfOverlaps(), etc. [OGC12], which can be suitably translated into CSW spatial filters over rectangles. For instance, operator sfWithin() checks whether a user-specified BBOX is totally within the coverage of a dataset, sfIntersects() identifies whether a given BBOX intersects the coverage of a dataset, etc.

Logical operators for conjunction (&&) and disjunction (||) can be used to combine filtering criteria, whereas a UNION clause can bind statements specifying multiple alternative patterns (e.g., searching for datasets characterized by subjects like "road" or "rail"). For instance, issuing the following SPARQL query in Figure 4 will retrieve INSPIRE metadata (in any of the listed CSWs) related to architectural features in the environment within a given geographic rectangle:

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/geosparql/function/>
SELECT *
WHERE { ?m dct:subject ?k .
        ?m dct:title "Architectural" .
        ?m geo:hasGeometry ?fWKT .
        FILTER ( REGEX(str(?k), "^environment*") &&
                  geof:sfWithin(?fWKT, "BOX2D(-6.51 53.27, -6.28 53.36)""^geo:wktLiteral) )
}
```

**Figure 4: Example GeoSPARQL query against INSPIRE metadata.**

Note that no graph is specified, as neither do we make use of a physically stored semantic repository nor any RDF triples get retained permanently. The TripleGeo-CSW middleware automatically rewrites this query into the equivalent GetRecords request shown in Figure 5, which may be submitted to each of the available CSWs. Consequently, all results are generated *on-the-fly* based on the response from CSWs.

In essence, processing components are wrapped under this virtual SPARQL endpoint, thus offering the ability to interact directly with any number of remote (yet compatible to CSW) catalogues and repurpose existing spatial metadata across Europe.

```
<?xml version='1.0' encoding='utf-8'?>
<GetRecords xmlns="http://www.opengis.net/cat/csw/2.0.2"
```

```

xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:ows="http://www.opengis.net/ows"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dct="http://purl.org/dc/terms/"
xmlns:gml="http://www.opengis.net/gml"
xmlns:gmd="http://www.isotc211.org/2005/gmd"
xmlns:apiso="http://www.opengis.net/cat/csw/apiso/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
service="CSW" version="2.0.2" startPosition="1" resultType="results" maxRecords="100"
outputFormat="application/xml" outputSchema="http://www.isotc211.org/2005/gmd"
xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2
http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd">
  <Query typeNames="gmd:MD_Metadata">
    <ElementSetName typeNames="gmd:MD_Metadata">full</ElementSetName>
    <Constraint version="1.1.0">
      <ogc:Filter>
        <ogc:And>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>dct:title</ogc:PropertyName>
            <ogc:Literal>Architectural</ogc:Literal>
          </ogc:PropertyIsEqualTo>
          <ogc:PropertyIsLike wildCard="^" singleChar="_">
            <ogc:PropertyName>dct:subject</ogc:PropertyName>
            <ogc:Literal>^environment*</ogc:Literal>
          </ogc:PropertyIsLike>
          <ogc:Within>
            <ogc:PropertyName>ows:BoundingBox</ogc:PropertyName>
            <gml:Envelope>
              <gml:lowerCorner>-6.51 53.27</gml:lowerCorner>
              <gml:upperCorner>-6.28 53.36</gml:upperCorner>
            </gml:Envelope>
          </ogc:Within>
        </ogc:And>
      </ogc:Filter>
    </Constraint>
  </Query>
</GetRecords>

```

**Figure 5: An example GetRecords request submitted to INSPIRE-compliant CSWs.**

## 5. Transforming INSPIRE Data into RDF

At a conceptual level, INSPIRE data models use UML [UML], while the actual entities are encoded in GML according to INSPIRE Implementing Rules [INSPIRE-Rules]. Geography Markup Language (GML) is an OGC standard [GML] for representing geospatial information. The basic primitives of GML are *spatial features* as locations on Earth, and their geometric shapes are modeled with vectors (points, lines, polygons, etc.). However, this model can be extended to accommodate more specific feature classes, e.g., roads or parcels, with the addition of application-defined properties. The original GML model (version 1.0) was based on RDF/RDFS profiles, but afterwards the OGC introduced XML schemas for interoperability with existing spatial databases. This means that automatic transformation of data and metadata from GML 1.0 into RDF was possible. Although this RDF profiling is no longer supported, subsequent GML models (current version is 3.3) still retain certain features of RDF, and most importantly the concept of child elements as properties of a parent resource (RDFS).

Thanks to this *relaxed "triple-like"* binding, GML features may be transformed into RDF. This dual representation in the geospatial domain is precious, since reasoning is possible with RDF, while GML enables operation of standardized SDIs. In our particular case, INSPIRE-aligned schemata are available in GML for several Data Themes [INSPIRE-GML]. However, a common methodology or guidelines for exposing such geospatial data as Linked Data are not yet agreed. Work-in-progress by the JRC on selected INSPIRE themes (like road networks, addresses, or administrative units) for the ISA CORE Location Vocabulary [ISA] focuses on rules for creating RDF vocabularies from UML models, but how to map UML stereotypes used in spatial models into RDF is not yet established.

In contrast to this abstract, *model-based* methodology, ours is a *data-centric* approach, attempting to encode elements from GML application schemata into RDF. As we are mostly concerned with accurate spatial representations, mapping from the GML schema is advantageous, because it has well-defined spatial semantics. Our goal is to develop a stack of tools suitable for such transformations, and also identify modeling peculiarities, intricate cases, and open issues. Using XSL stylesheets for GML representations of INSPIRE-compliant features seems most preferable and generic, as these scripts can be re-used and can work with any XSLT parser. The proposed XSLT transformation imitates INSPIRE schema and produces a compliant RDF representation. However, XSLT is not aware of the INSPIRE vocabulary in order to convert e.g., a name into `rdf:literal`. Therefore, we have introduced one custom XSL stylesheet for each INSPIRE Data Theme (Annex I), practically translating each domain-specific element in the respective GML schema into a suitable RDF resource. In addition, resulting geometry serializations are compliant with the recent OGC GeoSPARQL standard in order to be compatible with next-generation triple stores. Throughout this task, we had to choose a proper representation for RDF resources, handle many different cases in geometry representations (also due to earlier GML versions), and also resolve certain inconsistencies in namespaces. Next, we outline our approach on each of these challenging issues.

### 5.1 Specification of URIs

The most important requisite in exposing INSPIRE SDIs as Linked Data is *linking with http URIs*. Object referencing and associations are supported by INSPIRE Data Specifications [INSPIRE-Specs], but spatial datasets do not represent or maintain links. Stable identifiers must be supported and maintained for data, metadata, and services. The authorities publishing this content are responsible to assign persistent URIs to them. With a few exceptions, such as UK Location URI patterns [UK-URI] for `data.gov.uk`, currently there is no such support. Despite some general recommendations [CoolURI], discussions continue between EU member states and the



Commission towards an agreed URI strategy. Of course, this would require schemes that could manage assignment of http URIs and reuse existing ones for already published resources.

In order to avoid non-resolvable URNs in the resulting triples, we opt for a parametrized configuration of http URIs. Orthogonal to approaches on persistent URIs [SLP+14], the general pattern of our provisional URIs is

```
http://{domain}/id/{theme}/{dataSource}[/{resourceType}]/{localId}</AU:inspireId>
```

where:

- {domain} is the owner or domain root of a dataset (e.g., geodata.gov.gr);
- {theme} denotes the class where each spatial entity belongs to (e.g., administrative units, roads, etc.);
- {dataSource} is the dataset identifier in the national coprus of spatial data;
- and {localId} is a string identifier that could be the official code of the entity (e.g., municipality code, road code). Note that {localId} should not be confused with the gmlId of the respective GML feature, whereas it may not be necessarily equivalent to the composite inspireId required by INSPIRE. This {localId} should preferably be an official code that uniquely identifies the entity within the given dataset, e.g., in municipalities could be the official code assigned by the Ministry of Interior, for postal codes could be the code itself, etc.
- The *optional* {resourceType} can be used to specify nested resources that depend on the main entity, e.g., URIs for its geometry or its inspireId.

For example, the URI for municipality of Athens (having code 9186 under the "Kallikratis" subdivision of Greek municipalities) is `http://geodata.gov.gr/id/AdministrativeUnit/Kallikratis/9186`, while `http://geodata.gov.gr/id/AdministrativeUnit/Kallikratis/geometry/9186` is used to identify its respective geometry. Such provisional URIs may seem somewhat long, but we preferred this structured, self-describing notation instead of more compact ones.

## 5.2 Assigning INSPIRE Identifiers

There is no consensus on how to formulate an inspireId, i.e., the unique object identifier published by the responsible body, such that the respective object would be traceable in pan-European SDIs. Instead of issuing a single numeric identifier, we abide by the INSPIRE recommendations and we specify it through a combination of a namespace pertinent to the data provider (e.g., GR.ELSTAT\_AU for the current administrative boundaries of Greece as published by the Hellenic Statistical Authority - ELSTAT) and a localId for the feature (e.g., 9186 for the municipality of Athens).

This identifier concerns the spatial entity (not the real-world phenomenon), and serves as its reference for external applications. Thus, we suggest handling inspireId as a base RDF element, with all such resources having the same prefix, independent of their original theme. The following fragment of RDF/XML illustrates how such identifiers are being handled:

```
<AU:inspireId>
  <rdf:Description
    rdf:about="http://geodata.gov.gr/id/AdministrativeUnit/Kallikratis/inspireId/9186">
    <BASE:localId>9186</BASE:localId>
    <BASE:namespace>GR.ELSTAT_AU</BASE:namespace>
  </rdf:Description>
</AU:inspireId>
```



Note that prefixes AU: `<http://inspire.jrc.ec.europa.eu/schemas/au/3.0/>` and BASE: `<http://inspire.jrc.ec.europa.eu/schemas/base/3.2/>` have been used to specify these resources, although these namespaces do not have a RDF representation yet (i.e., these URIs are not yet resolvable). However, since construction of URIs and identifiers is entirely parametrized in the proposed XSL stylesheets, all these can be easily altered very easily to comply with any future official recommendations.

### 5.3 Geometry Serializations

Another innovation of our approach is its support for geometries according to the recent OGC GeoSPARQL standard [OGC12]. This is basically achieved through an XSLT transformation of the original 2-dimensional geometries from Geometry Markup Language (GML) into valid Well Known Text (WKT) serializations. This stylesheet can handle all basic shapes (*Point*, *MultiPoint*, *LineString*, *MultiLineString*, *Curve*, *Polygon*, *MultiPolygon*), as well as more complex spatial types (like *MultiSurface*, *MultiGeometry*, and *GeometryCollection*) defined by the respective OGC standard [OGC11]. Details about how to encode such geometries in RDF can be found in Deliverable D2.1.1 “Market and Research Overview” [GeoKnowD21].

The resulting RDF statements also encode the coordinate reference system (CRS) along with each geometry according to the EPSG catalogue [EPSG], e.g., lon/lat coordinates in WKT georeferenced in WGS84 (EPSG:4326) for this point (highlighted in the red box):

```
<geo:hasGeometry>
  <rdf:Description rdf:about="http://geodata.gov.gr/id/NamedPlace/geometry/A1010101">
    <geo:asWKT rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral">
      <http://www.opengis.net/def/crs/EPSG/0/4326> POINT (23.7279 37.9841)
    </geo:asWKT>
  </rdf:Description>
</geo:hasGeometry>
```

Note that geometries are defined according to the GeoSPARQL predicate `<geo:hasGeometry>`, in order to be fully conformant to the standard. The original INSPIRE specification for geometries depends on the data theme, e.g. `<GN:geometry>` for geographical names (GN); for consistency, we also include such triples in the results. The GeoSPARQL vocabulary defines a `<geo:asWKT>` property, where the WKT serialization is attached as a geometry literal. Finally, we generate triples that specify the spatial data type (as an XML attribute) of each geometry, along with its WKT literal.

### 5.4 Converting Thematic Attributes

Output RDF triples should comply to INSPIRE schemata, regarding data for (AD) Addresses, (GN) Geographical Names, etc. classified in Annex I. For most themes, these GML application schemata [INSPIRE-GML] are very detailed in an attempt to cover many diverse cases for entity representation that arise across Europe. For example, the `level` attribute in an AD:AddressLocator element could correspond to a value like *"site level"* or *"postal delivery point"* or a *"unit level"* locator, depending on national or regional regulations. This entails a certain degree of customization in XSL stylesheets, which we have chosen to reflect the original INSPIRE schema. All facts, especially in RDF predicates (i.e., the middle component in triples), respect the INSPIRE namespace. For instance, AU:nationalCode assigns the official code (a string) into the resource denoting an administrative unit, even though namespaces like AU:

<<http://inspire.jrc.ec.europa.eu/schemas/au/3.0/>> do not have an RDF representation yet.

Language specification in string literals is another concern because supporting multi-lingual content is one of INSPIRE objectives. In its schema specifications, attribute 'language' in geographical names is given according to ISO 639-3 (i.e., three-letter codes like 'eng', 'ell', or 'ita'). However, for language-tagged literals in RDF, a two-letter encoding (e.g., 'en', 'el', 'it') according to ISO 639-1 must be utilized instead. Our XSLT stylesheets can handle this case, as well as multiple alphabets (most data in [geodata.gov.gr](http://geodata.gov.gr) is in Greek), discriminate alternative spellings, etc. The following fragment in Figure 6 illustrates transformation into RDF of a geographical name from the Greek administrative units (for the municipality of Athens, highlighted in the red box):

```
<AU:name>
  <rdf:Description rdf:about="http://geodata.gov.gr/id/AdministrativeUnit/Kallikratis/name/9186">
    <GN:GeographicalName>
      <rdf:Description
rdf:about="http://geodata.gov.gr/id/AdministrativeUnit/Kallikratis/GeographicalName/9186">
        <GN:language>ell</GN:language>
        <GN:nativeness>endonym</GN:nativeness>
        <GN:nameStatus>official</GN:nameStatus>
        <GN:sourceOfName>Hellenic Mapping and Cadastre Organization</GN:sourceOfName>
        <GN:pronunciation/>
        <GN:grammaticalGender/>
        <GN:grammaticalNumber/>
        <GN:spelling>
          <rdf:Description rdf:about="http://geodata.gov.gr/id/AdministrativeUnit/
Kallikratis/spelling/9186">
            <GN:SpellingOfName>
              <rdf:Description
rdf:about="http://geodata.gov.gr/id/AdministrativeUnit/Kallikratis/SpellingOfName/9186">
                <GN:text xml:lang="el">Αθηναίωv</GN:text>
                <GN:script>Grek</GN:script>
              </rdf:Description>
            </GN:SpellingOfName>
          </rdf:Description>
        </GN:spelling>
      </rdf:Description>
    </GN:GeographicalName>
  </rdf:Description>
</AU:name>
```

**Figure 6: A geographic name (in Greek) as expressed in INSPIRE-compliant RDF/XML.**

The transformation should also account for voidable properties, e.g., roads without names. It is also possible that no information exists in the original dataset regarding several nillible attributes in the INSPIRE schemata. For example, a name may be present for a spatial entity, but further

details about pronunciation, grammatical gender etc. may be missing. This may yield RDF triples with blank literals in objects, which inevitably increase the amount of resulting RDF statements.

We have made extensive use of these XSLT transformations against Greek geospatial data, as detailed next. Nevertheless, in order to verify their suitability we made additional tests using sample data from other countries, such as geographical names and protected sites from Italy, cadastral parcels in the UK, as well as roads and addresses in the Netherlands. The original sample datasets (in GML) as well as the transformed ones (in RDF/XML) are available at [INSPIRE2RDF]. We stress that it is not the amount of data that matters, but its variety in attributes, geometries, and property specifications that affect the quality and validity of such transformations.

## 6. Exposing INSPIRE SDI as RDF: A Use Case in Greece

### 6.1 geodata.gov.gr: a SDI for Greece

The web service `geodata.gov.gr` was developed by IMIS/"Athena" RC, under the auspices of the Prime Minister's eGovernment Task Force with the dual purpose to promote open governmental data and assist in the technical implementation of the INSPIRE Directive in Greece. Until its introduction, Greece lacked a national SDI despite the INSPIRE Directive's mandates. As a result, data sharing agreements between Greek public authorities were scarce, and the same spatial datasets were procured multiple times.

The goal was to build a web site in order to offer government geospatial data under an open data license. Operating since August 2010 [`geodataGR`], `geodata.gov.gr` was only one out of eight national open data portals worldwide, has led to direct savings of €20 million for the public sector alone, and is actively used by about 1000 users/day. It offers open geospatial data provided by the public sector, across diverse thematic domains (e.g., protected sites, census, placenames, water quality, etc.).

Built entirely with open-source software and in only six months, `geodata.gov.gr` provides two main services: (i) a data catalogue and (ii) interactive maps. The catalogue is the main entrance point, enabling users to search and download geospatial data in several formats (e.g. shapefiles, KML, GML). It is based on Joomla/MySQL, with some custom extensions to support the required functionality. This was deemed necessary to promote uptake from experts (knowledgeable in GIS) and citizens alike (easy to publish on Google Maps). Full INSPIRE-compliant metadata is also provided, created and validated using the online INSPIRE metadata editor [`INSPIRE-editor`].

Interactive maps in `geodata.gov.gr` assist citizens in comprehending spatial data, and combine vector data and raster imagery. All vector spatial data are stored in PostGIS [`PostGIS`], while raster data are either kept on the file system, or integrated from other web services (WMS, Google Maps API, Microsoft Bing Maps API). The UMN MapServer [`MapServer`] is used to create maps which in turn are offered as WMS and/or WFS, depending on the complexity of the particular dataset. Further, the MapFish framework [`MapFish`] was used to aid in developing the web client, which incorporates several JavaScript libraries (OpenLayers, geoExt, jQuery). The full architecture is RESTful, offering increased performance and resilience.

The actual service is deployed in eight Ubuntu Virtual Machines (VM), kindly provided and hosted by GRNET S.A. [GRNET] with modest to low specifications (2GB main memory, 45GB storage, one 2.0 GHz core per VM). All VMs belong to the same software stack and their roles can be altered in a few minutes to support changing needs. A typical distribution is 2 VMs for the catalogue, and 6 VMs for the map server.

Although all metadata in `geodata.gov.gr` is fully compatible with INSPIRE regulations, data is not. Due to their variety, provenance, and excessive volume, their transformation into INSPIRE-compliant datasets is a time-consuming and demanding task. Therefore, we have chosen to expose indicative datasets for seven data themes of Annex I, as detailed in Table 2. More details on how we made use of this data are given in the Appendix, along with its licensing policy.

In the following, we first report our experience from aligning this data to INSPIRE specifications. Further, we present our methodology for (i) transforming this INSPIRE-compliant data and metadata into RDF and (ii) exposing them through GeoSPARQL and SPARQL endpoints.

**Table 2: Datasets from `geodata.gov.gr` exposed as INSPIRE-aligned RDF data.**

INSPIRE Data Theme	Greek dataset	Number of features	Number of triples
[GN] <i>Geographical names</i>	Settlements, towns, and localities in Greece.	13259	304957
[AU] <i>Administrative units</i>	All Greek municipalities after the most recent restructuring ("Kallikratis").	326	9454
[AD] <i>Addresses</i>	Street addresses in Kalamaria municipality.	10776	277838
[CP] <i>Cadastral parcels</i>	The building blocks in Kalamaria are used. Data from the official Greek Cadastre are not available through <a href="http://geodata.gov.gr">geodata.gov.gr</a> .	965	13510
[TN] <i>Transport networks</i>	Urban road network in Kalamaria.	2584	59432
[HY] <i>Hydrography</i>	All rivers and waterstreams in Greece.	4299	120372
[PS] <i>Protected sites</i>	All areas of natural preservation in Greece according to the EU Natura 2000 network.	419	10894
	<b>Total</b>	<b>32627</b>	<b>796457</b>

## 6.2 Data Alignment to INSPIRE Annex I

Despite their modest size, transforming the data of Table 2 according to the INSPIRE Annex I Data Specifications [INSPIRE-themes], posed several challenges. Primarily, the original GML files had to become INSPIRE-aligned through mappings between their schemata. To this goal, we utilized the *Humboldt Alignment Editor* [HALE], a powerful open-source tool with a graphical interface and a high-level language for expressing custom alignments.

Such transformation can be used to turn a non-harmonised data source to an INSPIRE-compliant dataset. It only requires a source schema (an .xsd for the local GML file) and a target one (an .xsd implementing an INSPIRE data schema). The operation is a "one-to-one" mapping, whereby each element from the local GML schema can be mapped to one in the INSPIRE schema. The editor contains mapping functions that can handle attribute types and properties (e.g., for calculating length or area of shapes). Although INSPIRE schemata include plenty of elements and sub-elements in order to cover all possible use cases, most of them are not obligatory. In effect, the mapping depends on the variety of elements in the local GML file, i.e., how many attributes are available per feature. So, the quality of the source schema is a factor that may affect the final mapping and the amount of information that will be transformed.

The following fragment in Figure 7 depicts a *mapping cell* from such an alignment (complete listing, as well as all other alignments for data from [geodata.gov.gr](http://geodata.gov.gr) can be found at [INSPIRE2RDF]). Such a cell specifies source and target entities, a relation function and possibly parameters for that function. In the example below, the type `oria_dhmwn_kallikraths_Type` with child name "geometryProperty" from the source schema is mapped to type `AdministrativeUnitType` with child name "geometry" in the target schema (both properties are highlighted with a red box). Their relation is represented by the `Rename` function, which is a property function that expresses a source and target property being semantically equal.

```
<alignment xmlns="http://www.esdi-humboldt.eu/hale/alignment">
  ...
  <cell relation="eu.esdihumboldt.hale.align.rename" id="Ce6e7f1f2-c61f-451d-8c8c-4dd947739268"
    priority="normal">
```

```

<source>
  <property>
    <type name="oria_dhmwn_kallikraths_Type" ns="http://ogr.maptools.org/">
    <child name="geometryProperty" ns="http://ogr.maptools.org/">
    <child name="choice" ns="http://www.opengis.net/gml/_Geometry"/>
    <child name="MultiPolygon" ns="http://www.opengis.net/gml"/>
    <child name="polygonMember" ns="http://www.opengis.net/gml"/>
    <child name="Polygon" ns="http://www.opengis.net/gml"/>
  </property>
</source>
<target>
  <property>
    <type name="AdministrativeUnitType" ns="urn:x-inspire:specification:gmlas:AdministrativeUnits:3.0"/>
    <child name="geometry" ns="urn:x-inspire:specification:gmlas:AdministrativeUnits:3.0"/>
    <child name="MultiSurface" ns="http://www.opengis.net/gml/3.2"/>
    <child name="surfaceMember" ns="http://www.opengis.net/gml/3.2"/>
    <child name="choice" ns="http://www.opengis.net/gml/3.2/AbstractSurface"/>
    <child name="Surface" ns="http://www.opengis.net/gml/3.2"/>
  </property>
</target>
<parameter value="false" name="structuralRename"/>
</cell>
...
</alignment>

```

**Figure 7: Excerpt of HALE alignment to INSPIRE for geometries of Greek administrative units.**

As soon as the schema mapping is defined, the source GML data can be loaded, and the INSPIRE-aligned GML file is produced. It is important to note that the editor not only exports the data, but the attribute alignments as well. Hence, these schema mappings can certainly be reused for the same source data in the future, e.g., in case of updates.

Although this INSPIRE harmonization has been prepared for datasets from [geodata.gov.gr](http://geodata.gov.gr), it may also be applied against similarly structured data with the necessary arrangements at conflicting or missing attributes (e.g., by replacing an attribute "roadName" with "street-Name" as listed in another source GML file).

### 6.3 Transforming Features from [geodata.gov.gr](http://geodata.gov.gr) into RDF

As soon as data were harmonized to INSPIRE, it had to be transformed into RDF. This process against such GML files is quite straightforward, provided a set of suitable XSL stylesheets. We developed all these transformations in XSLT 2.0, implementing one parametrized stylesheet per data theme in Table 2. By default, all geometries were encoded in WKT serializations according to GeoSPARQL. The only prerequisite was the selection of appropriate URIs for the resulting resources, as discussed in Section 5.1. In practice, the user should determine values for



the parameters involved, i.e., domain, theme, and dataSource. E.g., for the municipality of Athens (id=9186) in Greek administrative units, the main resource should be:

```
http://geodata.gov.gr/id/AdministrativeUnit/Kallikratis/9186
```

since domain='geodata.gov.gr', theme='AdministrativeUnit' and dataSource='Kallikratis'.

The case was simpler for metadata according to the process outlined in Section 3, since a common, general-purpose stylesheet is applied to each XML file. In total, 475 metadata triples were produced for the seven themes listed in Table 2.

As soon as the resulting RDF/XML files were derived, they could be inserted into a semantic repository. We have chosen Parliament RDF prototype [Parliament], which offers support for storage and search capabilities fully compliant to the GeoSPARQL standard. One RDF graph was used to hold triples on INSPIRE data, and another one for INSPIRE metadata, both supported by a spatial index.

In addition, we set up two similar graphs in Virtuoso Universal Server [Virtuoso], which offers a powerful query engine and a compressed column store representation for RDF. Although Virtuoso is very robust and highly scalable, and has recently (version 7.1) included support for several spatial data types, it is not GeoSPARQL-compliant yet and has limited support for geometric processing. Thus, we had to execute the XSLT transformations so as to convert all geometries into custom representations for Virtuoso. This has proved the versatility of our approach and its potential to support diverse geometry serializations.

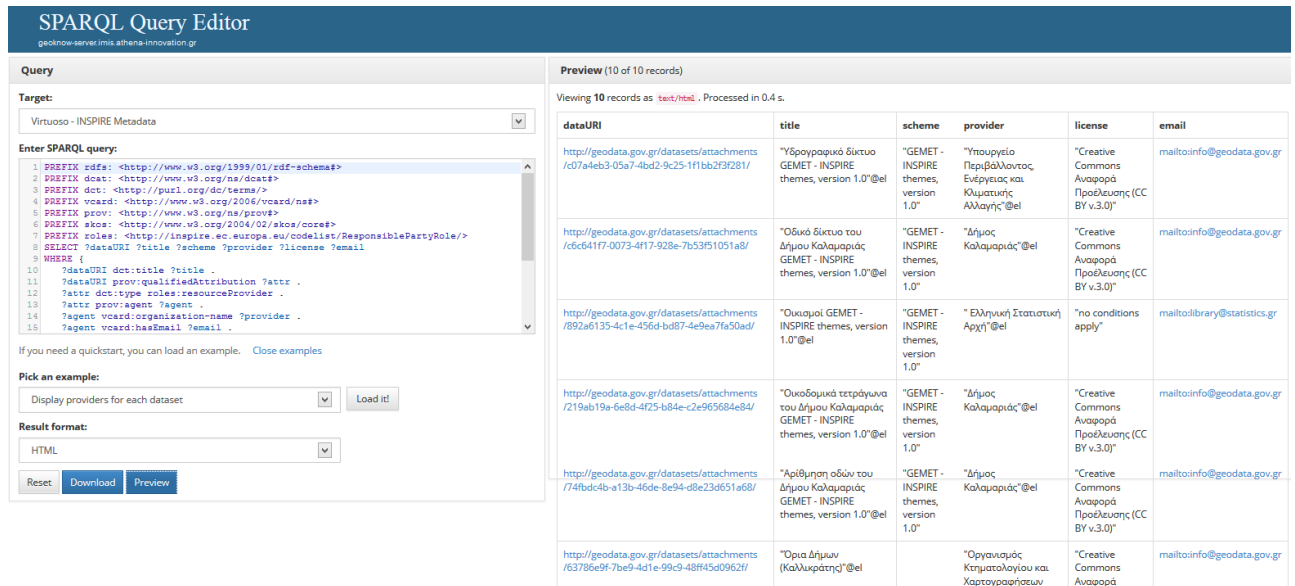
Configuration details for these repositories, along with information on the created graphs are available in the Appendix.

## 6.4 Enabling Queries through (Geo)SPARQL Endpoints

Taking advantage of the embedded capabilities of the two RDF stores (Parliament, Virtuoso), we were able to create endpoints that expose INSPIRE data and metadata. Actually, Parliament provides two GeoSPARQL endpoints (one for data, another for metadata); Virtuoso offers two similar SPARQL endpoints with geometric support as well, although not conforming to GeoSPARQL. In addition, there is another virtual SPARQL endpoint, which works on top of the CSW-to-RDF middleware (Section 4). Overall, the user has the ability to choose one of the following target repositories (Table 3) and submit a wide range of (Geo)SPARQL queries (listed and explained in the Appendix):

**Table 3: Target (Geo)SPARQL endpoints set up for the use case scenario.**

Backend repository	Contents	Language
Virtuoso	INSPIRE-compliant <b>data</b> from geodata.gov.gr	SPARQL
Virtuoso	INSPIRE-compliant <b>metadata</b> from geodata.gov.gr	SPARQL
Parliament	INSPIRE-compliant <b>data</b> from geodata.gov.gr	GeoSPARQL
Parliament	INSPIRE-compliant <b>metadata</b> from geodata.gov.gr	GeoSPARQL
TripleGeo-CSW	INSPIRE <b>metadata</b> from a collection of CSW catalogues in Europe	GeoSPARQL



**SPARQL Query Editor**  
geonow-server imis.athina-innovation.gr

**Query**

Target:  
Virtuoso - INSPIRE Metadata

Enter SPARQL query:

```

1 PREFIX rdfs: <http://www.w3.org/1999/01/rdf-schema#>
2 PREFIX dcat: <http://www.w3.org/ns/dcat#>
3 PREFIX dct: <http://purl.org/dc/terms/>
4 PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
5 PREFIX prov: <http://www.w3.org/ns/prov#>
6 PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
7 PREFIX roles: <http://inspire.ec.europa.eu/codelist/ResponsiblePartyRole/>
8 SELECT ?dataURI ?title ?scheme ?provider ?license ?email
9 WHERE {
10   ?dataURI dcat:title ?title .
11   ?dataURI prov:qualifiedAttribution ?attr .
12   ?attr dct:type roles:resourceProvider .
13   ?attr prov:agent ?agent .
14   ?agent vcard:organisation-name ?provider .
15   ?agent vcard:hasEmail ?email .
16 }

```

If you need a quickstart, you can load an example. [Close examples](#)

Pick an example:  
Display providers for each dataset ☐ [Load it!](#)

Result format:  
HTML ☐

[Reset](#) [Download](#) [Preview](#)

**Preview (10 of 10 records)**

Viewing 10 records as [text/html](#). Processed in 0.4 s.

dataURI	title	scheme	provider	license	email
<a href="http://geodata.gov.gr/datasets/attachments/c07a4eb3-05a7-4bd2-9c25-1f1bb2f3f281/">http://geodata.gov.gr/datasets/attachments/c07a4eb3-05a7-4bd2-9c25-1f1bb2f3f281/</a>	"Υδρογραφικό δίκτυο GEMET - INSPIRE themes, version 1.0"@el	"GEMET - INSPIRE themes, version 1.0"	"Υπουργείο Περιβάλλοντος, Ενέργειας και Κлимτικής Αλλαγής"@el	"Creative Commons Αναφορά Προέλευσης (CC BY v.3.0)"	<a href="mailto:mailto:info@geodata.gov.gr">mailto:info@geodata.gov.gr</a>
<a href="http://geodata.gov.gr/datasets/attachments/c6c641f7-0073-4f17-928e-7b53f51051a8/">http://geodata.gov.gr/datasets/attachments/c6c641f7-0073-4f17-928e-7b53f51051a8/</a>	"Οδικό δίκτυο του Δήμου Καλαμαριάς GEMET - INSPIRE themes, version 1.0"@el	"GEMET - INSPIRE themes, version 1.0"	"Δήμος Καλαμαριάς"@el	"Creative Commons Αναφορά Προέλευσης (CC BY v.3.0)"	<a href="mailto:mailto:info@geodata.gov.gr">mailto:info@geodata.gov.gr</a>
<a href="http://geodata.gov.gr/datasets/attachments/892a6135-4c1e-456d-bd87-4e9ea7fa50ad/">http://geodata.gov.gr/datasets/attachments/892a6135-4c1e-456d-bd87-4e9ea7fa50ad/</a>	"Όνομα GEMET - INSPIRE themes, version 1.0"@el	"GEMET - INSPIRE themes, version 1.0"	"Ελληνική Στατιστική Αρχή"@el	"no conditions apply"	<a href="mailto:mailto:library@statistics.gr">mailto:library@statistics.gr</a>
<a href="http://geodata.gov.gr/datasets/attachments/219ab19a-6e8d-4f25-b84e-c2e965684e84/">http://geodata.gov.gr/datasets/attachments/219ab19a-6e8d-4f25-b84e-c2e965684e84/</a>	"Οικοδομικό τετράγωνο του Δήμου Καλαμαριάς GEMET - INSPIRE themes, version 1.0"@el	"GEMET - INSPIRE themes, version 1.0"	"Δήμος Καλαμαριάς"@el	"Creative Commons Αναφορά Προέλευσης (CC BY v.3.0)"	<a href="mailto:mailto:info@geodata.gov.gr">mailto:info@geodata.gov.gr</a>
<a href="http://geodata.gov.gr/datasets/attachments/74f4bdc4b-e13b-46de-8e94-d8e23d651a68/">http://geodata.gov.gr/datasets/attachments/74f4bdc4b-e13b-46de-8e94-d8e23d651a68/</a>	"Αριθμός οδών του Δήμου Καλαμαριάς GEMET - INSPIRE themes, version 1.0"@el	"GEMET - INSPIRE themes, version 1.0"	"Δήμος Καλαμαριάς"@el	"Creative Commons Αναφορά Προέλευσης (CC BY v.3.0)"	<a href="mailto:mailto:info@geodata.gov.gr">mailto:info@geodata.gov.gr</a>
<a href="http://geodata.gov.gr/datasets/attachments/63786e9f-7be9-4d1e-99c9-48f45d0962f/">http://geodata.gov.gr/datasets/attachments/63786e9f-7be9-4d1e-99c9-48f45d0962f/</a>	"Όρια Δήμων (Καλυκτρή)"@el	"GEMET - INSPIRE themes, version 1.0"	"Όργανισμός Κτηματολογίου και Χαρτογραφίσεων"	"Creative Commons Αναφορά Προέλευσης (CC BY v.3.0)"	<a href="mailto:mailto:info@geodata.gov.gr">mailto:info@geodata.gov.gr</a>

**Figure 8: Web interface for submitting (Geo)SPARQL queries against five backend engines.**

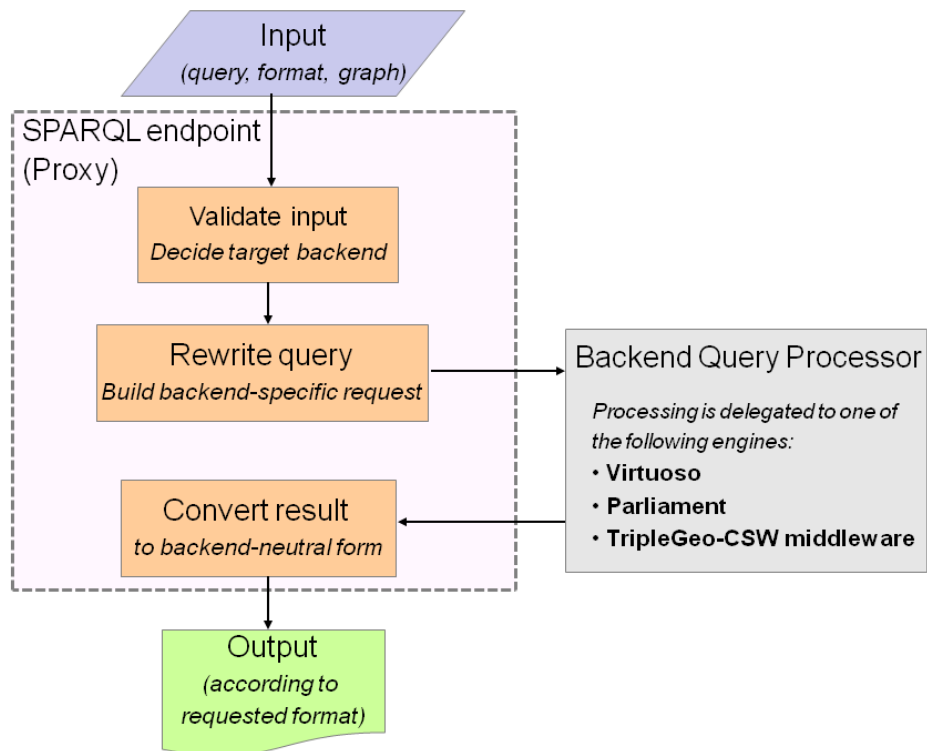
In order to hide the differences between those five backends and to expose a simple and uniform interface to the end-user, we have implemented a web application that provides the means to issue SPARQL queries and receive response in a variety of formats (RDF/XML, CSV, HTML, etc.). This web interface (illustrated in Figure 8) is currently available from

<http://geodata.gov.gr/sparql/>

and it is equipped with sample queries, a short documentation on the datasets and the transformation processes, as well as links to the source code.

The application consists of a client-side JavaScript application that provides a basic web interface. This allows the user to select a backend (SPARQL engine), edit a query, specify a result format, preview results, or even download them. In addition, a server-side PHP application acts both as an API proxy and an abstraction layer, as shown in Figure 9. Once it receives a query, our proxy validates it and determines which is the target backend, then builds the proper backend-specific request and finally sends back the results. Example queries against data or metadata (depending on the underlying graph) are listed in the Appendix, and this set of typical SPARQL requests is also available for execution through the web interface. Of course, users may specify their own queries provided that these comply with the underlying INSPIRE schema specifications. More technical details about the implementation of the web interface are given in the Appendix.

One final remark concerns our experience from posing SPARQL queries on these graphs. As can be verified from the example (Geo)SPARQL statements in the web interface (Section 9.2), queries tend to be quite verbose, mainly because they must reflect the underlying INSPIRE schema for the themes involved. This may result into many triple bindings during evaluation and thus increased response times. We observed that some queries in Parliament take much longer to execute than their counterparts in Virtuoso against the same datasets, mainly because of the limitations of Parliament's query engine. More details and evaluation results regarding performance of geospatial triple stores for typical queries can be found in Deliverable D2.1.1 "Market and Research Overview" [GeoKnowD21].



**Figure 9: Processing (Geo)SPARQL queries through the web interface.**

## 7. Conclusions and Perspectives

In this deliverable we presented our methodology and experiences towards repurposing the legally mandated and investment-intensive INSPIRE SDIs as high quality Linked Data sources. First, we have successfully enabled users to search for available INSPIRE data, by providing a virtual SPARQL interface for CSW services. This means that all INSPIRE Catalogue Services are accessible with Semantic Web technologies and thus INSPIRE data are discoverable. Further, we have developed a reusable transformation of INSPIRE metadata to RDF. As such, INSPIRE metadata can be automatically available to RDF, for all such metadata currently available, or in the future. Last, but not least, we have introduced a methodology for transforming INSPIRE Annex I data to RDF, and offer them through GeoSPARQL endpoints. This methodology may be applied as-is for all Annex I data across EU, and is inherently extensible for Annex II and Annex III data by means of simply authoring additional XSLT transformations. Overall, we demonstrated that exposing the entire spectrum of INSPIRE data as RDF through GeoSPARQL endpoints is possible, with relatively little work.

Exposing INSPIRE data in RDF and through GeoSPARQL endpoints can have significant impact for the GIS and Semantic Web communities. On the one hand, GIS researchers and practitioners can be acquainted with the advanced capabilities that Semantic Web technologies can offer, and also harness the wealth of LOD for novel applications. On the other hand, the Semantic Web can be seamlessly enriched with diverse, official, and high-quality geospatial data. Further, as INSPIRE licensing is gradually conforming to open knowledge licenses, we can envisage a LOD cloud with interlinked official and crowdsourced data, thus opening immense opportunities for new value added services.

Our work is by no means complete, and opens up several perspectives. First, an obvious extension is to create XSLT transformations for Annex II and Annex III data. In addition, we aim to apply domain-specific knowledge about the application schema in order to improve mapping, also taking into account established vocabularies, such as Dublin Core and SKOS. Second, we aim to integrate the INSPIRE-to-RDF transformations into TripleGeo [PAGA14], a generic utility we have developed in GeoKnow [TripleGeo], enabling the transformation of geospatial data to RDF. As such, we will provide a single utility enabling any geospatial data (standards-compliant or not) to be transformed to RDF and exposed through GeoSPARQL with limited effort. We intend to integrate this utility to open data catalogues and open source SDIs, thus offering a Semantic Web-ready solution for the GIS community. Another area we would like to explore concerns interlinking of INSPIRE RDF data with LOD sources, and consequently apply our data fusion framework FAGI [GMS+14] for producing new data sets of higher value, accuracy, and coverage. Of special interest is interlinking and fusion of OpenStreetMap data with relevant INSPIRE data sources. We believe this process will provide new challenges and solutions regarding multi-linguality, data integration and content quality.

**Acknowledgements.** We wish to thank Andrea Perego, Michael Lutz, and Robin Smith at the Joint Research Centre (JRC) of the European Commission for helpful discussions and for providing us their draft of RDF mappings on INSPIRE metadata.

## 8. References

- [AO13] S. Abbas and A. Ojo. Towards a Linked Geospatial Data Infrastructure. In EGOVIS/EDEM, pp. 196-210, 2013.
- [BK12] R. Battle and D. Kolas. GeoSPARQL: Enabling a Geospatial Semantic Web. Semantic Web Journal, 3(4): 355-370, IOS Press, November 2012.
- [BJQ14] L. van den Brink, P. Janssen, and W. Quak. Linking spatial data: automated conversion of geo-information models and GML data to RDF. IJSDIR, 2014.
- [CSW] OGC Inc. Catalogue Service. URL: <http://www.opengeospatial.org/standards/cat>
- [CoolURI] W3C. Cool URIs for the Semantic Web. URL: <http://www.w3.org/TR/cooluris/>
- [CC3] CC-BY 3.0. Creative Commons Attribution 3.0. URL: <http://creativecommons.org/licenses/by/3.0/>
- [DCAT] W3C. Data Catalog Vocabulary. URL: <http://www.w3.org/TR/vocab-dcat/>
- [DC11] Dublin Core Metadata Initiative. Dublin Core Metadata element set, Version 1.1. July 1999. URL: <http://dublincore.org/documents/dcmi-terms/>
- [DBpedia] DBpedia. URL: <http://dbpedia.org/About>
- [EPSG] European Petroleum Survey Group Geodetic Parameter Registry. URL: <http://www.epsg-registry.org/>
- [EU-open-data] EU Open Data Portal – Linked Data. URL: <https://open-data.europa.eu/en/linked-data>
- [geodataGR] IMIS/Athena R.C. Implementation of geodata.gov.gr (Flyer). URL: [http://geodata.gov.gr/geodata/images/geodatagovgr\\_info\\_en.pdf](http://geodata.gov.gr/geodata/images/geodatagovgr_info_en.pdf)
- [GeoKnowD21] S. Athanasiou, L. Bezati, G. Giannopoulos, K. Patroumpas, and D. Skoutas. GeoKnow EU/FP7 project. Deliverable 2.1.1: Market and Research Overview. URL: [http://svn.aksw.org/projects/GeoKnow/Public/D2.1.1\\_Market\\_and\\_Research\\_Overview.pdf](http://svn.aksw.org/projects/GeoKnow/Public/D2.1.1_Market_and_Research_Overview.pdf)
- [GeoKnowD22] S. Athanasiou, L. Bezati, G. Giannopoulos, K. Patroumpas, D. Skoutas, and C. Stadler. GeoKnow EU/FP7 project. Deliverable 2.2.1: Integration of External Geospatial Databases. URL: [http://svn.aksw.org/projects/GeoKnow/Public/D2.2.1\\_Integration\\_of\\_Geospatial\\_Databases.pdf](http://svn.aksw.org/projects/GeoKnow/Public/D2.2.1_Integration_of_Geospatial_Databases.pdf)
- [GPL3] GNU General Public License, version 3.0. URL: <http://www.gnu.org/licenses/gpl-3.0.html>
- [GeoNames] GeoNames geographical database. URL: <http://www.geonames.org/>
- [GeoNetDCAT] GeoNetwork Data Catalog Vocabulary services. URL: <http://trac.osgeo.org/geonetwork/wiki/proposals/DCATandRDFServices>
- [GMS+14] G. Giannopoulos, T. Maroulis, D. Skoutas, N. Karagiannakis, and S. Athanasiou. FAGI-tr: A tool for Aligning Geospatial RDF Vocabularies. In ESWC, 2014.
- [GSDI] Global Spatial Data Infrastructure Association. The SDI CookBook, 2009. URL: <http://www.gsdi.org/gsdicookbookindex>
- [GRNET] Greek Research & Technology Network. URL: <https://www.grnet.gr/>
- [HALE] HUMBOLDT Alignment Editor. URL: <http://community.esdi-humboldt.eu/projects/show/hale>

**[INSPIRE-Directive]** European Commission (EC). Directive 2007/2/EC establishing Infrastructure for Spatial Information in the European Community (INSPIRE), 14 March 2007. URL: <http://eur-lex.europa.eu/JOHtml.do?uri=OJ:L:2007:108:SOM:EN:HTML>

**[ISA]** European Commission. Interoperability Solutions for European Public Administrations (ISA) Programme. URL: <http://ec.europa.eu/isa/>

**[INSPIRE-Specs]** European Commission. INSPIRE Data Specifications. URL: <http://inspire.ec.europa.eu/index.cfm/pageid/2>

**[INSPIRE-Roadmap]** European Commission. INSPIRE Roadmap. URL: <http://inspire.ec.europa.eu/index.cfm/pageid/44>

**[INSPIRE-Rules]** European Commission. INSPIRE Implementing Rules. URL: <http://inspire.ec.europa.eu/index.cfm/pageid/47>

**[INSPIRE-editor]** European Commission. INSPIRE Metadata Editor. URL: <http://inspire-geoportal.ec.europa.eu/editor/>

**[INSPIRE-GML]** European Commission. INSPIRE GML application schemas. URL: <http://inspire.ec.europa.eu/schemas/>

**[INSPIRE-themes]** European Commission. INSPIRE Data Specifications. URL: <http://inspire.ec.europa.eu/index.cfm/pageid/2/list/7>

**[INSPIRE-GeoPortal]** European Commission. INSPIRE GeoPortal. URL: <http://inspire-geoportal.ec.europa.eu/>

**[INSPIRE2RDF]** Athena R.C. Tools for exposing INSPIRE metadata and data as linked data. URL: [https://web.imis.athena-innovation.gr/redmine/projects/geoknow\\_public/wiki/Inspire2RDF](https://web.imis.athena-innovation.gr/redmine/projects/geoknow_public/wiki/Inspire2RDF)

**[ISO19115]** ISO 19115-1:2014. Geographic information – Metadata. URL: [http://www.iso.org/iso/home/store/catalogue\\_ics/catalogue\\_detail\\_ics.htm?csnumber=53798](http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=53798)

**[ISO19119]** ISO 19119:2005. Geographic information – Services. URL: [http://www.iso.org/iso/home/store/catalogue\\_tc/catalogue\\_detail.htm?csnumber=39890](http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=39890)

**[ISO211]** ISO/TC 211 Geographic Information/Geomatics. URL: <http://www.isotc211.org/>

**[LFN+10]** F.J. Lopez-Pellicer, A.J. Florczyk, J. Nogueras-Iso, P.R. Muro-Medrano and F. J. Zarazaga-Soria. Exposing CSW Catalogues as Linked Data. In *Geospatial Thinking*, pp. 183-200, 2010.

**[MAHP11]** A. Mallica, M. Arenas, A. Hogan, and A. Polleres. On Blank Nodes. In ISWC, pp. 421-437, 2011.

**[MapFish]** OSGeo Foundation. MapFish. URL: <http://mapfish.org/>

**[MapServer]** MapServer Project. URL: <http://www.mapserver.org/>

**[Mimas]** Mimas Linked Data Project (UK). URL: <http://www.landmap.ac.uk/index.php/Projects/Linked-Data/Linked-Data-Project-Introduction>

**[OGC]** Open Geospatial Consortium. URL: <http://www.ogc.org>

**[GML]** OGC Inc. Geography Markup Language (GML). URL: <http://www.opengeospatial.org/standards/gml>

**[OGC11]** OGC Inc. Simple Feature Access – Part 1: Common Architecture, 2011. URL: [http://portal.opengeospatial.org/files/?artifact\\_id=25355](http://portal.opengeospatial.org/files/?artifact_id=25355)

**[OGC12]** OGC Inc. GeoSPARQL Standard – A Geographic Query Language for RDF Data, 2012. URL: [https://portal.opengeospatial.org/files/?artifact\\_id=47664](https://portal.opengeospatial.org/files/?artifact_id=47664)

**[Parliament]** BBN Technologies. Parliament RDF. URL: <http://parliament.semwebcentral.org/>



- [PAGA14]** K. Patroumpas, M. Alexakis, G. Giannopoulos, and S. Athanasiou. TripleGeo: an ETL Tool for Transforming Geospatial Data into RDF Triples. In *LWDM*, pp. 275-278, 2014.
- [Per12]** A. Perego. Inspiring Data? Cross-domain Interoperability for EU Spatial Data. In *Using Open Data Workshop*, Brussels, Belgium, June 2012.
- [Per14]** A. Perego. INSPIRE metadata and DCAT-AP: Mapping summary. *Personal communication*, 2 April 2014.
- [PL14]** A. Perego and M. Lutz. Interoperable Registers and Registries in the EU: Perspectives from INSPIRE. In *Linking Geospatial Data Workshop*, London, UK, 2014.
- [PostGIS]** PostGIS - Spatial and Geographic objects for PostgreSQL. URL: <http://postgis.net/>
- [RWB12]** J. Reid, W. Waites, and B. Butchart. An Infrastructure for Publishing Geospatial Metadata as Open Linked Metadata. In *AGILE*, 2012.
- [RDF]** Resource Description Framework Primer. URL: <http://www.w3.org/TR/rdf-primer/>
- [Saxon]** Saxon XSLT and XQuery Processor. URL: <http://saxon.sourceforge.net/>
- [SL10]** S. Schade and M. Lutz. Opportunities and Challenges for using Linked Data in INSPIRE. In *Workshop on Linked Spatiotemporal Data*, 2010.
- [SLP+14]** R. Smith, M. Lutz, A. Perego, A. Friis-Christensen, A. Vasilescu, J. Rodrigues Frade, D. Vandenbroucke, and D. Tirry. RDF and PIDs for INSPIRE: a missing item in ARE3NA. In *Linking Geospatial Data Workshop*, March 2014.
- [SPARQL]** SPARQL 1.1 Query Language. URL: <http://www.w3.org/TR/sparql11-query/>
- [TripleGeo]** Athena RC. TripleGeo utility. URL: <https://github.com/GeoKnow/TripleGeo>
- [TripleGeo-CSW]** Athena RC. TripleGeo-CSW middleware. URL: <https://github.com/GeoKnow/TripleGeo-CSW>
- [TSS11]** S. Tschirner, A. Scherp, and S. Staab. Semantic access to INSPIRE – How to publish and query advanced GML data. In *Terra Cognita*, pp. 75-87, 2011.
- [UK-URI]** UK Chief Technology Officer Council. Designing URI Sets for Location, May 2011. URL: <http://data.gov.uk/library/designing-uri-sets-for-location>
- [UML]** Object Management Group Inc. Unified Modeling Language. URL: <http://www.uml.org/>
- [Virtuoso]** OpenLink Software. Virtuoso Universal Server. URL: <http://virtuoso.openlinksw.com/>
- [Virtuoso71]** OpenLink Software. Geometric Support in Virtuoso 7.1. Press Release, 17/2/2014. URL: <http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VOSNews#Geometry%20Data%20Types>
- [VVS+10]** L.M. Vilches-Blázquez, B. Villazón-Terrazas, V. Saquicela, A. de León, O. Corcho, and A. Gómez-Pérez. GeoLinked Data and INSPIRE through an Application Case. In *ACM SIGSPATIAL GIS*, pp. 446-449, November 2010.
- [W3C]** World Wide Web Consortium (W3C). URL: <http://www.w3.org/>
- [Void]** W3C. Void Vocabulary (3/3/2011). URL: <http://www.w3.org/TR/void/>
- [ZarSPARQL]** Zaragoza municipality SPARQL endpoint. URL: <http://www.zaragoza.es/datosabiertos/sparql>

## Namespaces typically used as RDF prefixes in XSLT transformations and SPARQL queries

gmd: <http://www.isotc211.org/2005/gmd/>

gml: <http://www.opengis.net/gml/3.2>

geo: <http://www.opengis.net/ont/geosparql#>  
*properties*

*OGC GeoSPARQL datatypes & relational*

rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

rdfs: <http://www.w3.org/2000/01/rdf-schema#>

sf: <http://www.opengis.net/ont/sf#>

*OGC GeoSPARQL spatial functions*

xs: <http://www.w3.org/TR/2008/REC-xml-20081126#>

xsi: <http://www.w3.org/2001/XMLSchema-instance>

xsd: <http://www.w3.org/2001/XMLSchema#>

## Provisional *informal* namespaces for INSPIRE Data Themes (Annex I)

AD: <http://inspire.jrc.ec.europa.eu/schemas/ad/3.0/>

*(Addresses)*

AU: <http://inspire.jrc.ec.europa.eu/schemas/au/3.0/>

*(Administrative Units)*

BASE: <http://inspire.jrc.ec.europa.eu/schemas/base/3.2/>

*(Base Types)*

CP: <http://inspire.jrc.ec.europa.eu/schemas/cp/3.0/>

*(Cadastral Parcels)*

GN: <http://inspire.jrc.ec.europa.eu/schemas/gn/3.0/>

*(Geographical Names)*

HY: <http://inspire.jrc.ec.europa.eu/schemas/hy/3.0/>

*(Hydrography)*

HY-N: <http://inspire.jrc.ec.europa.eu/schemas/hy-n/3.0/>

*(Hydrographic Network)*

NET: <http://inspire.jrc.ec.europa.eu/schemas/net/3.2/>

*(Network features)*

PS: <http://inspire.jrc.ec.europa.eu/schemas/ps/3.0/>

*(Protected sites)*

TN: <http://inspire.jrc.ec.europa.eu/schemas/tn/3.0/>

*(Transportation Networks)*

TN-R0: <http://inspire.jrc.ec.europa.eu/schemas/tn-ro/3.0/>

*(Road Transportation Networks)*

## 9. Appendix

In this Appendix, we provide technical information regarding all transformations of data and metadata into INSPIRE-compliant RDF, as well as the implementation details of the TripleGeo-CSW middleware for discovering metadata from Catalogue Services. In addition, we offer several indicative queries that may be submitted to the SPARQL and GeoSPARQL endpoints that have been set up at the web interface.

All data (original GML or transformed into RDF/XML) and XSL stylesheets, along with instructions are available from:

[https://web.imis.athena-innovation.gr/redmine/projects/geoknow\\_public/wiki/Inspire2RDF](https://web.imis.athena-innovation.gr/redmine/projects/geoknow_public/wiki/Inspire2RDF)

Open-source tools TripleGeo and TripleGeo-CSW developed in this project are available at:

<https://github.com/GeoKnow/TripleGeo> and <https://github.com/GeoKnow/TripleGeo-CSW>

### 9.1 Transforming INSPIRE Data and Metadata into RDF

#### 9.1.1 Data Alignment to INSPIRE Schema

Data alignment to INSPIRE-compliant schemata for Data Themes from Annex I can be achieved using the **HUMBOLDT Alignment Editor** [HALE]. Since HALE comes with the respective INSPIRE schemata integrated into its platform, the user only needs to prepare the necessary mappings from its source schema (i.e., attributes in the local file containing the original data) to the target INSPIRE schema.

Data available in [geodata.gov.gr](http://geodata.gov.gr) come from various agencies and organizations of the Greek public sector and cover a wide range thematic categories and geographic areas. Since none of these datasets is currently INSPIRE-compliant, we have actually prepared such alignments using HALE for seven selected datasets, as listed in Table 4. All these alignment files, along with the original datasets (in GML) and their INSPIRE-aligned data (also in GML) are available at [INSPIRE2RDF].

**Table 4: HALE alignments to INSPIRE themes available for datasets from [geodata.gov.gr](http://geodata.gov.gr).**

INSPIRE Data Theme	Original dataset (.gml, .xsd) from <a href="http://geodata.gov.gr">geodata.gov.gr</a>	HALE Alignment file	INSPIRE-aligned dataset (.gml)
[GN]	oikismoi	GeographicalNames.hale	GR_GeographicalNames_Settlements
[AU]	oria_dhmwn_kallikraths	AdministrativeUnits.hale	GR_AdministrativeUnits_Kallikratis
[AD]	arithmhsh_dromwn	Address.hale	GR_Addresses_Kalamaria
[CP]	oikodomika_tetragwna	CadastralParcels.hale	GR_CadastralParcels_Kalamaria
[TN]	odikoi_axones	TransportNetwork.hale	GR_RoadTransportNetwork_Kalamaria

[HY]	ydrografiko_diktyo	Hydrography.hale	GR_HydroNetwork
[PS]	natura	ProtectedSites.hale	GR_ProtectedSites_Natura2000

Provided that open-source HALE software has been installed, these alignments may be freely reused or modified according to the data schema at hand. The following instructions indicate how to load in HALE and reuse these alignments:

- 1) Start HALE application.
- 2) From the menu, go to: File/Open Alignment Project...
- 3) Select a .hale file from those available for the seven data themes.
- 4) The editor will cause an error: "Loading Error, Can't find file:/C/.../\*.xsd"
- 5) Click OK and select the schema (.xsd) from your system.
- 6) Make any changes at Schema Explorer, in order to reflect your own schema.
- 7) From the menu, go to: File/Import/Source data and select the data you wish to align.
- 8) From the menu, go to: File/Export/Trasformed data and select the Export format.

## 9.1.2 Transforming INSPIRE Metadata into RDF

A parametrized XSL stylesheet is employed to transform INSPIRE-compliant metadata from XML into RDF. This script can be executed with any XSLT parser and takes as input a metadata XML file according to ISO specifications as explained in Section 3. It returns an RDF/XML file with geometries in GeoSPARQL WKT serializations. A variant has been also implemented in order to return geometries according to Virtuoso specifications, due to differing prefixes from the GeoSPARQL standard [GeoKnowD21].

Regarding metadata, this application profile is based on RDF mappings suggested by JRC [Per14], and covers all elements in the schema. This Metadata2RDF.xsl is listed below in Figure 10. It is customized and it actually requires specification of three important parameters:

- the resourceURI, which points to the location of the actual dataset (i.e., where its GML file can be found);
- the metadataURI, which identifies the location of the metadata (i.e., the XML metadata file);
- language, which specifies the respective tag for all string literals.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/TR/2008/REC-xml-20081126#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:dct="http://purl.org/dc/terms/"
  xmlns:dcatt="http://www.w3.org/ns/dcat#"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:gmd="http://www.isotc211.org/2005/gmd"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:locn="http://www.w3.org/ns/locn#"
  xmlns:prov="http://www.w3.org/ns/prov#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/1999/01/rdf-schema#"
  xmlns:schema="http://schema.org"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:vcad="http://www.w3.org/2006/vcard/ns#"
  xmlns:gml="http://www.opengis.net/gml/"
```

```

xmlns:gco="http://www.isotc211.org/2005/gco" >

<xsl:strip-space elements="*" />
<xsl:output method="xml" indent="yes" />
<xsl:template match="*">

<rdf:RDF>

  <xsl:for-each select="//gmd:MD_Metadata" >
    <xsl:variable name="pass"
select='gmd:identificationInfo//gmd:geographicElement//gmd:westBoundLongitude' />
    <xsl:if test="$pass > -179 " >
      <!-- Metadata identifier -->
      <xsl:variable name="id" select='gmd:fileIdentifier' />
      <xsl:variable name="resourceURI" select="gmd:distributionInfo//gmd:linkage/gmd:URL"/>
      <!-- <xsl:variable name='resourceURI'
select="concat('http://geodata.gov.gr/datasets/attachments/', $id)" /> -->

      <xsl:variable name="metadataURI" select="concat('urn:uuid:', $id)" />
      <!-- <xsl:variable name='metadataURI'
select="concat('http://geodata.gov.gr/datasets/data/', $id, '/')" /> -->

      <!-- common language -->
      <xsl:variable name="language" select='gmd:identificationInfo//gmd:language' />
      <!-- <xsl:variable name='language' select="'el'" /> -->

      <!-- Metadata on Metadata -->
      <rdf:Description rdf:about="{ $metadataURI}">
        <foaf:primaryTopic rdf:resource="{ $resourceURI}" />
        <!-- Metadata Language -->
        <dc:language rdf:datatype="http://purl.org/dc/terms/ISO639-2"><xsl:value-
of select='gmd:language' /></dc:language>

        <!-- Metadata Date -->
        <dc:modified
rdf:datatype='http://www.w3.org/2001/XMLSchema#date'><xsl:value-of
select='gmd:dateStamp' /></dc:modified>

        <xsl:variable name="mail" select='gmd:contact//gmd:electronicMailAddress' />
        <!-- Metadata Point of Contact -->
        <dc:creator>
          <foaf:Organisation>
            <foaf:name xml:lang="{ $language}"><xsl:value-of
select='gmd:contact//gmd:organisationName' /></foaf:name>
            <foaf:mbox rdf:resource="{concat('mailto:', $mail)}" />
          </foaf:Organisation>
        </dc:creator>
      </rdf:Description>

      <!-- Resource Metadata -->
      <rdf:Description rdf:about="{ $resourceURI}">

        <foaf:primaryTopicOf rdf:resource="{ $metadataURI}" />

        <!-- Resource Language -->
        <dc:language rdf:datatype="http://purl.org/dc/terms/ISO639-2"><xsl:value-
of select='(gmd:identificationInfo//gmd:language)[1]' /></dc:language>

        <!-- Resource Title -->
        <dc:title xml:lang="{ $language}"><xsl:value-of
select='gmd:identificationInfo//gmd:title/gco:CharacterString' /></dc:title>

        <!-- Resource Abstract -->

```

```

        <dc:description xml:lang='{ $language }'><xsl:value-of
select='//gmd:abstract' /></dc:description>

        <!-- Resource Type -->
        <rdf:type rdf:resource='http://www.w3.org/ns/dcat#Dataset' />
        <!-- Resource Locator-->
        <dcat:landingPage
rdf:resource='{gmd:distributionInfo//gmd:linkage/gmd:URL}' />

        <!-- Unique Resource Identifier -->
        <dc:identifier
rdf:datatype='http://www.w3.org/2001/XMLSchema#string'><xsl:value-of
select='gmd:identificationInfo//gmd:codeSpace' /></dc:identifier>

        <!-- Topic Category -->
        <dc:subject><xsl:value-of select='//gmd:MD_TopicCategoryCode'
/></dc:subject>

        <!-- Free Keywords-->
        <xsl:for-each select='gmd:identificationInfo//gmd:keyword'>
            <dcat:keyword xml:lang='{ $language }'><xsl:value-of
select='.' /></dcat:keyword>
        </xsl:for-each>

        <!-- Check for controlled vocabulary like Thesaurus, GEMET -->
        <xsl:if test='//gmd:thesaurusName'>
            <dcat:theme>
                <skos:Concept>
                    <skos:inScheme>
                        <skos:ConceptScheme>
                            <rdfs:label><xsl:value-of
select='//gmd:thesaurusName//gmd:title' /></rdfs:label>
                            <dc:issued
rdf:datatype='http://www.w3.org/2001/XMLSchema#date'>
                                <xsl:value-of
select='//gmd:thesaurusName//gco:Date' />
                            </dc:issued>
                        </skos:ConceptScheme>
                    </skos:inScheme>
                </skos:Concept>
            </dcat:theme>
        </xsl:if>

        <!-- Geographic Bounding Box, GeoSparql Specs -->
        <xsl:variable name='MAXX'
select='gmd:identificationInfo//gmd:extent//gmd:eastBoundLongitude' />
        <xsl:variable name='MINX'
select='gmd:identificationInfo//gmd:extent//gmd:westBoundLongitude' />
        <xsl:variable name='MAXY'
select='gmd:identificationInfo//gmd:extent//gmd:northBoundLatitude' />
        <xsl:variable name='MINY'
select='gmd:identificationInfo//gmd:extent//gmd:southBoundLatitude' />

        <!-- BBOX according to GeoSparql Specs -->
        <dc:spatial>
            <dc:Location>
                <locn:geometry
rdf:datatype='http://www.opengis.net/ont/sf#wktLiteral'>
                    <xsl:value-of select='concat("POLYGON (('", $MINX, ' ', $MINY, ' ', $MAXX, '
', $MINY, ' ', $MAXX, ' ', $MAXY, ' ', $MINX, ' ', $MAXY, ' ', $MINX, ' ', $MINY, ' '))')' />
                </locn:geometry>
            </dc:Location>
        </dc:spatial>

```



```

        <!-- Create Date elements -->
        <xsl:for-each
select='gmd:identificationInfo//gmd:citation//gmd:date/gmd:CI_Date' >
            <xsl:if test='gmd:dateType/gmd:CI_DateTypeCode[@codeListValue =
"publication"]' >
                <!-- Date of publication -->
                < dct:issued rdf:datatype='http://www.w3.org/2001/XMLSchema#date'>
                    <xsl:value-of select='gmd:date/gco:DateTime' />
                </dct:issued>
            </xsl:if>

            <xsl:if test='gmd:dateType/gmd:CI_DateTypeCode[@codeListValue =
"revision"]' >
                <!-- Date of last revision -->
                < dct:modified rdf:datatype='http://www.w3.org/2001/XMLSchema#date'>
                    <xsl:value-of select='gmd:date/gco:DateTime' />
                </dct:modified>
            </xsl:if>

            <xsl:if test='gmd:dateType/gmd:CI_DateTypeCode[@codeListValue =
"creation"]' >
                <!-- Date of creation -->
                < dct:created rdf:datatype='http://www.w3.org/2001/XMLSchema#date'>
                    <xsl:value-of select='gmd:date/gco:DateTime' />
                </dct:created>
            </xsl:if>
        </xsl:for-each>

        <!-- Lineage -->
        < dct:provenance>
            < dct:ProvenanceStatement>
                < rdfs:label xml:lang='{ $language }'><xsl:value-of
select='//gmd:LI_Lineage/gmd:statement/gco:CharacterString' /></rdfs:label>
            </dct:ProvenanceStatement>
        </dct:provenance>

        <!-- Conditions for access and use -->
        < dcat:distribution>
            < dcat:Distribution>
                < dct:rights>
                    < dct:RightsStatements>
                        < rdfs:label><xsl:value-of
select='//gmd:useLimitation/gco:CharacterString' /></rdfs:label>
                    </dct:RightsStatements>
                </dct:rights>
                <!-- Limitations on public access -->
                < dct:accessRights>
                    < dct:RightsStatement>
                        < rdfs:label><xsl:value-of
select='//gmd:otherConstraints/gco:CharacterString' /></rdfs:label>
                    </dct:RightsStatement>
                </dct:accessRights>
            </dcat:Distribution>
        </dcat:distribution>

        <!-- Responsible Organization -->
        <!-- role owner-->
        <xsl:variable name='mail2'
select='gmd:identificationInfo//gmd:pointOfContact//gmd:electronicMailAddress' />
        < dct:rightsHolder>
            < foaf:Organisation>

```

```

        <foaf:name xml:lang='{$language}'><xsl:value-of
select='(//gmd:organisationName/gco:CharacterString)[1]'/></foaf:name>
        <foaf:mbox rdf:resource="{concat('mailto:',$mail2)}" />
    </foaf:Organisation>
</dct:rightsHolder>
<!-- Resource provider -->
<prov:qualifiedAttribution>
    <prov:Attribution>
        <prov:agent>
            <vcard:Kind>
                <vcard:organization-name
xml:lang='{$language}'><xsl:value-of
select='(//gmd:organisationName/gco:CharacterString)[1]'/></vcard:organization-name>
                <vcard:hasEmail
rdf:resource="{concat('mailto:',$mail2)}"/>
            </vcard:Kind>
        </prov:agent>
    </dct:type>
    <vcard:organization-name
rdf:resource='http://inspire.ec.europa.eu/codelist/ResponsiblePartyRole/resourceProvider'/>
    </prov:Attribution>
</prov:qualifiedAttribution>

<!-- Role- point of contact -->
<dcat:contactPoint>
    <vcard:Kind>
        <vcard:organization-name xml:lang='{$language}'><xsl:value-of
select='(//gmd:organisationName/gco:CharacterString)[1]'/></vcard:organization-name>
        <vcard:hasEmail rdf:resource="{concat('mailto:',$mail2)}"/>
    </vcard:Kind>
    </dcat:contactPoint>
    <prov:qualifiedAttribution>
        <prov:Attribution>
            <prov:agent>
                <vcard:Kind>
                    <vcard:organization-name
xml:lang='{$language}'><xsl:value-of
select='(//gmd:organisationName/gco:CharacterString)[1]'/></vcard:organization-name>
                    <vcard:hasEmail
rdf:resource="{concat('mailto:',$mail2)}"/>
                </vcard:Kind>
            </prov:agent>
        </dct:type>
        <vcard:organization-name
rdf:resource='http://inspire.ec.europa.eu/codelist/ResponsiblePartyRole/pointOfContact'/>
        </prov:Attribution>
    </prov:qualifiedAttribution>

<!-- Conformity -->
<dct:conformsTo>
    <dct:Standard>
        <dct:title xml:lang='en'><xsl:value-of
select='//gmd:report//gmd:title/gco:CharacterString' /></dct:title>
        <dct:issued
rdf:datatype='http://www.w3.org/2001/XMLSchema#date'><xsl:value-of
select='//gmd:report//gco:Date' /></dct:issued>
    </dct:Standard>
</dct:conformsTo>

</rdf:Description>

</xsl:if>
</xsl:for-each>

</rdf:RDF>
</xsl:template>

```

```
</xsl:stylesheet>
```

**Figure 10: XSL stylesheet Metadata2RDF.xsl for transforming INSPIRE metadata into RDF.**

#### 9.1.2.1 Execution of the XSLT Transformation

This XSL stylesheet can be used with *any XSLT parser* in order to transform spatial metadata elements from XML into INSPIRE-compliant RDF/XML. For example, using Saxon XSLT parser [Saxon], metadata about protected sites (*input file: natura.xml*) can be transformed into RDF/XML (*output file: GeoSPARQL\_Metadata\_ps\_natura2000\_GR.rdf*) as follows:

```
java -cp .;saxon9he.jar net.sf.saxon.Transform -s:natura.xml -xsl:Metadata2RDF.xsl -o:
GeoSPARQL_Metadata_ps_natura2000_GR.rdf
```

For convenience, this stylesheet has been integrated into the TripleGeo suite [TripleGeo], and its source code is also available from [INSPIRE2RDF]. A new version 1.1 of TripleGeo is publicly available, offering the entire Java source code as well as a .JAR package that contains executable binaries. Java JRE (or SDK) 1.7 (or later) must be installed and properly configured in order to execute TripleGeo from its binaries. The tool has been successfully tested in both MS Windows and Linux environments. As documented in Deliverable D2.2.1 [GeoKnowD22], TripleGeo has been implemented with several Java classes that perform specific tasks in a modular fashion. From a user's perspective, the utility works in an opaque fashion according to some preconfigured settings. In terms of output serializations, the triples can be obtained in RDF/XML format. This is the default output format that represents RDF as XML, according to the RDF specifications. In terms of standardization, the output triples are conformant to W3C standards, thanks to methods provided by the underlying Jena API for creating resources, properties and literals and the statements linking them. Therefore, all output triples are compatible with the most commonly used standards, including RDF, RDFS, OWL, and SPARQL.

Users may edit stylesheet Metadata2RDF.xsl in order to define suitable values for the parameters according to the metadata at hand, and then perform an XSLT transformation to obtain the resulting RDF files. In order to use TripleGeo for extracting triples from a metadata file, the user should follow these steps:

- Open a terminal window and navigate to the directory where TripleGeo has been extracted. Normally, this folder includes a lib/ subdirectory with the required libraries.
- Verify that Java JRE (or SDK) version 1.7 or later is installed. Currently installed version of Java can be checked using: `java -version` from the command line.
- Check that all properties are correctly configured in the necessary XSL stylesheets, as explained above (i.e., values for resourceURI, metadataURI, language).
- Assuming that Java binaries are bundled together in TripleGeo.jar, the general pattern of the command that performs this transformation through TripleGeo is as follows:

```
java -cp lib/*;TripleGeo.jar eu.geoknow.athenarc.triplegeo.MetadataToRdf input.xml output.rdf
```

For example, if someone uses TripleGeo to transform metadata regarding protected sites (contained in file natura.xml), then the following command may be invoked in order to get the resulting RDF file:

```
java -cp lib/*;TripleGeo.jar eu.geoknow.athenarc.triplegeo.MetadataToRdf natura.xml
GeoSPARQL_Metadata_ps_natura2000_GR.rdf
```

As soon as processing is finished and all triples are written into the RDF file, the user is notified about the total amount of extracted triples and the overall execution time. Then, the resulting RDF/XML representation can be readily loaded into a triple store.

### 9.1.3 Implementation of the TripleGeo-CSW Middleware

As explained in Section 4, Catalogue Services for the Web (CSW) support the ability to publish and search collections of descriptive information (metadata) for data, services, and related information objects. Metadata in CSW catalogues represent resource characteristics that can be queried and presented for evaluation and further processing by both humans and software.

In order to facilitate discovery of INSPIRE data from such CSWs through SPARQL, we have implemented the **TripleGeo-CSW middleware**. This middleware has been developed in Python 2.7.3, and it makes use of other modules:

- `urllib2`, a Python module for fetching URLs (Uniform Resource Locators) and posing requests;
- `etree`, for xml parsing using the concepts of the ElementTree API for Python;
- `re`, which provides Perl-style *regular expression* pattern matching and is being used when parsing such expressions in users' SPARQL requests.

#### 9.1.3.1 Parsing of SPARQL Queries

Each SPARQL query that can be handled by this middleware consists of two parts:

- a **SELECT** clause, which identifies the variables to appear in the query results, and
- a **WHERE** clause, which provides the basic graph pattern to match against the data graph. A graph pattern in a **WHERE** clause consists of the subject, predicate and object of triples to find a match for in the data.

Parsing correctly such a query and analyzing its components is crucial for exposing CSWs as RDF resources. Such queries may be quite complex, as they could involve several conditions in the **WHERE** or even a **FILTER** clause. Consider the following example:

```
SELECT *
WHERE {
  ?a dc:subject "environment" .      (1)
  ?b dc:title ?t .                  (2)
  ?m geo:hasGeometry ?fwKT .        (3)
  FILTER ( REGEX(str(?t), "%network%") (4A)
    && geof:sfWithin(?fwKT,"BOX2D(-6.51 53.27, -6.28 53.36)""^geo:wktLiteral)) (4B)
}
```

The query searches all (\*) available results by binding triples qualifying for subject, title and geometry. As illustrated with the red boxes, there are three triple patterns in the **WHERE** clause and two filtering criteria for those patterns.

- Triple pattern (1) searches all metadata where the element subject has value "environment".
- Pattern (2) has a **REGEX** (Regular Expression) filtering criterion. Thus, if any element title contains (%) the string value "network", it will be matched.

- Pattern (3) is a geometry pattern. It searches for datasets whose coverage area (expressed as a box) falls within (spatial operator `sfWithin`) the BoundingBox (BOX2D) with the given coordinates.

Overall, this query should fetch metadata matching all three conditions, as specified in the WHERE clause. Note that any filtering criteria not only make the query more difficult to parse, but increase its complexity as well. Analyzing this query line by line, regarding all conditions specified in the WHERE clause, it is possible to recognize all interesting components:

(1): `?a dc:subject "environment" .`  
→ `dc:subject = environment`

(2): `?b dc:title ?t .`  
→ `dc:title = ?t`

(3): `?m geo:hasGeometry ?fWKT .`  
→ `geo:hasGeometry = ?WKT`

(4A): `REGEX ( str(?t), "%network%" )`  
→ `?t = %network%`  
`dc:title = ?t`  
`dc:title = %network%`

(4B): `geof:sfWithin( ?fWKT, "BOX2D(-6.51 53.27, -6.28 53.36)"^^geo:wktLiteral)`  
→ `?WKT = BOX2D(-6.51 53.27, -6.28 53.36)`  
`sfWithin(?WKT)`  
`geo:hasGeometry = sfWithin( BOX2D(-6.51 53.27, -6.28 53.36) )`

In essence, the result of such parsing must be a definite mapping between elements and values, as shown in Table 5 for this particular example.

**Table 5: Mapping pairs of elements to particular values.**

Element	Value
<code>dc:subject</code>	<code>environment</code>
<code>dc:title</code>	<code>%network%</code>
<code>geo:hasGeometry</code>	<code>sfWithin( BOX2D(-6.51 53.27, -6.28 53.36) )</code>

A graph pattern in a SPARQL WHERE clause consists of a subject, a predicate and an object triple. So, every triple pattern has a specific format like `?s ?p ?o`, where `?p` is the element we are looking for, and `?o` is either its value or a binding variable. So, we are searching for triples satisfying the matching patterns:

Case 1: `?s ?p ?o` OR Case 2: `?s ?p "literal"`.

In order to handle the matching results, we make use of one *List* and one *Dictionary* structure. The list is used for handling all the triples with a variable as an object (*Case 1*). On the other hand, the dictionary is an unordered set of key:value pairs with the requirement that the keys are unique, which means that in our case, is an unordered set of element:value pairs with unique

elements (Case 2). Assuming a valid SPARQL query has been submitted, we match the triple patterns regardless the format of query, i.e., whether it is well-formatted (line by line) or not.

The FILTER criteria are checked separately from the triple patterns. Supported criteria are: *regular expressions* (REGEX), *bounding boxes* (BOX2D), and *dates*. More specifically:

- **Regular Expression (REGEX)** provides an operation to test string matches with the following syntax:

```
FILTER regex(?x, "pattern" [, "flags"])
```

where argument flags is optional.

The first argument refers to an object in the List. Consequently, if this argument is found in the List, the second argument “pattern” in REGEX is the value of the element (second value in that pair in the List), signifying Case 2.

- **BOX2D.** The important part in this geographic rectangle refers to the coordinate bounds (two antidiagonal points), which are enclosed inside the parentheses. So, if a BOX2D is matched in any line, the content inside parentheses is retrieved. The first two coordinates are assigned to the lowerCorner and the other two to the upperCorner. Apart from the rectangle bounds, the middleware can handle topological operators as well. So, if any spatial operator (Equals, Overlaps, Disjoint, Intersects, Touches, Crosses, Within, Contains) [OGC11] is specified along with the box, it is also used in the subsequent CSW request. If no spatial operator is being specified with the box, the default is BBOX. This is why BBOX functionality includes both Within and Overlaps predicates. As a result, the lowerCorner and upperCorner of a BBOX, as well as a Spatial Operator are located in a list named box.
- **Date.** A common queryable element is date. Date functionality depends on comparison operators (>,<,<=,>=) and a constant value which follows a specific syntax like:

```
{ YYYY-MM-DD } ex. ?date > "2000-01-01"^^xsd:date
```

All the triple patterns in the List structure are being compared with the elements in the line where the object is found. The comparison operator and the date are inserted into the dictionary as value.

### 9.1.3.2 Formulating CSW Requests

As soon as the query is analyzed and all its criteria and filters have been identified, the respective CSW GetRecords POST request must be formulated. All necessary parametrization for this request is available in the Dictionary or/and box list and will be used for this request. Of course, the major concern is the <Filter> element in that CSW request, since this is the one that controls whether metadata should be obtained according to specific criteria. This is carried out with functions CreateLikeXML() and Request() functions, and the final CSW request includes properly tagged elements, as indicated in Table 6. A typical CSW request (including a spatial condition) is shown in Figure 5.

**Table 6: Elements supported in a CSW GetRecords request.**

Elements specified or returned	Tags
OGC comparison operators	ogc:PropertyIsEqualTo, ogc:PropertyIsLessThan, ogc:PropertyIsGreaterThan, ogc:PropertyIsLike
OGC logical operators	ogc:Or, ogc:And
OGC Filter Spatial Predicate	ogc:BBOX, ogc:Equals, ogc:Disjoint, ogc:Crosses, ogc:Contains, ogc:Touches, ogc:Intersects, ogc:Within, ogc:Overlaps



OGC geometry operands	<code>gml:Envelope</code>
Returnables	<code>dc:identifier, dc:title, dc:type, ows:BoundingBox, dc:subject, dct:modified, dc:abstract</code>

### 9.1.3.3 Executing the TripleGeo-CSW Middleware

As mentioned in Section 6.4, the TripleGeo-CSW middleware has been integrated into the web interface and it is available for use against a set of predefined CSW services across Europe (Section 4.2).

But this tool (implemented in python) can also work in standalone mode, by using the open-source code available at [TripleGeo-CSW]. First, one must make sure all requirements are properly installed. This applies to both command-line usage and the WSGI application:

```
pip install -r pip-requirements.txt
```

In order to start the TripleGeo-CSW Middleware, you must edit your configuration at `config.ini` and adjust it to your needs (host, port, CSW endpoints etc.). Afterwards, you just have to start the WSGI application:

```
python wsgi.py
```

Given a SPARQL query in file `q1.sparql`, an example command-line invocation would be:

```
python query.py input/q1.sparql
```

To keep error/output streams separated, invoke as:

```
python query.py input/q1.sparql 2>err.log 1>out.rdf
```

To send a query to the WSGI service running at 127.0.0.1:5000:

```
curl -X POST http://localhost:5000/query -d @input/q0-post.txt
```

### 9.1.4 Transforming INSPIRE Data into RDF

Once an INSPIRE-aligned GML dataset is available for a Data Theme of Annex I, its RDF representation can be produced by using a set of customized XSL stylesheets. Each such stylesheet reflects the respective INSPIRE schema of the seven themes (e.g., Addresses, Cadastral Parcels, etc.), as detailed in Table 7:

**Table 7: XSL stylesheets developed for INSPIRE Data Themes of Annex I.**

INSPIRE Data Theme	XSL stylesheet
[GN] <i>Geographical names</i>	<code>InspireGN2RDF.xsl</code>
[AU] <i>Administrative units</i>	<code>InspireAU2RDF.xsl</code>
[AD] <i>Addresses</i>	<code>InspireAD2RDF.xsl</code>
[CP] <i>Cadastral parcels</i>	<code>InspireCP2RDF.xsl</code>
[TN] <i>Transport networks (Road network)</i>	<code>InspireTN-R02RDF.xsl</code>
[HY] <i>Hydrography</i>	<code>InspireHY2RDF.xsl</code>
[PS] <i>Protected sites</i>	<code>InspirePS2RDF.xsl</code>

Each stylesheet has dependencies on several others in order to handle several common elements, such as string literals used for naming various geographic entities (GeographicalNames.xsl), and particularly those concerning geometric representations (GML2WKT.xsl). Auxiliary stylesheets are listed in the following Table 8:

**Table 8: Auxiliary XSL stylesheets.**

Role	XSL stylesheet
Main stylesheet with user-defined customizations	Inspire_Main.xsl
Handling of various geometry representations	GML2WKT.xsl
Handling of multi-lingual names	GeographicalNames.xsl

Stylesheet Inspire\_Main.xsl is the main entry point for transforming INSPIRE-compliant data from GML into RDF. It is basically a wrapper of all other XSL scripts available, and allows the user to specify several parameters. Among others, these include the configuration of the URIs that will describe the resulting resources through:

- the *domain* where that resource belongs to (parameter: baseURI);
- the *theme* describes the Data Theme of Annex I (parameter: theme is extracted from elements);
- the *data source* refers to the particular data set provided by a stakeholder (parameter: topic);
- a *dataset identifier* is extracted by the respective gmlId found in the source file (parameter: datasetid);
- the *preferred WKT* allows extraction of geometries according to several serializations, e.g., GeoSPARQL or Virtuoso (parameter: typewKT).

Depending on the theme of available data, the user should enable the respective options regarding calls to the specific XSL stylesheets, as well as specify suitable values to the aforementioned parameters. For example, in case of data regarding protected sites (PS), the respective stylesheet (e.g., InspirePS2RDF.xsl) must be included. The complete listing of Inspire\_Main.xsl is given below in Figure 11 and it refers to handling of data regarding protected sites (other options are commented out):

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xs="http://www.w3.org/TR/2008/REC-xml-20081126#"
  xmlns:gml="http://www.opengis.net/gml/3.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:gmd="http://www.isotc211.org/2005/gmd/"
  xmlns:geo='http://www.opengis.net/ont/geosparql#'
  xmlns:ad="urn:x-inspire:specification:gmlas:Addresses:3.0"
  xmlns:AD="http://inspire.jrc.ec.europa.eu/schemas/ad/3.0/"
  xmlns:au="urn:x-inspire:specification:gmlas:AdministrativeUnits:3.0"
  xmlns:AU="http://inspire.jrc.ec.europa.eu/schemas/au/3.0/"
```

```

xmlns:base="urn:x-inspire:specification:gmlas:BaseTypes:3.2"
xmlns:BASE="http://inspire.jrc.ec.europa.eu/schemas/base/3.2/"
xmlns:cp="urn:x-inspire:specification:gmlas:CadastralParcels:3.0"
xmlns:CP="http://inspire.jrc.ec.europa.eu/schemas/cp/3.0/"
xmlns:gn="urn:x-inspire:specification:gmlas:GeographicalNames:3.0"
xmlns:GN="http://inspire.jrc.ec.europa.eu/schemas/gn/3.0/"
xmlns:hy="urn:x-inspire:specification:gmlas:HydroBase:3.0"
xmlns:HY="http://inspire.jrc.ec.europa.eu/schemas/hy/3.0/"
xmlns:hy-n="urn:x-inspire:specification:gmlas:HydroNetwork:3.0"
xmlns:HY-N="http://inspire.jrc.ec.europa.eu/schemas/hy-n/3.0/"
xmlns:net="urn:x-inspire:specification:gmlas:Network:3.2"
xmlns:NET="http://inspire.jrc.ec.europa.eu/schemas/net/3.2"
xmlns:ps="urn:x-inspire:specification:gmlas:ProtectedSites:3.0"
xmlns:PS="http://inspire.jrc.ec.europa.eu/schemas/ps/3.0/"
xmlns:tn="urn:x-inspire:specification:gmlas:CommonTransportElements:3.0"
xmlns:TN="http://inspire.jrc.ec.europa.eu/schemas/tn/3.0/"
xmlns:tn-ro="urn:x-inspire:specification:gmlas:RoadTransportNetwork:3.0"
xmlns:TN-RO="http://inspire.jrc.ec.europa.eu/schemas/tn-ro/3.0/" >

<xsl:strip-space elements="*" />
<xsl:output method="xml" indent="yes" />

<!-- Include other XSLs that handle subsets of the data -->
<xsl:include href="GML2WKT.xsl" />
<xsl:include href="GeographicalNames.xsl" />

<!-- EDIT: Include custom stylesheet for ONE specific data theme of INSPIRE Annex I -->
<!-- xsl:include href="InspireAD2RDF.xsl" / --> <!-- Addresses -->
<!-- xsl:include href="InspireAU2RDF.xsl" / --> <!-- Administrative Units -->
<!-- xsl:include href="InspireCP2RDF.xsl" / --> <!-- Cadastral Parcels -->
<!-- xsl:include href="InspireGN2RDF.xsl" / --> <!-- GeographicalNames -->
<!-- xsl:include href="InspireHY2RDF.xsl" / --> <!-- Hydrography (Network) -->
<xsl:include href="InspirePS2RDF.xsl" /> <!-- Protected Sites -->
<!-- xsl:include href="InspireTN-R02RDF.xsl" / --> <!-- Transport Networks (Roads) -->

<!-- EDIT: Specify ONE data topic to be included in URIs of the resulting RDF triples -->
<!-- xsl:variable name='topic'><xsl:value-of>AddressesGR</xsl:value-of></xsl:variable -->
<!-- xsl:variable name='topic'><xsl:value-of>Kallikratis</xsl:value-of></xsl:variable -->
<!-- xsl:variable name='topic'><xsl:value-of>CadastralParcelsGR</xsl:value-
of></xsl:variable -->
<!-- xsl:variable name='topic'> <xsl:value-of>GeoNamesGR</xsl:value-of> </xsl:variable -->

```

```

<!-- xsl:variable name='topic'>xsl:value-of>RiversGR</xsl:value-of></xsl:variable -->
<xsl:variable name='topic'>xsl:value-of>Natura2000GR</xsl:value-of></xsl:variable>
<!-- xsl:variable name='topic'>xsl:value-of>RoadsGR</xsl:value-of></xsl:variable -->

<!--EDIT: Compose base URI for all RDF entities and the identifier of output RDF data -->
<xsl:variable name='baseURI' select="concat('http://geodata.gov.gr/', 'id/')" />
<xsl:variable name="datasetid"> <xsl:value-of select="base:SpatialDataSet/@gml:id"/>
</xsl:variable>

<xsl:variable name="typeWKT"> <xsl:value-of select=
"concat('http://www.opengis.net/ont/geosparql#', 'wktLiteral')" /></xsl:variable>

<!-- Matching starts from here -->
<xsl:template match="/">
  <rdf:RDF>
    <!-- Export boundary (i.e., geographic extent) of the entire dataset -->
    <!-- (if included in the header of the original dataset) -->
    <!-- Using a provisional URI for RDF triple of this polygon geometry -->
    <xsl:if test="base:SpatialDataSet/gml:boundedBy">
      <rdf:Description>
        <xsl:attribute name='rdf:about'>
          <xsl:value-of select="concat($baseURI, 'boundary/',
$topic, '/', $datasetid)" />
        </xsl:attribute>
        <!-- Rectangle extent of dataset as GeoSPARQL WKT literal -->
        <xsl:element name='geo:asWKT'>
          <xsl:attribute name='rdf:datatype'>
            <xsl:value-of select='$typeWKT' />
          </xsl:attribute>
          <xsl:value-of><xsl:apply-templates
select="base:SpatialDataSet/gml:boundedBy" /></xsl:value-of>
        </xsl:element>
      </rdf:Description>
    </xsl:if>

    <!-- Handle all other tags as specified by the custom templates -->
    <!-- and for each feature create its RDF representation -->
    <xsl:apply-templates select="//base:member/*" />
  </rdf:RDF>
</xsl:template>
</xsl:stylesheet>

```

**Figure 11: XSL stylesheet Inspire\_Main.xsl for transforming INSPIRE data into RDF.**

Regarding 2-dimensional OGC geometries, the following stylesheet GML2WKT.xsl is used to convert from Geometry Markup Language (GML) to Well Known Text (WKT) representations according to the recent GeoSPARQL standard [OGC12]. As it can be observed from the next listing in Figure 12, this script can cope with a wide range of geometric shapes, including not only basic geometric types (such as point, linestring, polygon), but also more complex ones (such as multipoint, multilinestring, multipolygon), and even geometry collections. In addition, it can identify the spatial reference system of every geometry in order to provide a complete WKT representation for the resulting RDF dataset.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:gml="http://www.opengis.net/gml/3.2" >
  <xsl:output method="xml" indent="yes"/>
  <xsl:strip-space elements="*" />

  <!-- Compose the base URI for georeference systems (SRS), according to EPSG catalogue -->
  <xsl:variable name='epsg' select="concat('http://www.opengis.net/def/crs/EPSG/', '0/')" />

  <!-- Use metadata with extreme corners of the bounding box for the dataset and construct a
  polygon equivalent to this rectangle. -->
  <xsl:template match="gml:boundedBy">
    <xsl:apply-templates /> <!-- May involve either a gml:Box (older
    specification) or a gml:Envelope (latest OGC standard) -->
  </xsl:template>

  <xsl:template match="gml:Box">
    <xsl:param name="separator" select="' ' ' ' />
    <xsl:call-template name="gml:srsName"/>
    <xsl:variable name="coords"
    select="tokenize(
      replace(replace(gml:coordinates, '\s$', ''), '^(\s+', ''), '[\s\n\r]+'))" />
    <xsl:variable name="MINX"><xsl:value-of select="$coords[1]"/></xsl:variable>
    <xsl:variable name="MINY"><xsl:value-of select="$coords[2]"/></xsl:variable>
    <xsl:variable name="MAXX"><xsl:value-of select="$coords[3]"/></xsl:variable>
    <xsl:variable name="MAXY"><xsl:value-of select="$coords[4]"/></xsl:variable>
    <xsl:value-of select="concat('POLYGON((' , $MINX, ' ', $MINY, ' ', ' ', $MAXX, ' ',
    $MINY, ' ', ' ', $MAXX, ' ', $MAXY, ' ', ' ', $MINX, ' ', $MAXY, ' ', ' ', $MINX, ' ', $MINY, ' '))' )"/>
  </xsl:template>

  <xsl:template match="gml:Envelope">
    <xsl:param name="separator" select="' ' ' ' />
    <xsl:call-template name="gml:srsName"/>
    <xsl:variable name="MINX"><xsl:value-of select="normalize-space(substring-
    before(gml:lowerCorner, $separator))"/></xsl:variable>
```

```

        <xsl:variable name="MINY"><xsl:value-of select="normalize-space(substring-
after(gml:lowerCorner, $separator))"/></xsl:variable>

        <xsl:variable name="MAXX"><xsl:value-of select="normalize-space(substring-
before(gml:upperCorner, $separator))"/></xsl:variable>

        <xsl:variable name="MAXY"><xsl:value-of select="normalize-space(substring-
after(gml:upperCorner, $separator))"/></xsl:variable>

        <xsl:value-of select="concat('POLYGON((' , $MINX, ' ', $MINY, ' ', $MAXX, ' ',
$MINY, ' ', $MAXX, ' ', $MAXY, ' ', $MINX, ' ', $MAXY, ' ', $MINX, ' ', $MINY, ' '))')"/>
    </xsl:template>

    <xsl:template match="gml:Point">
        <xsl:call-template name="gml:srsName"/>
        <xsl:text>POINT</xsl:text>
        <xsl:call-template name="gml:posList"/>
    </xsl:template>

    <xsl:template match="gml:LineString">
        <xsl:call-template name="gml:srsName"/>
        <xsl:text>LINESTRING</xsl:text>
        <xsl:call-template name="gml:posList"/>
    </xsl:template>

    <xsl:template match="gml:Polygon">
        <xsl:call-template name="gml:srsName"/>
        <xsl:text>POLYGON</xsl:text>
        <xsl:apply-templates select="gml:exterior|gml:outerBoundaryIs"/>
        <xsl:apply-templates select="gml:interior|gml:innerBoundaryIs"/>
        <xsl:text>)</xsl:text>
    </xsl:template>

    <xsl:template match="gml:MultiPoint">
        <xsl:call-template name="gml:srsName"/>
        <xsl:text>MULTIPOINT</xsl:text>
        <xsl:for-each select="gml:pointMember//gml:Point|gml:pointMembers//gml:Point">
            <xsl:if test="not(position()=1)">
                <xsl:text>,</xsl:text>
            </xsl:if>
            <xsl:apply-templates />
        </xsl:for-each>
        <xsl:text>)</xsl:text>
    </xsl:template>

```



```
<xsl:template match="gml:MultiLineString">
  <xsl:call-template name="gml:srsName"/>
  <xsl:text>MULTILINESTRING(</xsl:text>
  <xsl:for-each select="gml:lineStringMember">
    <xsl:call-template name="gml:lineStringMember"/>
    <xsl:if test="(position() != last())">
      <xsl:text>,</xsl:text>
    </xsl:if>
  </xsl:for-each>
  <xsl:text>)</xsl:text>
</xsl:template>
```

```
<xsl:template match="gml:Curve">
  <xsl:call-template name="gml:srsName"/>
  <xsl:for-each select="gml:segments//gml:LineStringSegment">
    <xsl:choose>
      <xsl:when test="not(position()=1)">
        <xsl:text>,</xsl:text>
      </xsl:when>
      <xsl:otherwise>
        <xsl:text>MULTILINESTRING(</xsl:text>
        </xsl:otherwise>
      </xsl:choose>
      <xsl:apply-templates />
    </xsl:for-each>
    <xsl:text>)</xsl:text>
  </xsl:template>
```

<!-- CAUTION: Call to template for "gml:srsName" must be placed in one of the two possible places: either (a) or (b) -->

```
<xsl:template match="gml:MultiPolygon">
  <!-- (a) SRS is specified in the MultiPolygon element -->
  <xsl:call-template name="gml:srsName"/>
  <xsl:for-each select="gml:polygonMember//gml:Polygon">
    <xsl:choose>
      <xsl:when test="not(position()=1)">
        <xsl:text>,</xsl:text>
      </xsl:when>
      <xsl:otherwise>
        <!-- (b) SRS is specified for each member Polygon element -->
        <xsl:call-template name="gml:srsName"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:for-each>
  <xsl:text>)</xsl:text>
</xsl:template>
```

```

        <xsl:text>MULTIPOLYGON(</xsl:text>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:text>(</xsl:text>
            <xsl:apply-templates select="gml:exterior|gml:outerBoundaryIs"/>
            <xsl:apply-templates select="gml:interior|gml:innerBoundaryIs"/>
            <xsl:text>)</xsl:text>
        </xsl:for-each>
        <xsl:text>)</xsl:text>
    </xsl:template>

    <xsl:template match="gml:MultiSurface">
        <xsl:for-each select="gml:surfaceMember//gml:Surface|gml:surfaceMember//gml:Polygon">
            <xsl:choose>
                <xsl:when test="not(position()=1)">
                    <xsl:text>,</xsl:text>
                </xsl:when>
                <xsl:otherwise>
                    <!--Workaround in order to print the SRS before the WKT geometry-->
                    <xsl:call-template name="gml:srsName"/>
                    <xsl:text>MULTIPOLYGON(</xsl:text>
                        </xsl:otherwise>
                    </xsl:choose>
                    <xsl:text>(</xsl:text>
                        <xsl:apply-templates />
                        <xsl:text>)</xsl:text>
                    </xsl:for-each>
                    <xsl:text>)</xsl:text>
                </xsl:template>

                <xsl:template match="gml:MultiGeometry|gml:GeometryCollection">
                    <xsl:call-template name="gml:srsName"/>
                    <xsl:text>GEOMETRYCOLLECTION(</xsl:text>
                        <xsl:for-each select="gml:geometryMember">
                            <xsl:if test="not(position()=1)">
                                <xsl:text>,</xsl:text>
                            </xsl:if>
                            <xsl:apply-templates select="*[not(local-name()='gml:srsName')]" />
                        </xsl:for-each>
                        <xsl:text>)</xsl:text>
                    </xsl:template>

```

```

</xsl:template>

<xsl:template match="gml:pointMember">
    <xsl:apply-templates select="gml:Point"/>
</xsl:template>

<xsl:template name="gml:lineStringMember">
    <xsl:call-template name="gml:posList"/>
</xsl:template>

<xsl:template match="gml:LineStringSegment">
    <xsl:apply-templates select="gml:posList"/>
</xsl:template>

<xsl:template match="gml:polygonMember|gml:patches//gml:PolygonPatch">
    <!-- xsl:if test="not(position()=1)"><xsl:text>,</xsl:text></xsl:if -->
    <!-- xsl:text></xsl:text -->
    <xsl:apply-templates select="gml:exterior|gml:outerBoundaryIs"/>
    <xsl:apply-templates select="gml:interior|gml:innerBoundaryIs"/>
    <!-- xsl:text></xsl:text -->
</xsl:template>

<xsl:template match="gml:exterior|gml:outerBoundaryIs">
    <xsl:apply-templates select="gml:LinearRing"/>
</xsl:template>

<xsl:template match="gml:interior|gml:innerBoundaryIs">
    <xsl:text>,</xsl:text>
    <xsl:apply-templates select="gml:LinearRing"/>
</xsl:template>

<xsl:template match="gml:LinearRing">
    <xsl:call-template name="gml:posList"/>
</xsl:template>

<xsl:template match="gml:pos">
    <xsl:call-template name="gml:posList"/>
</xsl:template>

<xsl:template match="gml:coordinates">
    <xsl:call-template name="gml:posList"/>
</xsl:template>

```

```

<!-- Allowing two possible expressions for lists of GML coordinates; the latter allows
attributes in the list. -->
<xsl:template name="gml:posList">
    <xsl:call-template name="gml:processPosList"/>
</xsl:template>

<!-- Convert list of GML coordinates into WKT by exchanging commas and space characters -->
<xsl:template name="gml:processPosList">
    <xsl:text></xsl:text>
    <xsl:for-each select="tokenize(replace(replace(.,'\s+$',''),'^\s+', ''),
'[, \s\n\r]+' )">
        <xsl:variable name="t" select="." />
        <xsl:value-of select="concat($t, if (position() = last()) then ' '
else if ((position() mod 2) = 0) then ', ' else ' ')" />
    </xsl:for-each>
    <xsl:text></xsl:text>
</xsl:template>

<!-- Spatial reference system SRS must be declared according to the EPSG catalogue. -->
<xsl:template name="gml:srsName">
    <xsl:variable name="srsName" select="." />
    <xsl:if test="string-length(@srsName)>0">
        <xsl:value-of select="concat('&lt;', $epsg, tokenize(@srsName,
'[:#]')[last()], '&gt; ')" />
    </xsl:if>
</xsl:template>

</xsl:stylesheet>

```

**Figure 12: XSL stylesheet GML2WKT.xsl for transforming geometries from GML into WKT.**

#### 9.1.4.1 Execution of the XSLT Transformation

As for metadata, all stylesheets for INSPIRE data have been integrated into the TripleGeo suite [TripleGeo], and their source code is also available from [INSPIRE2RDF]. Users may edit them in order to define suitable values for the parameters according to the dataset(s) at hand, and then perform an XSLT transformation to obtain the resulting RDF files.

Execution is similar to that carried out for metadata (Section 9.1.2.1), although a different class is invoked. The general pattern of the command that performs data transformation through TripleGeo is:

```
java -cp lib/*;TripleGeo.jar eu.geoknow.athenarc.triplegeo.InspireToRdf input.gml output.rdf
```

Assuming that the aforementioned parametrization of Inspire\_Main.xsl is used for handling data about protected sites (**PS**), the following command may be invoked:

```
java -cp lib/*;TripleGeo.jar eu.geoknow.athenarc.triplegeo.InspireToRdf
GR_ProtectedSites_Natura2000.gml ps_natura2000_GR.rdf
```

This will transform the INSPIRE-aligned dataset `GR_ProtectedSites_Natura2000.gml` into its RDF/XML representation `ps_natura2000_GR.rdf`, which then may be readily loaded into a triple store.

Of course, XSL stylesheets can be used outside the TripleGeo suite, by making use of any XSLT parser. For example, the same dataset can be transformed into RDF/XML using Saxon [Saxon] as follows:

```
java -cp .;saxon9he.jar net.sf.saxon.Transform -s:GR_ProtectedSites_Natura2000.gml -xsl:
Inspire_main.xsl -o:ps_natura2000_GR.rdf
```

## 9.2 (Geo)SPARQL Endpoints for INSPIRE-Aligned Data and Metadata

### 9.2.1 Implementation of SPARQL Endpoints

As explained in Section 6.4, we have set up five (5) backends for querying INSPIRE data and metadata. In order to hide the differences between them, and expose a simple and uniform interface to the end-user, we have implemented a web application that provides the means to issue (Geo)SPARQL queries and receive responses in a variety of formats (RDF/XML, CSV, HTML, etc.). This web interface is currently available from:

<http://geodata.gov.gr/sparql/>

The application consists of the following basic parts:

- *A client-side JavaScript application.* This part is built on the Underscore/Backbone JavaScript framework. It provides a basic user-interface (web UI) to:
  - (1) select a backend (SPARQL engine);
  - (2) edit a query ;
  - (3) load typical example queries to start or experiment with (optional);
  - (4) select a result format (supported by the chosen backend);
  - (5) issue a query;
  - (6) preview the result (if any) into an integrated result panel;
  - (7) directly download the result.
- *A server-side PHP application.* This part is responsible to act as both an API proxy and an abstraction layer. It expects a request which consists of 3 fields:
  - (1) a query string: a SPARQL query;
  - (2) a graph-uri: a URI identifying a backend;
  - (3) a result-format: a preferred MIME type for the result.

When it receives a valid request, our proxy decides which is the target backend, builds the proper backend-specific request and issues the newly-created request to the target. Then, it starts a timeout timer and it blocks, waiting for a reply. On completion, it proxies back the result to the end-user. On a timeout or on a failed request, a descriptive error message is generated.

Next, we provide details about the set up of the backends, as well as indicative queries that can be used to explore all available data and metadata.

## 9.2.2 Using (Geo)SPARQL Endpoints for INSPIRE Metadata

We have set up two (2) SPARQL endpoints for metadata regarding seven datasets from geodata.gov.gr, in triple stores running on separate Virtual Machines (VM) with these configurations:

Triple store	OS version	CPU	RAM	Disk space
1) Parliament 2.7.4 <i>quickstart</i>	Linux CentOS 6.4 (64-bit)	Intel Core i7-3820 with 4 (virtual) CPU cores	2 GB	40 GB
2) Virtuoso Universal Server 7.1 (ColumnStore edition)	Linux Debian 6.x squeeze (64-bit)	Intel Core i7-3820 with 4 (virtual) CPU cores	2 GB	40 GB

A detailed examination of the capabilities of each store has been carried out in Deliverable D2.1.1 [GeoKnowD21]. In terms of their geospatial support, it is important to mention that:

- Parliament is GeoSPARQL-compliant. However, WKT literals for geometries must be prefixed with `sf:` and not with `geo:` as [OGC12] specifies. Otherwise, queries return no results. This inconsistency will be remedied in future releases of Parliament. Coordinates may be georeferenced at any valid EPSG or OGC system, and not only in lon/lat coordinates.
- Virtuoso is not yet GeoSPARQL compliant [Virtuoso71], so every `sf:wktLiteral` had to be turned into `virtrdf:Geometry` in order for queries to work properly. Coordinates must be always given in WGS84 reference system (i.e., lon/lat coordinates).

For convenience, we demonstrate the following steps for loading triples and submitting queries in the GeoSPARQL endpoint built on top of Parliament. Similar steps can be executed against the same graph in Virtuoso. All details can be found at [INSPIRE2RDF].

- To create a graph:  
`CREATE GRAPH <urn:x-geoknow-eu:sparql:parliament:metadata:inspire>`
- Bulk insertion of all RDF triples was carried out with custom Java scripts, exactly as explained in Deliverable D2.1.1 “Market and Research Overview” [GeoKnow21].

All queries about METADATA should target the following named graph URI in the GeoSPARQL endpoint: `urn:x-geoknow-eu:sparql:parliament:metadata:inspire`. Hence, in their WHERE clauses, they must specify that GRAPH `<urn:x-geoknow-eu:sparql:parliament:metadata:inspire>` should be used as a source.

Next, we present a set of indicative SELECT queries that are readily available for execution from the web interface. Of course, users may specify their own (Geo)SPARQL requests as well, including CONSTRUCT statements for returning results as triples. Any other SPARQL statements are not allowed.

- **M1:** Show all keywords available in the metadata:

```
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>
SELECT DISTINCT ?key
WHERE {
  ?dataURI dct:description ?details .
  ?dataURI dcat:landingPage ?homeURL .
  ?dataURI dcat:keyword ?key .
```



```
}
ORDER BY ?key
```

- **M2:** Report when the latest update of each dataset did actually occur (after 2011):

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?dataURI ?title (MAX(?date) AS ?lastModified)
WHERE {
  { ?dataURI dct:title ?title .
    ?dataURI dct:created ?date }
  UNION
  { ?dataURI dct:title ?title .
    ?dataURI dct:modified ?date }
  FILTER ( ?date >= "2011-01-01"^^xsd:date )
}
GROUP BY ?dataURI ?title
ORDER BY ASC(?lastModified)
```

- **M3:** Display providers (i.e. data publishers) for each dataset:

```
PREFIX rdfs: <http://www.w3.org/1999/01/rdf-schema#>
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX roles: <http://inspire.ec.europa.eu/codelist/ResponsiblePartyRole/>
SELECT ?dataURI ?title ?scheme ?provider ?license ?email
WHERE {
  ?dataURI dct:title ?title .
  ?dataURI prov:qualifiedAttribution ?attr .
  ?attr dct:type roles:resourceProvider .
  ?attr prov:agent ?agent .
  ?agent vcard:organization-name ?provider .
  ?agent vcard:hasEmail ?email .
  ?dataURI dcat:distribution ?distr .
  ?distr dct:rights ?r .
  ?r rdfs:label ?license .
  OPTIONAL { ?dataURI dcat:theme ?theme .
    ?theme skos:inScheme ?s .
    ?s rdfs:label ?scheme . }
}
```

- **M4:** Find any available datasets with metadata that contain a given keyword:

```
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>
SELECT ?dataURI ?title ?details ?key ?subject ?homeURL
WHERE {
  ?dataURI dct:title ?title .
  ?dataURI dct:subject ?subject .
  ?dataURI dct:description ?details .
  ?dataURI dcat:landingPage ?homeURL .
  ?dataURI dcat:keyword ?key .
  FILTER (REGEX(?subject, "^environment*", "i"))
}
```

- **M5:** Identify datasets with total spatial extent (MBB) overlapping the given area of interest (a rectangle):

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX locn: <http://www.w3.org/ns/locn#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX sf:<http://www.opengis.net/ont/sf#>
SELECT ?dataURI ?title ?details (geof:envelope(?fWKT) AS ?totalExtent)
WHERE {
  ?dataURI dct:title ?title .
  ?dataURI dct:description ?details .
  ?dataURI dct:spatial ?extent .
  ?extent locn:geometry ?fWKT .
  FILTER (geof:sfIntersects ("POLYGON((21.46 37.96, 22.73 37.96, 22.73 39.11, 21.46 39.11, 21.46 37.96))"^^sf:wktLiteral, geof:envelope(?fWKT) ))
}
```

- **M6:** Get details for all available datasets. This query binds together several pieces from previous ones:

```
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX roles: <http://inspire.ec.europa.eu/codelist/ResponsiblePartyRole/>
SELECT ?dataURI ?title ?provider ?details ?subject ?homeURL
WHERE {
  ?dataURI dct:title ?title .
  ?dataURI dct:subject ?subject .
  ?dataURI dct:description ?details .
  ?dataURI dcat:landingPage ?homeURL .
  ?dataURI prov:qualifiedAttribution ?attr .
  ?attr dct:type roles:resourceProvider .
  ?attr prov:agent ?agent .
  ?agent vcard:organization-name ?provider .
}
```

### 9.2.3 Using (Geo)SPARQL Endpoints for INSPIRE Data

We have set up another two SPARQL endpoints for seven datasets from [geodata.gov.gr](http://geodata.gov.gr), in triple stores running on separate Virtual Machines (VM) with the following configurations:

TripleStore	OS version	CPU	RAM	Disk space
1) Parliament 2.7.4 <i>quickstart</i>	Linux CentOS 6.4 (64-bit)	Intel Core i7-3820 with 4 (virtual) CPU cores	2 GB	40 GB
2) Virtuoso Universal Server 7.1 (ColumnStore edition)	Linux Debian 6.x squeeze (64-bit)	Intel Core i7-3820 with 4 (virtual) CPU cores	2 GB	40 GB

As in the case of metadata, Parliament is GeoSPARQL-compliant, whereas Virtuoso is currently not. For convenience, we demonstrate the following steps for loading triples and

submitting queries in the GeoSPARQL endpoint built on top of Parliament. Similar steps can be executed against the same graph in Virtuoso. All details can be found at [INSPIRE2RDF].

- To create a graph:  
CREATE GRAPH <urn:x-geoknow-eu:sparql:parliament:data:inspire>
- Bulk insertion of all RDF triples was carried out with custom Java scripts, exactly as explained in Deliverable D2.1.1 [GeoKnow21].

All queries about this DATA should target the following named graph URI in the GeoSPARQL endpoint: urn:x-geoknow-eu:sparql:parliament:data:inspire . Hence, in their WHERE clauses, they must specify that GRAPH <urn:x-geoknow-eu:sparql:parliament:data:inspire> should be used as a source.

Next, we present a set of indicative SELECT queries that are readily available for execution from the web interface. Of course, users may specify their own (Geo)SPARQL requests as well, including CONSTRUCT statements for returning results as triples. Any other SPARQL statements are not allowed.

Note that some queries should better specify a 'LIMIT k' clause, otherwise it may take a long time to provide a response (e.g., in a spatial join search). Most queries involve multiple data themes (as denoted with the initials, e.g., AU for Administrative Units, TN for Transport Networks, etc.), in order to demonstrate the wealth of information that may be extracted from this data.

- **D1:** Which administrative unit (AU) is at the specified geographic location (given with coordinates):

```
PREFIX gn: <http://inspire.jrc.ec.europa.eu/schemas/gn/3.0/>
PREFIX au: <http://inspire.jrc.ec.europa.eu/schemas/au/3.0/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX sf: <http://www.opengis.net/ont/sf#>
SELECT ?nCode ?aName
WHERE {
  ?f au:nationalCode ?nCode .
  ?f au:name ?a .
  ?a gn:GeographicalName ?gnName .
  ?gnName gn:spelling ?spn .
  ?spn gn:SpellingOfName ?spt .
  ?spt gn:text ?aName .
  ?f geo:hasGeometry ?fGeom .
  ?fGeom geo:asWKT ?fWKT .
  FILTER (geof:sfContains(?fWKT, "<http://www.opengis.net/def/crs/EPSC/0/4326> POINT (40.582051 22.952149)""^sf:wktLiteral))
}
```

- **D2:** Find protected sites (PS) within a distance of 20 km from the given location:

```
PREFIX gn: <http://inspire.jrc.ec.europa.eu/schemas/gn/3.0/>
PREFIX ps: <http://inspire.jrc.ec.europa.eu/schemas/ps/3.0/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/>
PREFIX sf: <http://www.opengis.net/ont/sf#>
SELECT ?fName ?dist
WHERE {
  ?f ps:siteName ?p .
  ?p gn:GeographicalName ?gnName .
  ?gnName gn:spelling ?spn .
  ?spn gn:SpellingOfName ?spt .
}
```

```
?spt gn:text ?fName .
?f geo:hasGeometry ?fGeom .
?fGeom geo:asWKT ?fWKT .
BIND (geof:distance(?fWKT, "<http://www.opengis.net/def/crs/EPSSG/0/4326> POINT (37.975598
23.735933)"^^sf:wktLiteral, uom:metre) AS ?dist) .
FILTER (?dist < 20000)
}
ORDER BY ?dist
```

- **D3:** Identify the administrative unit (AU) where each protected site (PS) belongs to:

```
PREFIX gn: <http://inspire.jrc.ec.europa.eu/schemas/gn/3.0/>
PREFIX au: <http://inspire.jrc.ec.europa.eu/schemas/au/3.0/>
PREFIX ps: <http://inspire.jrc.ec.europa.eu/schemas/ps/3.0/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
SELECT ?adminName ?siteName
WHERE {
  ?q au:name ?r .
  ?r gn:GeographicalName ?rnName .
  ?rnName gn:spelling ?rspn .
  ?rspn gn:SpellingOfName ?rspt .
  ?rspt gn:text ?adminName .
  ?q geo:hasGeometry ?qGeom .
  ?qGeom geo:asWKT ?qWKT .
  ?f ps:siteName ?p .
  ?p gn:GeographicalName ?gnName .
  ?gnName gn:spelling ?spn .
  ?spn gn:SpellingOfName ?spt .
  ?spt gn:text ?siteName .
  ?f geo:hasGeometry ?fGeom .
  ?fGeom geo:asWKT ?fWKT .
  FILTER (geof:sfWithin(?fWKT, ?qWKT))
}
LIMIT 5
```

- **D4:** Find settlements (GN) that are contained within the administrative unit (AU) with code 9126:

```
PREFIX gn: <http://inspire.jrc.ec.europa.eu/schemas/gn/3.0/>
PREFIX au: <http://inspire.jrc.ec.europa.eu/schemas/au/3.0/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
SELECT ?sName
WHERE {
  ?f au:nationalCode "9126" .
  ?f geo:hasGeometry ?fGeom .
  ?fGeom geo:asWKT ?fWKT .
  ?s gn:name ?p .
  ?s geo:hasGeometry ?sGeom .
  ?sGeom geo:asWKT ?sWKT .
  ?p gn:GeographicalName ?stName .
  ?stName gn:spelling ?stn .
  ?stn gn:SpellingOfName ?sptn .
  ?sptn gn:text ?sName .
  FILTER (geof:sfWithin(?sWKT, ?fWKT))
}
ORDER BY ?sName
```

- **D5:** Retrieve all cadastral parcels (CP) of area greater than 20,000 sq.m.:

```
PREFIX cp: <http://inspire.jrc.ec.europa.eu/schemas/cp/3.0/>
```

```
SELECT ?pCode ?area
WHERE {
    ?f cp:label ?pCode .
    ?f cp:areaValue ?area .
    FILTER ( ?area > 20000)
}
ORDER BY ?area
```

- **D6:** Retrieve road names (TN) in lexicographical order:

```
PREFIX tn: <http://inspire.jrc.ec.europa.eu/schemas/tn/3.0/>
PREFIX gn: <http://inspire.jrc.ec.europa.eu/schemas/gn/3.0/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
SELECT DISTINCT((?roName) AS ?roadName)
WHERE {
    ?r geo:hasGeometry ?rGeom .
    ?rGeom geo:asWKT ?rWKT .
    ?r tn:geographicalName ?p .
    ?p gn:GeographicalName ?rdName .
    ?rdName gn:spelling ?rpn .
    ?rpn gn:SpellingOfName ?rpt .
    ?rpt gn:text ?roName .
}
ORDER BY ?roadName
LIMIT 10
```

- **D7:** Identify waterstreams (HY) of length longer than 30km:

```
PREFIX gn: <http://inspire.jrc.ec.europa.eu/schemas/gn/3.0/>
PREFIX hy-n: <http://inspire.jrc.ec.europa.eu/schemas/hy-n/3.0/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?riverName ?len
WHERE {
    ?q hy-n:geographicalName ?r .
    ?r gn:GeographicalName ?rnName .
    ?rnName gn:spelling ?rspn .
    ?rspn gn:SpellingOfName ?rspt .
    ?rspt gn:text ?riverName .
    ?q hy-n:length ?len .
    FILTER ( xsd:decimal(?len) > 30000)
}
```

Note that results refer to parts of rivers, due to geometry splits at intersections with tributaries.

- **D8:** Find rivers (HY) that intersect with protected sites (PS):

```
PREFIX gn: <http://inspire.jrc.ec.europa.eu/schemas/gn/3.0/>
PREFIX hy-n: <http://inspire.jrc.ec.europa.eu/schemas/hy-n/3.0/>
PREFIX ps: <http://inspire.jrc.ec.europa.eu/schemas/ps/3.0/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
SELECT ?riverName ?siteName
WHERE {
    ?q hy-n:geographicalName ?r .
    ?r gn:GeographicalName ?rnName .
    ?rnName gn:spelling ?rspn .
    ?rspn gn:SpellingOfName ?rspt .
    ?rspt gn:text ?riverName .
    ?q geo:hasGeometry ?qGeom .
    ?qGeom geo:asWKT ?qWKT .
    ?f ps:siteName ?p .
    ?p gn:GeographicalName ?gnName .
```

```

    ?gnName gn:spelling ?spn .
    ?spn gn:SpellingOfName ?spt .
    ?spt gn:text ?siteName .
    ?f geo:hasGeometry ?fGeom .
    ?fGeom geo:asWKT ?fWKT .
    FILTER (geof:sfIntersects(?fWKT, ?qWKT))
  }
LIMIT 10

```

Note that this query may be also specified with the topological predicate `sfCrosses`, in case the user wishes to identify rivers that enter and leave from a given protected site.

- **D9:** Identify all addresses (AD) that refer to a particular road (with code 374):

```

PREFIX base: <http://inspire.jrc.ec.europa.eu/schemas/base/3.2/>
PREFIX ad:<http://inspire.jrc.ec.europa.eu/schemas/ad/3.0/>
PREFIX gn: <http://inspire.jrc.ec.europa.eu/schemas/gn/3.0/>
SELECT ?sName ?addNum ?pCode
WHERE {
    ?f ad:locator ?aLoc .
    ?aLoc ad:AddressLocator ?addDes .
    ?addDes ad:designator ?aDes .
    ?aDes ad:LocatorDesignator ?aLocDes .
    ?aLocDes ad:designator ?addNum .
    ?f ad:component ?cp .
    ?cp ad:postCode ?pCode .
    ?f ad:component ?r .
    ?r ad:inspireId ?rId .
    ?rId base:localId "374" .
    ?r ad:name ?thName .
    ?thName gn:GeographicalName ?rnName .
    ?rnName gn:spelling ?rspn .
    ?rspn gn:SpellingOfName ?rspt .
    ?rspt gn:text ?sName .
}

```

- **D10:** Report addresses (AD) within 2km from the given location (given with specific coordinates):

```

PREFIX ad:<http://inspire.jrc.ec.europa.eu/schemas/ad/3.0/>
PREFIX gn: <http://inspire.jrc.ec.europa.eu/schemas/gn/3.0/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX sf: <http://www.opengis.net/ont/sf#>
PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/>
SELECT ?tName ?addNum ?pCode ?dist
WHERE {
    ?f ad:locator ?aLoc .
    ?aLoc ad:AddressLocator ?addDes .
    ?addDes ad:designator ?aDes .
    ?aDes ad:LocatorDesignator ?aLocDes .
    ?aLocDes ad:designator ?addNum .
    ?f ad:component ?cp .
    ?cp ad:postCode ?pCode .
    ?f ad:component ?r .
    ?r ad:name ?thName .
    ?thName gn:GeographicalName ?rnName .
    ?rnName gn:spelling ?rspn .
    ?rspn gn:SpellingOfName ?rspt .
    ?rspt gn:text ?tName .
}

```



```
?f ad:position ?addr .
?addr ad:GeographicPosition ?aPos .
?aPos geo:hasGeometry ?fGeom .
?fGeom geo:asWKT ?fWKT .
BIND (geof:distance(?fWKT, "<http://www.opengis.net/def/crs/EPSSG/0/4326> POINT (40.582051
22.952149)""^^sf:wktLiteral, uom:metre) AS ?dist) .
FILTER (?dist < 2000)
}
LIMIT 10
```

## 9.2.4 Using a Virtual SPARQL Endpoint for Discovering INSPIRE CSWs

As explained in Section 4, discovering INSPIRE-compliant metadata from CSWs available by several European agencies and portals does not employ a semantic repository. No triple store is used to hold any RDF metadata received from such CSW services. Instead, a virtual SPARQL endpoint has been set up, where users can pose their queries according to the ISO-compliant metadata schema. Our **TripleGeo-CSW middleware** transforms this query into a proper GetRecords request against each of the CSWs listed in Section 4.2. Any qualifying metadata records are then collected and transformed into RDF/XML format.

We have prepared and tested several (Geo)SPARQL queries against these CSW services in order to verify the robustness of our middleware, and also validate its functionality. These SELECT queries explore a wide range of metadata features, e.g., keywords, subjects, titles, as well as the geographical area covered by the available datasets.

The following indicative (Geo)SPARQL queries are available for use from the web interface and their details are explained at [INSPIRE2RDF]. Users may submit them “as is”, modify them to reflect their specific search criteria, or even write their own SELECT queries in order to discover INSPIRE-compliant datasets offered by those CSWs.

- **C1:** Identify datasets modified after a specific date:

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT *
WHERE {
  ?s dct:modified ?date .
  FILTER ( ?date > "2010-01-01"^^xsd:date )
}
```

- **C2:** Find datasets on multiple subjects:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT *
WHERE {
  {?s dc:subject "oceans"}
  UNION
  {?s dc:subject "boundaries"}
}
```

- **C3:** Find datasets that include a specific term in their title:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT *
WHERE {
  ?s dc:title ?title .
  FILTER ( REGEX( ?title , "*population*" )
}
```

- **C4:** Identify datasets that specify the given title and subject:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT *
WHERE {
  ?b dc:subject "biota" .
  ?s dc:title ?title .
  FILTER ( REGEX(?title , "*ecology*"))
}
```

- **C5:** Identify datasets that include a specific term in their title and were modified before a given date:

```
PREFIX dct: <http://purl.org/dc/terms>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT *
WHERE {
  ?s dct:modified ?date .
  ?b dc:title ?title .
  FILTER ( (?date < "2014-01-01"^^xsd:date) && REGEX(str(?title) , "%water%"))
}
```

- **C6:** Search for data with an area of coverage that geographically contains the given rectangle (BBOX):

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/geosparql/function/>
SELECT *
WHERE {
  ?s geo:hasGeometry ?fWKT .
  FILTER (geof:sfContains(?fWKT, "BOX2D(-5.01 50.23, 1.69 56.12)"^^geo:wktLiteral))
}
```

- **C7:** Find datasets mentioning a given term in their title and also having an area of coverage that is within the given rectangle (BBOX):

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/geosparql/function/>
SELECT *
WHERE {
  ?b dc:subject "biota" .
  ?s dc:title ?title .
  ?s geo:hasGeometry ?fWKT .
  FILTER ( REGEX(?title , "*ecology*") &&
    geof:sfWithin(?fWKT, "BOX2D(-5.01 50.23, 1.69 56.12)"^^geo:wktLiteral))
}
```

## 9.3 Licensing

The TripleGeo tool, XSL stylesheets, and TripleGeo-CSW middleware are free software (including the Java and python source code). They can be redistributed and/or modified under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3.0 of the License, or (optionally) any later version. A copy of the GNU General Public License should have been received along with these tools.



This suite of tools is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose. Please consult the GNU General Public License for more details [GPL3].

Data and metadata for Greece from [geodata.gov.gr](http://geodata.gov.gr) are licensed under Creative Commons CC BY 3.0 [CC3]. These open datasets are available for download by any interested party wishing to validate our methodology and/or experiment with their transformation into INSPIRE-compliant RDF data.