Open Geospatial Consortium

Publication Date: 2014-07-14

Approval Date: 2014-06-14

Posted Date: 2014-05-20

Reference number of this document: OGC 14-001

Reference URL for this document: http://www.opengis.net/doc/ER/testbed10/provenance

Category: Public Engineering Report

Editors: Joan Masó, Guillem Closa Yolanda Gil and Benjamin Proß

OGC® Testbed 10 Provenance Engineering Report

Document type:	OGC® Engineering Report
Document subtype:	NA
Document stage:	Approved for public release
Document language:	English

Abstract

The provenance activities reported in this document were part of the OGC Testbed 10 Cross Community Interoperability (CCI) thread. This OGC® document gives guidelines for the capture and documentation of provenance information at dataset, feature and attribute level. It only considers vector features (mainly, points and lines) and does not elaborate on the coverage data model (so it does not talk about provenance of raster information). It proposes an approach to use the W3C PROV standard with geospatial information that can come from different sources and are integrated through different processing steps. It also reviews the applicability of ISO19115 and ISO19115-2 lineage.

Keywords

ogcdoc, ogc documents, testbed10, cci, wps, conflation, provenance, w3c prov, 19115

Preface

This Engineering Report is part of the outcomes of the OGC Testbed 10 (Testbed-10), an interoperability test bed that is part of the OGC Interoperability Program (IP). The IP is a global, hands-on collaborative agile prototyping program designed to rapidly develop, test and deliver proven candidate standards into the OGC Standards Program, where they are formalized for public release.

Testbed 10 was organized around the following threads:

- Cross-Community Interoperability (CCI): Increase Geospatial community interoperability by building on CCI OWS-9 work in semantic mediation, volunteer geographic information (VGI), provenance and data quality, and Global Gazetteer. Explore the potential of interoperability in the hydrology domain and utilizing ontologies to more easily share and visualize geospatial data.
- Open Mobility: Explore the geospatial standards requirements needed to support the growing emerging mobile environment where client applications are mobile, information services are mobile, and increasingly distributed across cloud infrastructures. The Open Mobility thread will address these requirements while leveraging on the work achieved in the OWS-9 Testbed in the areas of Geopackages and Geopackaging services and new OWS Context encodings.
- Aviation: Develop and demonstrate the use of the Aeronautical Information Exchange Model (AIXM) and the Flight Information Exchange Model (FIXM), building on the work accomplished in prior testbeds to advance the applications of OGC Web Services standards in next generation air traffic management systems to support European and US aviation modernization programs.

The provenance activities reported in this document were part of the CCI thread.

License Agreement

Contents

Page

Figures

Page

OGC® Testbed 10 Provenance Engineering Report

1 Introduction

1.1 Scope

This OGC® document gives guidelines for the capture and documentation of provenance information at dataset, feature and attribute level. It only considers vector features (mainly, points and lines) and does not elaborate on the coverage data model (so it does not talk about provenance of raster information). It proposes an approach to use the W3C PROV standard with geospatial information that can come from different sources and are integrated through different processing steps. It also reviews the applicability of ISO19115 and ISO19115-2 lineage.

The document records the experiences in addressing the provenance needs of the conflation interoperability experiments conducted in OWS-10. The conflation Web Processing Service is the main use case.

This OGC® document is applicable to WPS services that process data and need to record detailed provenance information and to clients that will present it.

1.2 Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
Joan Masó	joan.maso@uab.cat
Guillem Closa	g.closa@creaf.uab.cat
Yolanda Gil	gil@isi.edu
Benjamin Proß	b.pross@52north.org

1.3 Revision history

Date	Release	Editor	Primary clauses modified	Description
10/04/2014	0.0.1	Guillem Closa	All	Initial Document
17/04/2014	0.0.2	Joan Masó	All	Content enrichment
30/04/2014	0.0.3	Yolanda Gil	All	General organization and review
05/05/2014	0.0.4	Benjamin Proß	8.2.1	Description of WPS 52 North service

We would like to thank Daniel Garijo and Luc Moreau for comments and suggestions on the use of W3C PROV and all comments and contribution of the OWS10 group

1.4 Future work

Future versions of this document could explore the applicability of different queries into the provenance information and address query performance depending on the storage model. Future versions could also consider more optimizations to reduce the amount of information to be recorded.

See also more generic future work in Section 12.

1.5 Forward

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

2 References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

ISO 19115:2003 - Geographic information – Metadata

ISO 19115-1:2014 - Geographic information -- Metadata -- Part 1: Fundamentals

ISO 19115-2:2008 - Geographic information -- Metadata -- Part 2: Extensions for imagery and gridded data

ISO 19139:2007 Geographic information -- Metadata -- XML schema implementation

W3C PROV-Overview. W3C Working Group Note 30 April 2013

W3C PROV Primer: PROV Model Primer. Working Group Note, World Wide Web Consortium (W3C), 30 April 2013. Available from http://www.w3.org/TR/prov-primer/

PROV-DM: The PROV Data Model. W3C Recommendation 30 April 2013

PROV-O: The PROV Ontology. W3C Recommendation 30 April 2013

PROV-XML: The PROV XML Schema. W3C Working Group Note 30 April 2013

W3C PROV XG: Final Report of the W3C Provenance Incubator Group. World Wide Web Consortium (W3C), 30 November 2010. Available from http://www.w3.org/2005/Incubator/prov/XGR-prov-20101214/

3 Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r3] shall apply. In addition, the following terms and definitions apply.

3.1
application profile
a subtype of an OGC Implementation Standard (OGC 08-062r4). It serves more detailed information about a specific domain than the standard itself. An AP specifies required interfaces, bindings and encodings to publish and access data.

3.2
conflation
a process of unifying two or more separate datasets, which share certain characteristics, into one integrated all-encompassing result [OGC 12-159]

3.3
feature
abstraction of real world phenomena [ISO 19101]

NOTE: A feature may occur as a type or an instance. Feature type or feature instance should be used when only one is meant.

3.4
feature attribute
characteristic of a feature [ISO 19109]

NOTE: A feature attribute has a name, a data type, and a value domain associated to it. A feature attribute for a feature instance also has an attribute value taken from the value domain.

3.5
geographic information
information concerning phenomena implicitly or explicitly associated with a location relative to the Earth [OGC 01-101]

3.6
input
data provided to a **process** [OGC 05-007r7]

3.7
instance
object that realizes a class [OGC 01-101]

3.8
lineage
chain of legal ownership of content; history of ownership [OGC 06-004r4]

3.9
operation
specification of a transformation or query that an object may be called to execute [OGC 02-112]

NOTE: An operation has a name and a list of parameters.

3.10
output
result returned by a **process** [OGC 05-007r7]

3.11
process
model or calculation that is made available at a **service instance** [OGC 05-007r7]

3.12
provenance
information on the place and time of origin or derivation or a resource or a record or proof of authenticity or of past ownership [OGC 06-004r4].
information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness [W3C PROV overview].

NOTE Even if the definition of lineage and provenance is slightly different, they are used as synonymous in this document.

3.13
WPS profile
a standalone standard profile containing: the Universal Resource Name that uniquely identifies the profiled process, a reference response to a DescribeProcess request for that process (reference process schema), a human-readable description of the process and its implementation (optional) and a Web Service Description Language (WSDL) document for that process (optional). [OGC 05-007r7]

4 Conventions

4.1 Abbreviated terms

Some more frequently used abbreviated terms:

PROV	World Wide Web Consortium Provenance
WFS	Web Feature Service
WPS	Web Processing Service
W3C PROV	World Wide Web Consortium Provenance

5 Overview

5.1 Feature overview

A feature is an abstraction of a real world phenomenon; it is a geographic feature if it is associated with a location relative to the Earth. Vector data consists of geometric and topological primitives used, separately or in combination, to construct objects that express the spatial characteristics of geographic features.

5.2 Geospatial process

Geospatial processes are pre-programmed calculations and/or computation models that operate on spatially referenced data. The data required by the service can be delivered across a network, or available at server level. The calculation can be as simple as subtracting one set of spatially referenced numbers from another, or as complicated as a global climate change model. In our case, the calculation is called conflation and is done on geometric features. The Web Processing Service standard (WPS), enabling geospatial processing on the Internet. WPS standardizes the way that these processes are called, in order to reduce amount of programming required, and to facilitate the implementation and adoption of new services.

The WPS interface specifies three operations that can be requested by a client and performed by a WPS server, all mandatory implementation by all servers. Those operations are GetCapabilities, DescribeProcess and Execute. Execute, is the operation that allows a client to run a specified process implemented by the WPS, using provided input parameter values and returning the outputs produced.

5.3 Provenance overview

Provenance provides important information about the origins of data that is crucial to assessing its quality, including: 1) the data sources used to generate the data, 2) the processes used and their characteristics, and 3) the actors and other entities involved in those processes.

The OGC OWS-9 Cross Community Interoperability “Conflation with Provenance” OGC 12-159 (https://portal.opengeospatial.org/files/?artifact_id=51818), describes earlier efforts done in recording provenance for conflating datasets with Web Processing Services. Section 5.6 and Annex B of that document propose to use ISO19115 Metadata standard to encode provenance of a conflation process. The recently published W3C PROV standard provides an alternative for encoding provenance. However, this standard

was designed for generic purposes and some work was needed in OWS-10 to assess its applicability to geospatial data, reported here.

5.3.1 Provenance in geospatial processing using data sources

A WPS is a service that exposes some individual calculations or processes. It has the potential of creating or modifying information by processing it. To do this, an individual process in invoked by means of an Execute operation call. Each time that an Execute operation is invoked, the services can potentially output new features. This document is focusing on capturing the provenance of the newly produced features by recording the source features but also the executions and parameters involved in each execution.

5.3.2 Provenance conflation use case

The OGC abstract specification topic 12 provides a non exclusive list of geographic processing services types. One of them is the “Feature matching service: A service that determines which features and portions of features are representing the same real world phenomena from multiple data sources”. Examples of this are edge matching and limited conflation.

In a conflation, data from two different sources is aligned and integrated. In our scenario, we assume that one dataset is not complete or up to date and thus there is a need to update it with features and attributes coming from another dataset. In some cases, a feature in the second dataset was completely missing in the first dataset and the feature and all its attributes are included together. In other cases, a feature in the second dataset is identified as a feature on the first dataset but some attributes of the second dataset are considered better and are imported while others remain intact. Both the geometrical features and non-geometrical information can be integrated during conflation. It is important to note that capturing the differences in the way that some features are conflated is the focus of this provenance report.

In addition, conflation operations are commonly used in this way and it is important to define a WPS profile for conflation that can be used by clients and services in the OWS10 interoperability experiment.

5.3.3 Conflation interoperability experiment examples in OWS10

These are some of the conflation experiments conducted during OWS10. In all cases, source datasets are served with WFS. The conflation WPS is able to access this data sources directly and returns both the conflated features and the provenance information (see Figure 1).

- Conflation of the roads in the USGS TNM dataset and the NGA Topographic Data Store (TDS) dataset.

- Conflation of coast lines coming from DNC Dataset and the NGA Topographic Data Store (TDS).

- Conflation of the forestations in the USGS TNM dataset and the NGA Topographic Data Store (TDS) dataset.

- Conflation of the roads in the USGS TNM dataset and the Open Street Map (OSM) dataset.

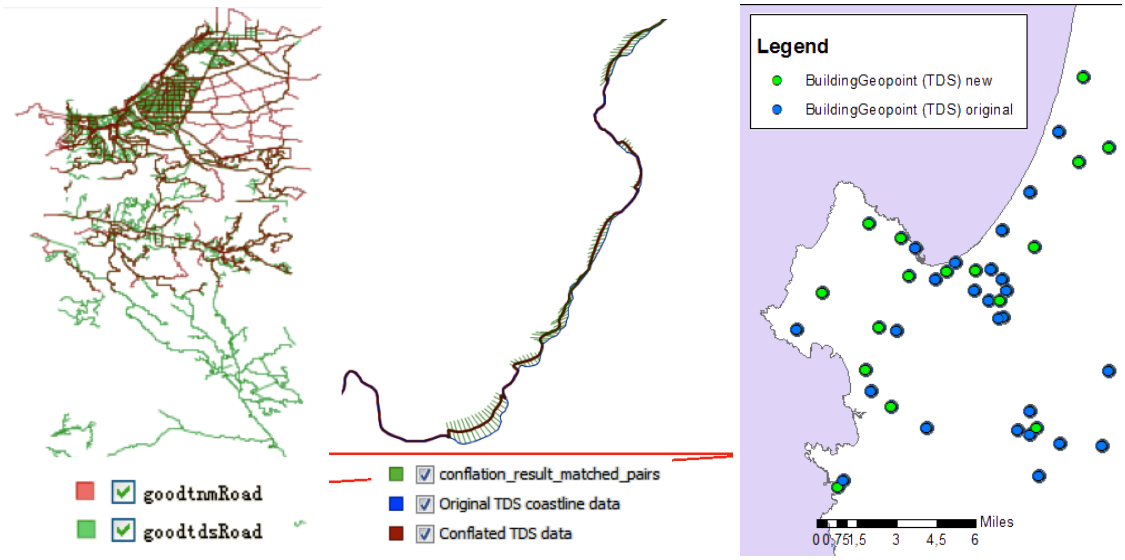

Figure 1— Different examples of conflation experiments.

5.4 Different elements captured in provenance

In provenance we want to capture the following information:

- *Sources:* datasets, features, geometric attributes and non-geometric attributes that were used to derive the resulting elements. These elements can be referenced using a descriptive citation, an element id (e.g. a gml:id), a metadata id, an element uri (or a url) or a metadata uri (or a url).

- *Process executions (process steps):* These are operations applied to the dataset (e.g. a WPS execution of an operation). They can be referenced by providing the name of the operation, an uri of the operation or a full description of the operation (e.g. a WPS execute document).

- *Process:* An engine that is able to execute a process step.

- *Algorithm:* The abstract logic that describes how a process engine can be implemented.

- *Responsible parties:* People and institutions in charge of the sources, the algorithms, and the execution of the conflation operations.

- *Dates:* The time when the source element was updated, when the process algorithm was implemented, or when the actual execution took place.

NOTE: In this section we are using the ISO 19115 naming conventions because we assume that the reader will be more familiarized with them. Next section will describe the equivalence between these names and the W3C PROV vocabulary.

5.5 Levels of granularity to capture provenance

We consider capturing provenance at three different levels:

- *Dataset-level provenance*: Includes the process steps involved in creating an entire dataset and the information of the complete process (e.g a WPS conflation process).
- *Feature-level provenance*: Some provenance is not common for all the features in the entire dataset, and in this case, provenance needs to be attached to particular features. This way a feature can carry its own provenance information. This is particularly useful when a small dataset is conflated with a very large dataset.
- *Attribute-level provenance*: It is common for the geometry and the non-geometrical attributes to be processed with different techniques and to be derived from different origins. In many cases, non-geometrical attributes are collected by different people and later attached to features. This opens the door to provenance information that is not common to all the attributes of that feature, and there is specific provenance for one or more of its attributes. In this case, the attribute can include additional provenance information. This is particularly useful when a single attribute is incorporated to a preexisting feature in a conflation process.

Another level of aggregation could be the *dataset series* but this level is considered out of the scope of this work.

There are different approaches to recording provenance, each with different merits and tradeoffs. Recording feature-level and attribute-level provenance has to be done with care, as it may require a large amount of storage space. One way to save storage space is recording only feature-level provenance when it is different from the provenance of the overall dataset, and attribute-level provenance when it differs from the provenance of the feature. Even if this storage method has its advantages, this introduces more steps in recovering the provenance of a single feature from the server and it can impact the service performance, when resolving complex queries.

5.6 Current standards to encode provenance

Traditionally, the geospatial world has been using first the FGDC metadata standards and later the ISO19115 (with some extensions) to encode provenance information at a dataset level. In OWS-9 some experiments were done to demonstrate the feasibility of using that standard also at the feature level as documented in [OGC 12-159]. At the dataset level, the ISO XML encoding for ISO19115 was used but at the feature level provenance and

quality was implemented by mapping ISO19115 concepts into feature attributes to avoid having to associate complex and large properties to each feature result of the conflation.

In OWS-10 the aim was to use the linkage capacity in ISO 19139 to demonstrate that a more compact notation can be used to embed small parts of metadata in geospatial information encoded in GML and to compare them with the two encodings provided by W3C PROV standards (XML and RDF) and implement the W3C PROV in RDF for the first time in an OGC experiment. Finally, we wanted to document the lessons learned when tracking provenance of complex conflation processes.

This section describes the two alternatives to encode provenance considered in OWS-10. The following sections will explain the considerations involved in using both. This document gives priority to the final implemented solution, W3C PROV, but it will also discuss the other options.

5.6.1 ISO 19115 lineage

ISO 19115 is a standard elaborated by the ISO TC 211 to describe metadata. It is also included in the OGC Abstract model Topic 11. This standard defines data quality as a composition of two parts: Lineage and Data Quality elements (see Figure 2).

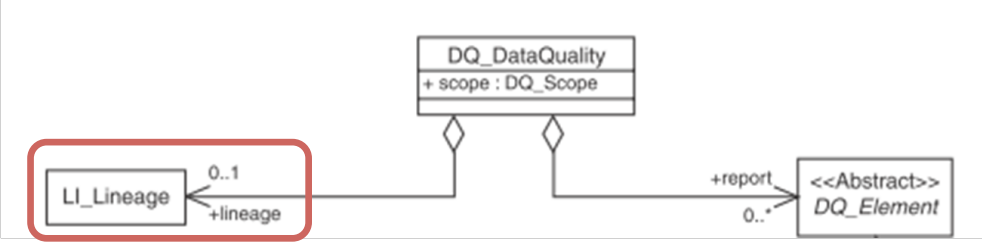

Figure 2 — Two components of the Data Quality in ISO 19115:2003

Besides a textual statement, lineage is formed by an array of sources, and an array of process steps. They can be provided as two independent lists but the way both are related, allows specifying a process step that relates to sources that can be related to older processes that have previous sources and so on in a tree structure. Even this possibility seems interesting, the common metadata tools does not implement this relations between sources and processes.

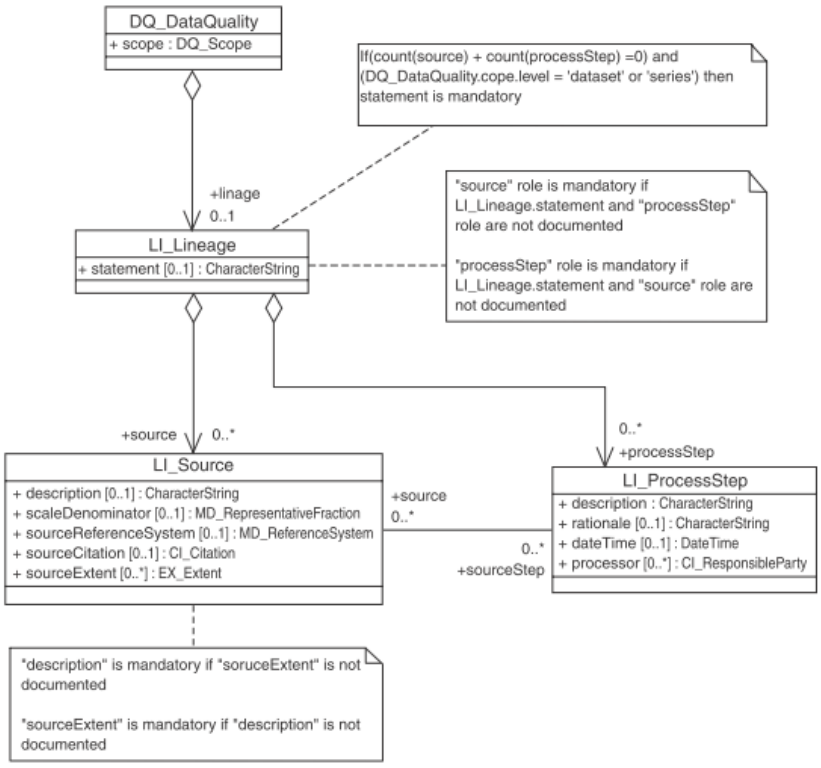

Figure 3 — Lineage elements in ISO 19115:2003.

ISO 19115-2 extends this model to include more sub elements that were suggested for the Earth Observation community but are of general interest. In particular, it includes the capability to reference a process and the corresponding algorithm or even a report describing the result.

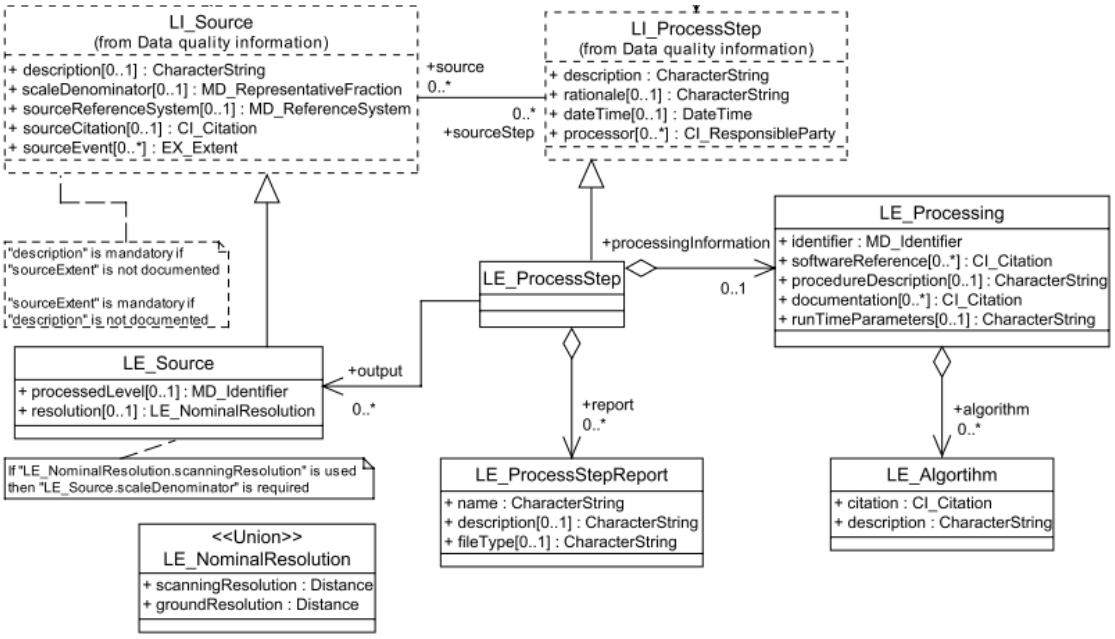

Figure 4 — Lineage elements in ISO 19115-2:2008.

ISO19115-1:2014 introduces minor modifications to ISO19115:2003.

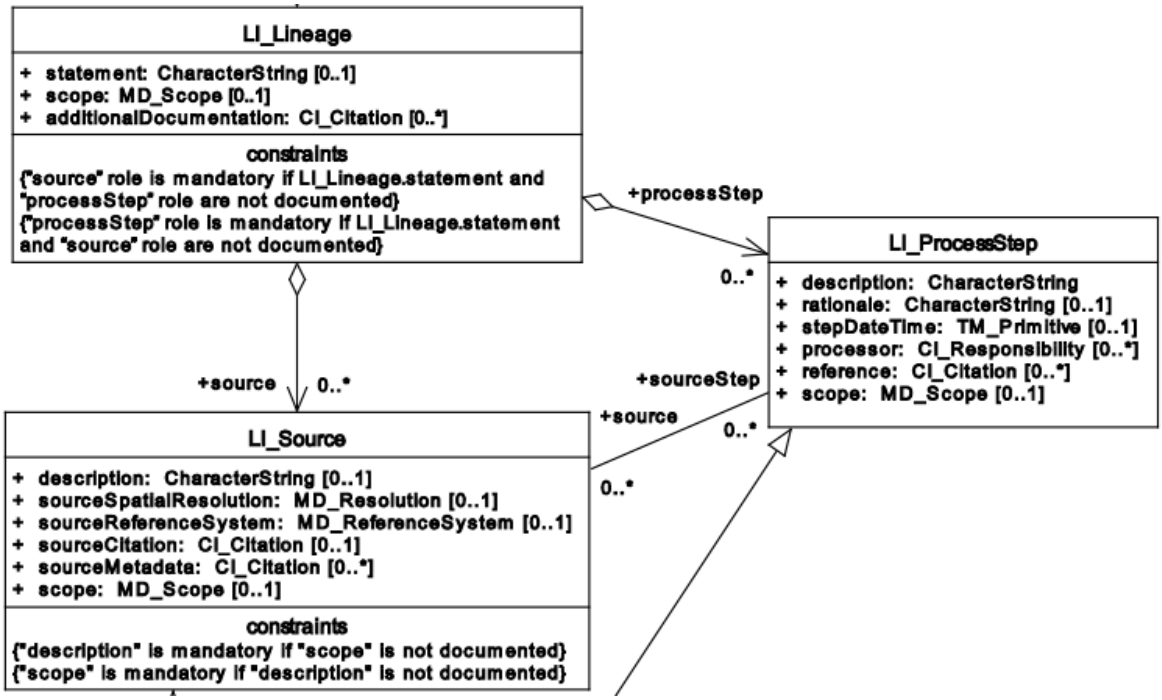

Figure 5 — Lineage elements in ISO 19115-1:2014.

5.6.1.1 XML encoding

ISO 19139 describes the only official encoding for ISO19115. The schemas are available in the OGC repository at http://schemas.opengis.net/iso/19139 but they do not include the ISO 19115-2 extensions. There is no official encoding for ISO19115-1:2014 yet. This aspect is elaborated in subsections 10.1.1 and 10.2.

5.6.2 W3C PROV

The W3C PROV standard provides a model to record provenance information in a generic way. The W3C initiated an incubator process in 2009, which collected many use cases from the community, articulated technical and usage requirements based on those use cases, analyzed the state of the art in provenance research and implementations, reviewed existing provenance vocabularies, and recommended a core set of terms to represent provenance as a starting point for a provenance standard for the Web [W3C PROV XG]. In 2011, a W3C Working Group was established to produce this standard, which led to the release of PROV in April 2013. The documents pertinent to the W3C PROV standard are summarized in Section 2 of this report. An overview of the standard is provided in [W3C PROV Overview], which includes a model (PROV-DM), an ontology (PROV-O), constraints in the data model, an XML serialization (PROV-XML), a notation (PROV-N), and semantics. Additional documents include a listing of PROV implementations. A brief overview of PROV is provided in the [W3C PROV PRIMER].

The PROV provenance model supports the representation of the entities, activities, and agents involved in producing a piece of data or thing, and the relations between those elements.

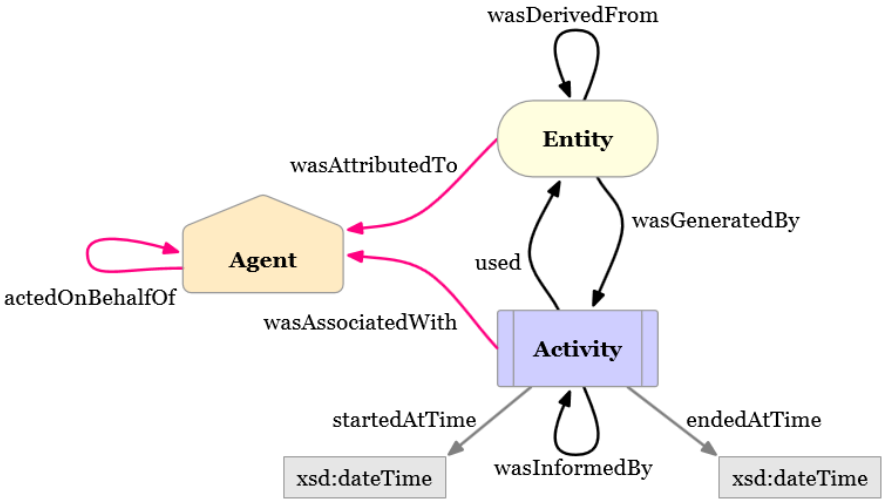

Figure 6 — Some elements and relations in W3C PROV

Figure 6 gives an overview of the W3C PROV provenance model. The model includes:

- *Entities:* In PROV, things we want to describe the provenance of are called entities.
- *Activities:* An activity is something that occurs over a period of time and acts upon or with entities; it may include processing, transforming, modifying, relocating, using, or generating entities.
- *Agents:* They are something or someone that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent's activity.
- *Plans:* A plan is special kind of entity that represents a set of actions or steps intended by one or more agents to achieve some goals. The execution of a plan results in activities.
- *Bundles:* A bundle is a named set of provenance descriptions, and it is itself an entity, allowing for provenance of provenance to be expressed.
- *Collections:* A collection is an entity composed by constituents that are themselves entities.

Activities and entities are associated with each other in two different ways: activities *use* entities and activities *generate* entities. Entities can be *attributed* to an agent, and activities can be *associated* to agents. The constituents of a collection are *members* of it.

Activities, entities, agents, etc. have some predefined attributes (such as type, label, location). Activities have also predefined relations for start and end times. Relations can also have attributes. One important attribute is the *role* that expresses the function of an entity or agent with respect to an activity, in the context of a usage, generation, invalidation, association, start, and end. Additional relations between elements can be found in the W3C PROV Data Model.

Applying W3C PROV mainly requires identifying entities, activities, agents and their relations. The process of identifying and encoding these elements for the geospatial domain is detailed in Section 6.

5.6.2.1 XML encoding

PROV-XML provides an XSD schema modularized so that xml-elements for different aspects of PROV are defined in separate extension schemas. The default schema, http://www.w3.org/ns/prov.xsd, imports http://www.w3.org/ns/prov-core.xsd and all extension schemas.

5.6.2.2 RDF encoding

The PROV Ontology (PROV-O) expresses the PROV Data Model [PROV-DM] using the W3C OWL2 Web Ontology Language (OWL2). It provides a set of classes, properties, and constraints that can be used to represent and interchange provenance information. Using this ontology, provenance can be encoded in RDF.

Since RDF allows for dynamic definitions of objects and their properties, we favored this encoding in the work reported here. More details are provided in Section 11.

6 Applying W3C PROV to geospatial information

The OWS-9 Cross Community Interoperability "Conflation with Provenance" OGC 12-159 (https://portal.opengeospatial.org/files/?artifact_id=51818), describes previous efforts done for using the ISO19115 Metadata standard to encode provenance on a conflation processes. The work on provenance in OWS-10 started out by reviewing that work and extending it to include attribute level provenance and a more compact notation. Then W3C PROV XML was considered. Finally, W3C PROV RDF was chosen for representing provenance of conflation process. More details on these decisions are provided in Section 11.

To give the reader the most relevant information first we decided to present the work in the reverse order: the following section describes W3C PROV in RDF and then in XML. Later, Section 10 describes the efforts done for the ISO model and Section 11 briefly compares both approaches.

6.1 Mapping Geospatial concepts and W3C PROV

W3C PROV is a general model, and we extent it to map it to geospatial information.

One of the first things that we have to do to use W3C PROV in the geospatial domain and in the OGC in particular is to map OGC concepts to PROV.

- A feature instance is an *entity.*
- A feature type is a *subclass of entity.*
- Datasets (some of them called *sources* in ISO lineage) are *collections of entities*.
- A WPS execution (process step) is an *activity.*
- A WPS execution parameter is an *entity.*
- An operation (WPS operation) is a *plan.*
- A processing service is a *collection of plans.*
- The definition of a parameter in a WPS execution is expressed as a *role* of an entity in an activity.
- The WPS standard is a *subclass of a plan collection.*
- The responsible party involved in the creation of plans and entities is an *agent.*
- The role or contribution of a responsible party to the conflation process is expressed as the *role* of an agent in an activity or in a plan.

Geospatial information is structured in different levels of granularity as explained in Section 5.5. When mapping the W3C PROV concepts to the geospatial domain we have adopted the feature level as the "basic" level, i.e. *features* are *entities* in PROV. A *dataset* (considered as a collection of features), is mapped to a *collection of entities* in PROV. Subsection 7.5.3.2 explains how the attribute level was addressed.

6.1.1 Encoding OGC concepts in W3C PROV

Now that we have mapped OGC terms to the W3C PROV standard (see previous Section), we can encode provenance records. We show here some examples, using a readable notation for the Resource Description Framework (RDF) called Turtle.

- Datasets and dataset sources are *collections of entities*.

```
ows:FeatureCollection rdfs:subClassOf prov:Collection .
```

- An abstract feature is a *subclass of an entity.*

```
ows:Feature rdfs:subClassOf prov:Entity .
```

- WPS execution (process steps) is an *activity.*

```
ows:WPSExecution rdfs:subclassOf prov:Activity .
```

- WPS execution parameter is an *entity* (in this way, we can associate roles to them and eventually we could even describe their provenance).

```
ows:WPSParameter rdfs:subclassOf prov:Entity .
```

- The generic input and output parameters of a WPS operation are *subclasses of role.*

```
ows:ProcessInput rdfs:subclassOf prov:Role .
ows:ProcessOutput rdfs:subclassOf prov:Role .
```

- The type of a responsible party is expressed as a *role* of an agent in an activity or in a plan

```
ows:Client rdfs:subclassOf prov:Role .
ows:Developer rdfs:subclassOf prov:Role .
```

- The WPS standard is a *subclass of plan collection* (that is also a *plan*)

```
ows:WPSService rdfs:subclassOf prov:Plan .
```

- A WPS operation is a *plan*

```
ows:WPSProcess rdfs:subclassOf prov:Plan .
```

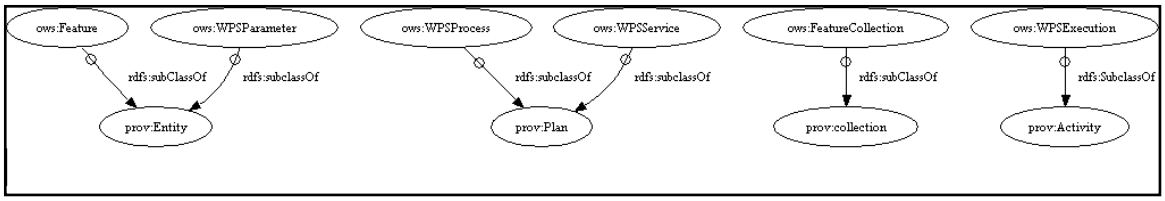

Figure 7 —OGC concepts in W3C PROV

All these triples are stored together in the file 0_Prov_OGC_RDF.n3 that can be found in Annex B.

In W3C PROV, *entities* can have additional attributes, and even a *location*. In W3C, entities are the *things* that have provenance information associated with them. This work addresses also attribute level provenance. To do so, feature properties need to be also considered *entities.* Even if this could not be strictly necessary, we also want to record the relation between *properties* and the *features* they are characterizing. We also wanted to emphasize that a feature has geometric and non-geometric properties because in many cases geometric properties are collected with different techniques than non-geometric so the provenance information will be different.

- A geometric property is a subclass of *entity*

```
ows:Geometry rdfs:subClassOf prov:Entity .
```

- Points, Lines and Polygons are subclasses of Geometry

```
ows:Point rdfs:subClassOf ows:Geometry .
ows:Line rdfs:subClassOf ows:Geometry .
ows:Polygon rdfs:subClassOf ows:Geometry .
```

- A non-geometric property is a subclass of *entity* (for notation simplification purposes, we assume that a *Property* is non geometrical property):

```
ows:Property rdfs:subClassOf prov:Entity .
```

Properties (geometric and non-geometric) need to be related to features. W3C PROV does not have the needed kind of relation. For that reason, we define this relation from scratch.

- A *feature* can have *geometric* properties by declaring that *hasGeometry*:

```
ows:hadGeometry a owl:ObjectProperty ;
       rdfs:comment "A geometric property of a resource" ;
       rdfs:domain ows:Feature;
       rdfs:isDefinedBy <http://www.opengis.net/ogc/ows/ows-core-
ontology/> ;
       rdfs:label "hasGeometry" ;
       rdfs:range ows:Geometry .
```


- A *feature* can have non-*geometric* properties by declaring that *hasProperty*:

```
ows:hadProperty a owl:ObjectProperty ;
       rdfs:comment "A Non geometric property of a resource" ;
       rdfs:domain ows:Feature;
       rdfs:isDefinedBy <http://www.opengis.net/ogc/ows/ows-core-
ontology/> ;
       rdfs:label "hasProperty" ;
       rdfs:range ows:Property .
```

All this triples are stored together in the file 0_Prov_OGC_RDF.n3 that can be found in Annex B.

6.2 Feature identifiers and namespaces

Every *entity* needs a name in W3C PROV. In particular, *features* need names. The simplest method to obtain an identifier is using a name inspired in the feature id (e.g. gml:id) escaping all none allowed characters in the selected notation (e.g. “.”s are converted to “_”s). These names are unique in the dataset and so they need to be in a namespace that represents the dataset.

For instance a feature with an id “road.66” in the “usgs” dataset will be defined as

```
@prefix usgs: <http://www.usgs.gov/usgs_dataset1/> .
usgs:USGS_Map1 a ows:FeatureCollection .
usgs:USGS_Feature rdfs:subClassOf ows:Feature .
usgs:road_66 a usgs:USGS_Feature .
```

Nevertheless, the name restrictions in the RDF turtle notation does not allow us to use the exact same name than in other encodings such as gml:id’s, so we need to set some equivalences. One possibility is to use a full URL to get the feature from a WFS request or a GML document.

```
USGS_Map1 owl:sameAs
<http://portal.cubewerx.com/cubewerx/projects/ows9/cubeserv.cgi?service
=WFS&amp;datastore=OWS9&amp;request=GetFeature&amp;typename=usgs:fireSt
ationEmsStation> .
usgs:road_66 owl:sameAs
<http://portal.cubewerx.com/cubewerx/projects/ows9/cubeserv.cgi?service
=WFS&amp;datastore=OWS9&amp;request=GetFeature&amp;typename=usgs:fireSt
ationEmsStation&amp;GetFeatureById=road.66> .
```

Another way to relate the rdf name to the feature id (e.g. gml:id) we could also do this.

```
@prefix gml: <http://www.opengis.net/gml/> .
usgs:road_66 gml:id "road.66"
```

7 Applying W3C PROV to the conflation process case

To describe the conflation process implemented in OWS-10, we use a modular approach and divide the problem in 6 layers: from the most general and abstract concepts, to the more specific and exact executions in OWS-10. Each layer is defined in different namespaces.

Figure 8 — W3C PROV Provenance Conflation Diagram showing the 6 layers (see the legend) in the modular approach and all the relations between objects (see it magnified in Annex A).

(magnified in Annex A) shows all abstract elements in the upper layers and some samples of individual features and attributes. The 6 layers are:

- The upper layer (layer 0) describes the general OGC concepts (explained in Section 6.2): includes the abstract WPS service, the abstract WPS operation and how they are related to the W3C PROV concepts. These concepts are also shown in the topmost layer and come directly from the W3C PROV owl definition (ogc namespace).

- The WPS Conflation Profile layer (layer 1) defines a generic conflation WPS, connecting it to a generic conflation algorithm and providing the list of input and output parameters and their roles. This is, in fact, what in OGC is called a WPS profile for conflation (described in OWS-7 Web Processing Service Profiling Engineering Report) (ows10 namespace).

- The layer 2 describes the concrete conflation process developed by 52North. Thus, in this example: the 52N Conflation WPS, which is a concrete WPS service and uses the specific algorithm of 52N(f2n namespace).

- This is the user layer (layer 3), where the person or the agent who executed the process (in this example, the NGA organization) is defined (nga namespace).

- The layer 4 defines the conflation session parameters (that is a sequence of conflation steps): the datasets (the sources maps and the conflated map) and the generic concept of a distance are described and related (each dataset has its own namespace, resulting in 3 namespaces in the example).

- In the conflation use case, the user is supervising conflation steps that involve a set of a few features and, for each step, we are documenting the features and attributes participating in it (layer 5) (each feature in each dataset has its own namespace, resulting in 3 namespaces in the example).

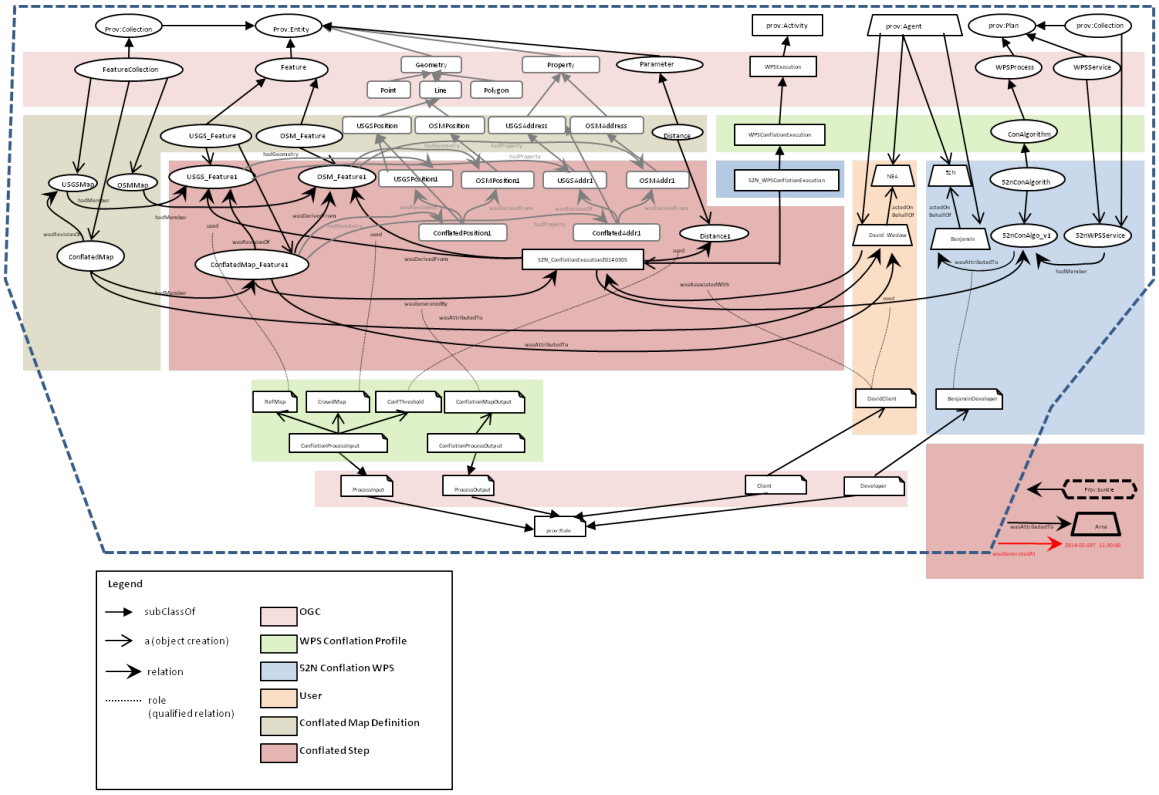

Figure 8 — W3C PROV Provenance Conflation Diagram showing the 6 layers (see the legend) in the modular approach and all the relations between objects (see it magnified in Annex A).

7.1 WPS Conflation Profile Encoding

The WPS Conflation Profile layer (layer 1) defines a generic conflation WPS, connecting it to a generic conflation algorithm and providing the list of input and output parameters and their roles. This is, in fact, what in OGC is called a WPS profile for conflation. The need for WPS profiles and the way they have to be implemented was described in the OWS7 Web Processing Service Profiling Engineering Report.

- A generic WPSConflationExecution of this profile is an *activity* and a subclass of WPSProcess.

```
ows10:WPSConflationExecution rdfs:subclassOf ows:WPSExecution .
```

- The generic conflation process implements a generic conflation algorithm, defined as a subclass of a *Plan*

```
ows10:ConflationAlgorithm rdfs:subclassOf ows:WPSProcess .
```

- The generic conflation process has conflation process inputs and conflation process outputs.

```
ows10:ConflationProcessInput rdfs:subclassOf ows:ProcessInput .
ows10:ConflationProcessOutput rdfs:subclassOf ows:ProcessInput .
```

- More concretely, a generic conflation process has 3 inputs: the reference map that we want to review, a crowdsourced map, and a threshold distance input (beyond this distance, the algorithm will not look for new matches). This generic process generates the *conflated map* as an only output.

```
ows10:referenceMapSource     a ows10:ConflationProcessInput .
ows10:crowdsourcedMapSource  a ows10:ConflationProcessInput .
ows10:conflationThreshold    a ows10:ConflationProcessInput .
ows10:conflatedMapOutput     a ows10:ConflationProcessOutput .
```

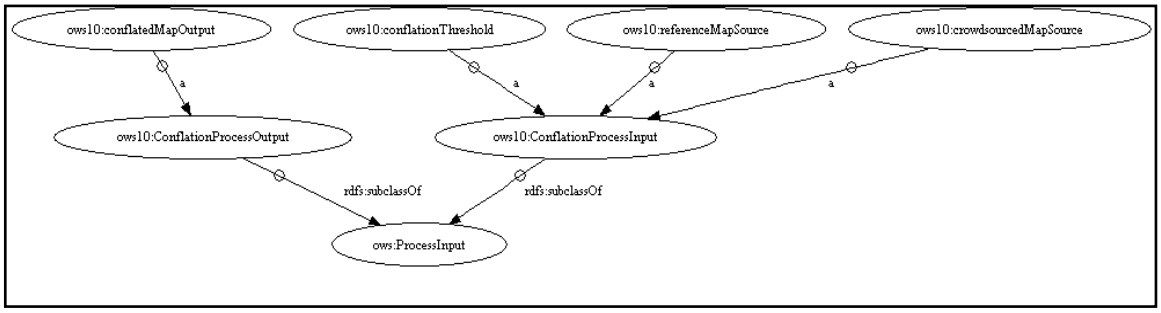

Figure 9 — Generic conflation inputs and outputs

7.2 52N Conflation WPS

This layer describes the specific aspects of the conflation process done by 52ºNorth: the concrete process, the algorithm used, the organizations and their roles involved in the process.

- The implementation of 52North follows the WPS conflation profile:

```
f2n:52N_WPSConflationExecution rdfs:subclassOf
ows10:WPSConflationExecution .
f2n:52N_ConflationAlgorithm rdfs:subclassOf ows10:ConflationAlgorithm .
```

- 52 North uses a particular conflation algorithm and this algorithm can have different implementations (versions) over time, each defined as an instance of the algorithm and therefore as an instance of PROV Plan:

```
f2n:52N_ConflationAlgorithm_v1 a f2n:52N_ConflationAlgorithm ;
```


- 52 North and its affiliates (Benjamin) are agents related with the 52 North conflation execution. Concretely, Benjamin has developed the algorithm and acts on behalf of 52 North:

```
f2n:52north a prov:Agent ;
f2n:benjamin a prov:Agent ;
f2n:benjamin prov:actedOnBehalfOf f2n:52north ;
f2n:52N_ConflationAlgorithm_v1 prov:wasAttributedTo f2n:benjamin .
```

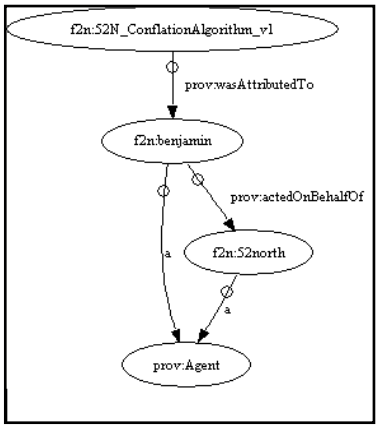

Figure 10 — Conflation algorithm attribution

7.3 WPS Execution User

This is the user layer (layer 3), where the person or the agent who executed the process (in this example in the NGA organization) is defined.

- The user (David, in this case) is an *agent* (it is affiliated to NGA) which is associated with the execution of a 52 North conflation processes to create the conflated map.

```
nga:NGA a prov:Agent   ;
nga:david a prov:Agent ;
nga:david prov:actedOnBehalfOf nga:NGA .
```

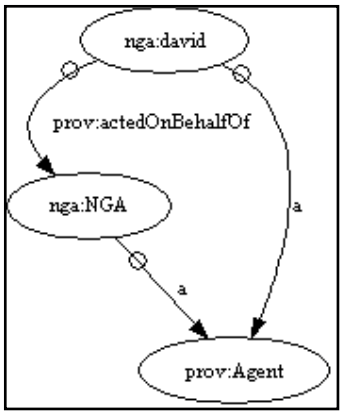

Figure 11 — NGA Agent levels

7.4 Conflated Map

This layer defines the precise inputs and outputs of the conflation session: 3 maps that are going to be conflated (dataset level) and a distance. As part of the definition of the dataset, generic features and attributes (feature and attribute types) are also defined.

- Three namespaces are defined; one for each dataset involved in the conflation process:

```
@prefix usgs: <http://www.opengis.net/ogc/ows10/usgs_dataset/> .
@prefix osm: <http://www.opengis.net/ogc/ows10/osm_dataset/> .
@prefix usgs_conf: <http://www.opengis.net/ogc/usgs_conf-dataset/> .
```

- There are three feature collections (*Datasets*) involved on the conflation process.

```
usgs:USGSMap a ows:FeatureCollection;
osm:OSMMap a ows:FeatureCollection;
usgs_conf:conflatedMap a ows:FeatureCollection .
```

- Additionally, a distance is also defined (values of it will be used as a threshold in the conflation step).

```
ows10:Distance rdfs:subClassOf ows:WPSParameter.
```

NOTE: To be rigorous, the distance concept has to be defined in a different namespace (e.g. in an O&M namespace) as it is not part of the WPS conflation profile definition.

- At feature level, two *feature types* are involved on the conflation process (subClassOf ows:feature)-please remember that in this use case the output features share the same feature type as the usgs map (reference map)-.

```
usgs:USGS_Feature rdfs:subClassOf ows:Feature .
osm:USGS_Feature rdfs:subClassOf ows:Feature .
```


- The attribute level is also represented by *entities*. Concretely, in this example, the process conflates positions (subClassOf ows:Line) and addresses (non-geographical property); both properties of the *Features* (subClassOf ows:Property).

```
usgs:USGS_Position rdfs:subClassOf ows:Line .
osm:OSM_Position rdfs:subClassOf ows:Line .
usgs:USGS_Address rdfs:subClassOf ows:Property .
osm:OSM_Address rdfs:subClassOf ows:Property .
```

- The result of the conflation process is a conflated map which is a revision of the USGS map.

```
usgs_conf:conflatedMap prov:wasRevisionOf usgs:USGSMap .
```

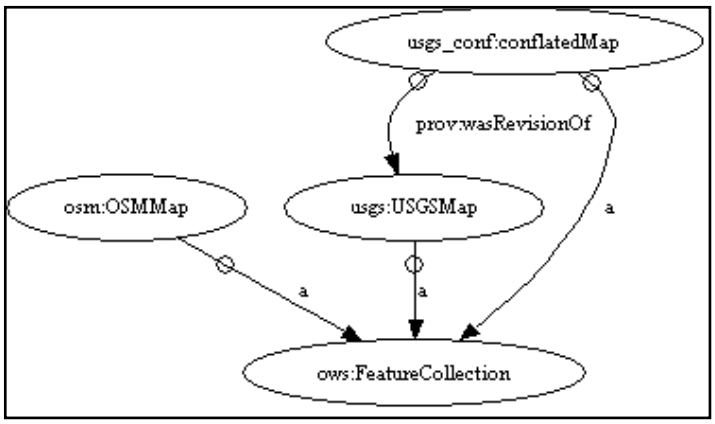

Figure 12 — Dataset conflated map inputs diagram.

7.5 Conflated Step

In the conflation use case, the user is supervising the conflation steps that involve a set of a few features. For each step, we are documenting the features and the attributes participating in it (layer 5).

7.5.1 Defining the process step

- Process step number 20140305 runs a conflation processes (in this case we use the date as an identifier).

```
f2n:52N_ConflationExecution20140305 a f2n:52N_WPSConflationExecution .
```

- The individual execution uses a plan that was executed by David as a client. The process execution started at 2014-03-03 08:10 and ended at 2014-03-03 10:20.

```
f2n:52N_ConflationExecution20140305  prov:used
f2n:52N_ConflationAlgorithm_v1 .

f2n:52N_ConflationExecution20140305 prov:qualifiedUsage [
                a prov:Usage ;
```

```
                prov:hadPlan  f2n:52N_ConflationAlgorithm_v1 ;
                prov:hadRole nga:Client
    ] .

f2n:52N_ConflationExecution20140305  prov:startedAtTime "2014-03-
03T08:10:00"^^xsd:dateTime ;
              prov:endedAtTime   "2014-03-05T10:20:00"^^xsd:dateTime .
```

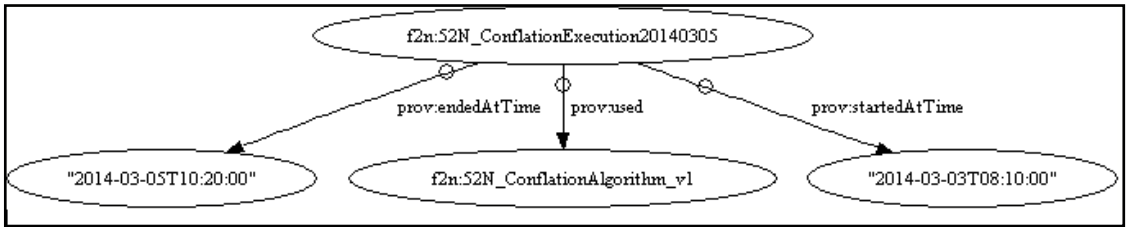

Figure 13 — Conflation execution is defined by the use of the 52N algoritm and the date.

7.5.2 Defining the features and describing their respective data model

- In this process step, one feature example and the properties of the first source dataset are involved, so they have to be defined. The same for the other features and properties.

```
usgs:USGS_Feature1 a usgs:USGS_Feature .
usgs:USGS_Position1 a usgs:USGS_Position .
usgs:USGS_Address1 a usgs:USGS_Address .

usgs:USGS_Feature1 ows:hadGeometry usgs:USGS_Position1;
                   ows:hadProperty usgs:USGS_Address1 .
```

- The same for the second source dataset.

```
osm:OSM_Feature1 a osm:OSM_Feature .
osm:OSM_Position1 a osm:OSM_Position .
osm:OSM_Address1 a osm:OSM_Address .

osm:OSM_Feature1 ows:hadGeometry osm:OSM_Position1;
                 ows:hadProperty osm:OSM_Address1 .
```

- The same for the target dataset (note that the target dataset follows the first source data model).

```
usgs_conf:conflated_Feature1 a usgs:USGS_Feature .
usgs_conf:conflated_Position1 a usgs:USGS_Position .
usgs_conf:conflated_Address1 a usgs:USGS_Address .

usgs_conf:conflated_Feature1 ows:hadGeometry osm:OSM_Position1;
                             ows:hadProperty osm:OSM_Address1 .
```


- The three example features are part of the dataset:

```
usgs:USGSMap prov:hadMember usgs:USGS_Feature1 .
osm:OSMMap prov:hadMember osm:OSM_Feature1 .
usgs_conf:conflatedMap prov:hadMember usgs_conf:conflated_Feature1 .
```

7.5.3 Recording the provenance information about the new conflated features

Now it is time to describe the feature and attribute level provenance for this process step. Here we provide some examples of different possibilities.

7.5.3.1 Feature level

- The conflation process step used some source features:

```
f2n:52N_ConflationExecution20140305   prov:used  usgs:USGS_Feature1;
                                      prov:used  usgs:USGS_Feature2;
                                      prov:used  osm:OSM_Feature1 .
```

1. The resulting feature was not affected by the conflation execution and it is copied as it is.

```
usgs_conf:conflated_Feature1 owl:sameAs usgs:USGS_Feature1 .
```

2. The feature is a revision of a first source feature

```
usgs_conf:conflated_Feature2 prov:wasRevisionOf usgs:USGS_Feature2 .
```

3. In fact, the feature is derived from a second source feature

```
usgs_conf:conflated_Feature2 prov:wasDerivedFrom osm:OSM_Feature1 .
```

- The conflation process step used a feature of the first source as a reference source information and a feature second source as a crowd source information:

```
f2n:52N_ConflationExecution20140305 prov:qualifiedUsage [
                  a prov:Usage ;
                  prov:entity  usgs:USGS_Feature1 ;
                  prov:hadRole ows10:ReferencedMapSource

f2n:52N_ConflationExecution20140305 prov:qualifiedUsage [
                a prov:Usage ;
                prov:entity  osm:OSM_Feature1 ;
                prov:hadRole ows10:crowdsourcedMapSource
     ] .
```

- The conflated feature was generated by the conflation process step in the time 2014-03-05 10:20.

```
usgs_conf:conflated_Feature1  prov:wasGeneratedBy
f2n:52N_ConflationExecution20140305 .
```

```
usgs_conf:conflated_Feature1 prov:generatedAtTime "2014-03-
05T10:20:00"^^xsd:dateTime .
```

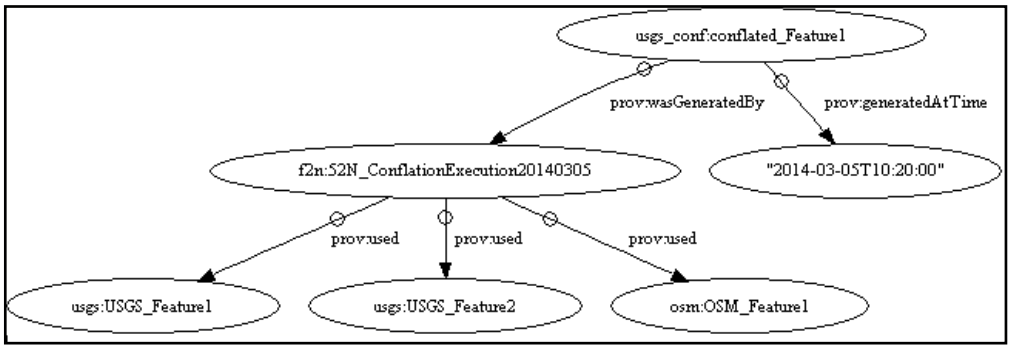

Figure 14—Conflated feature level

- The conflation process step created a feature as and output:

```
usgs_conf:conflated_Feature1 prov:qualifiedGeneration [
                    a prov:Generation ;
                    prov:activity  f2n:52N_ConflationExecution20140305
;
                    prov:hadRole   ows10:conflatedMapOutput
  ] .
```

7.5.3.2 Attribute level

The same possibilities can be applied also to attribute level:

1. The resulting geometry was not affected by the conflation execution and it is copied as it is.

```
usgs_conf:conflated_Position1 owl:sameAs usgs:USGS_Geometry1 .
```

2. The resulting address is a revision of a first source feature address.

```
usgs_conf:conflated_Address1 prov:wasRevisionOf usgs:USGS_Address1 .
```

3. In fact, the address is derived from a second source feature address.

```
usgs_conf:conflated_Address2 prov:wasDerivedFrom osm:OSM_Address1 .
```

7.5.3.3 Distance

- The threshold distance used in this process step is 10:

```
ows10:distance1 a ows10:Distance;
      prov:value 10 .
f2n:52N_ConflationExecution20140305 prov:used ows10:distance1 .
```



```
f2n:52N_ConflationExecution20140305 prov:qualifiedUsage [
                  a prov:Usage ;
                  prov:entity  ows10:distance1 ;
                  prov:hadRole ows10:conflationThreshold
     ] .
```

7.5.3.4 Bundle

Everything was documented by Arne at 2014-03-05 10:22. In W3C PROV specification, a bundle is defined as a provenance of the provenance information.

8 Using W3C PROV into a Conflation WPS

Conflation is understood as the process of unifying two or more separate datasets, which share certain characteristics, into one integrated all-encompassing result.

In the OWS-10 interoperability experiments, WPSs were used to generate conflated objects under user demand. The response of the WPSs was a combination of the conflated features instances and their property values encoded in GML and the provenance of the same features encoded in RDF triples. GML data was stored in WFS-T services and RDF information stored in its transactional triple store databases.

In the developed scenarios, the conflation process is not completely automated, i.e. the results of the conflation WPS is no automatic uploaded to a WFS-T. Instead, the user gets a chance to inspect the conflated result and decide whether to upload it or rerun the process with different parameters.

These are the components used in the architecture (see Figure 15):

- Client (PIXIS): Presents source 1, source 2, features result of the conflation and provenance of the conflated features to the user. It also allows the user to request a new conflation of some objects to the WPS and it is responsible to send the conflation result to the WFS-T and the SPARQL Server once visually inspected and validated by the user.

- WFS Source1: Contains a feature collection from source 1. These are the features to be conflated. E.g. USGS

- WFS Source2: Contains a feature collection from source 2. These are the features that will be identified as the same as some in Source 1 and they can be conflated. E.g. OSM

- Conflation WPS: Conflation process that generates a new set of conflated features. Responds back to the Client (PYXIS)

- SPARQL Server: Stores provenance information (entities, activities and agents) at dataset level (feature collection) and at the individual feature and attribute level

using the W3C PROV and RDF encoding. Individual feature provenance is linked to the features by id

- WFS-T: Stores the resulting conflated features and their attributes encoded in GML

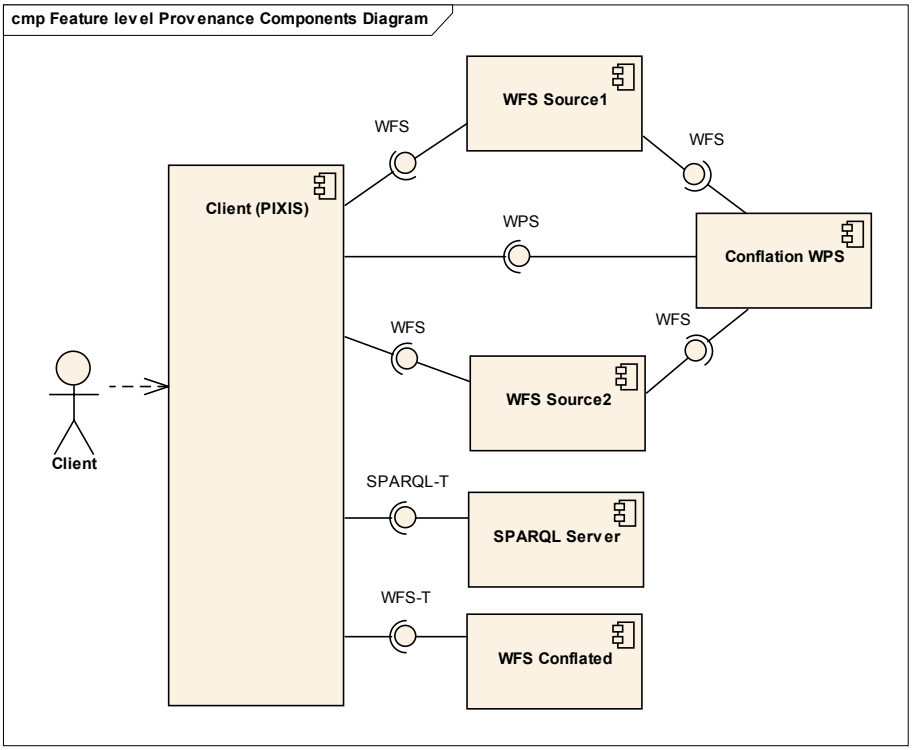

Figure 15 — Components involved in the conflation process

In the component diagram we see that the client is responsible for executing the WPS Conflation process but also to deal with the WPS result (that contains both the GML and the W3C PROV) and to send the final GML features to the WFS conflated layer and the W3C PROV in the SPARQL server.

8.1 User assisted conflation process

In the user assisted conflation process scenario, several steps are required to complete the task of generating a conflated result. Figure 1 UML diagram explains the sequence of steps that where implemented in this use case.

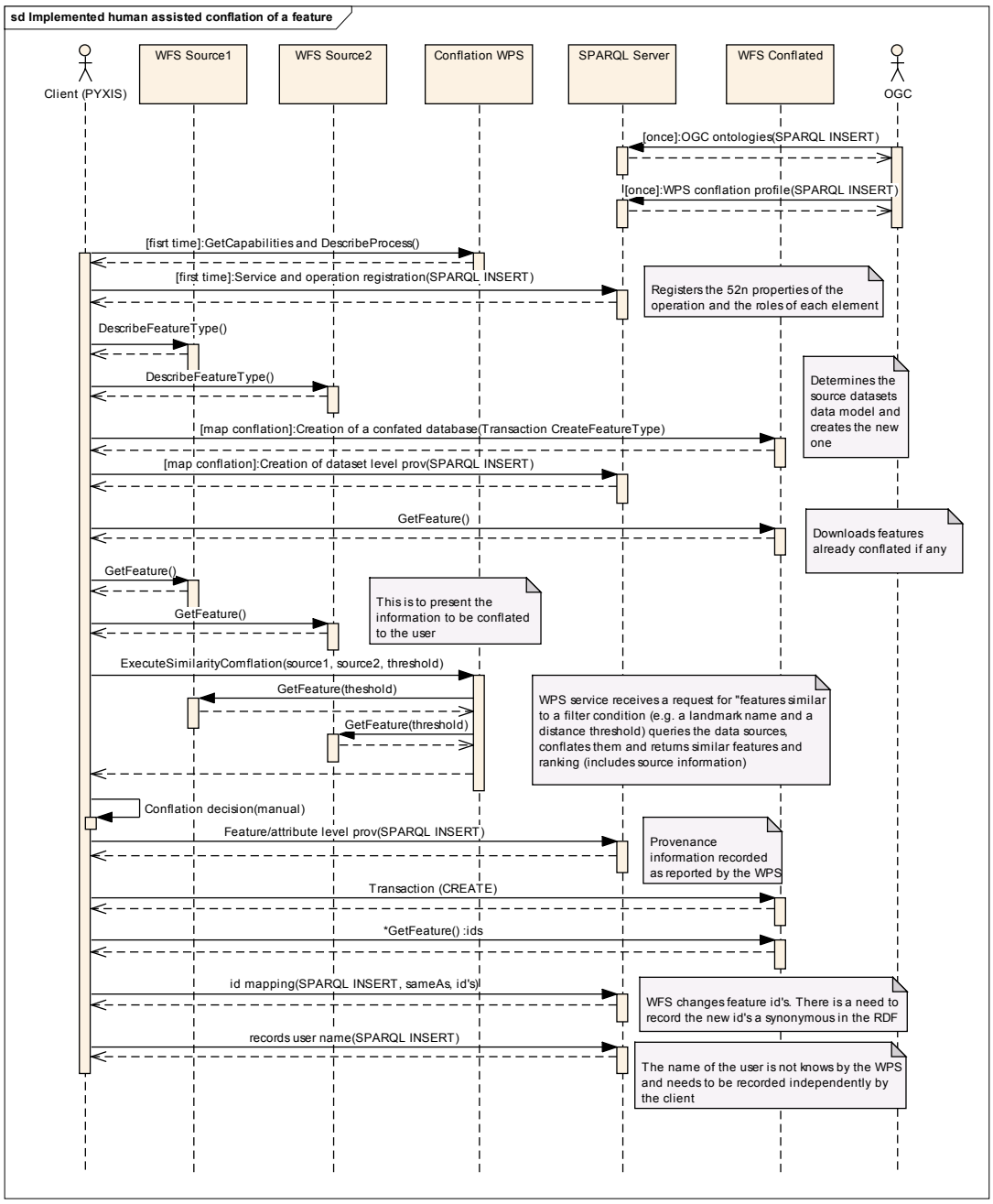

Figure 16 — Human assisted conflation process including storage of provenance

1. The OGC ontologies described in Section 6.1.1 are inserted in the SPARQL server. This has to be only once when the service is setup or at the beginning of the interoperability experiment.
2. The WPS conflation profile described in Section 7.1 are inserted in the SPARQL server. This has to be only once at the beginning of the interoperability experiment.

3. The client (PYXIS) discovers the process services, verifies that conforms to the conflation profile. It gets some metadata about the service (GetCapabilities) and the conflation process (DescribeProcess), in particular about the organization and individual responsible for creating the conflation process and its role on the organization.
4. The client (PYXIS) translates this information into RDF and sends it to the SPARQL server. The result of this translation corresponds to the RDF described in Section 7.2.
5. The client (PYXIS) request properties of the different feature types available in WFS services.
6. The user searches for: feature types from different datasets in a constrained area (e.g. all hospitals in Monterey from a TNM WFS and a TDS WFS).
7. Both datasets are presented in a map using different symbols.

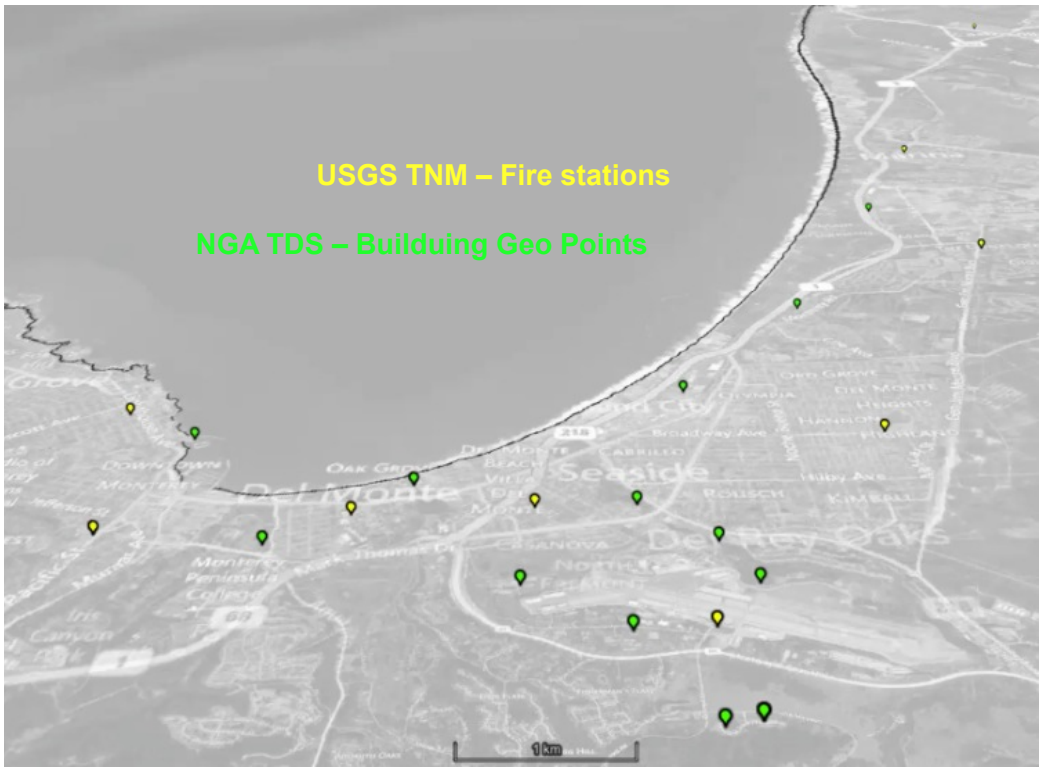

Figure 16 — USG TNM and NGA TDS fire stations presented to the user

8. The client (PYXIS) presents the option to the user to conflate.

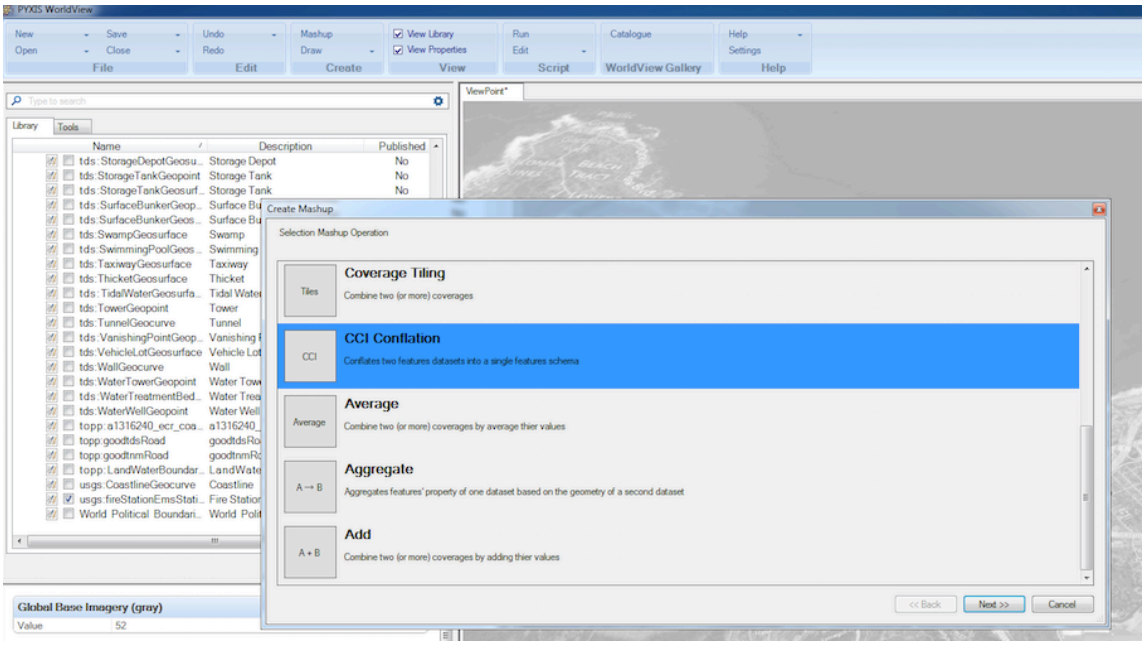

Figure 17 — The conflation option is presented to the user

9. The client (PYXIS) registers the feature type of the conflate results (if the conflated result is saved in a different WFS service as represented in the UML diagram).
10. The client (PYXIS) retrieves previously conflated features (if any conflation process was started before)
11. The client (PYXIS) retrieves previously conflated provenance (if any the conflation process started before).
12. Both datasets are presented in a map using different symbols.
13. The client (PYXIS) registers the dataset level provenance in the SPARQL server. The result of this translation corresponds to the RDF described in Section 7.4.
14. Client connects to a WPS "Synchronously" - and provides the GetFeature requests of the WFS to be conflated as well as the rules for conflation.
15. The WPS retrieves the features and executes the conflation process. The resulting features have the schema of the first (source) dataset.
16. The client presents a map of the conflated features as well as provenance information about the process and features. Both can be obtained by reading the WPS response.
17. The user validates the conflation output and approves the results (with or without modifications).
18. The client (PYXIS) adds the provenance information to a SPARQL-server. The result of this translation corresponds to the RDF described in Section7.5.
19. The client (PYXIS) adds the new features doing a transactional request to the source WFS.

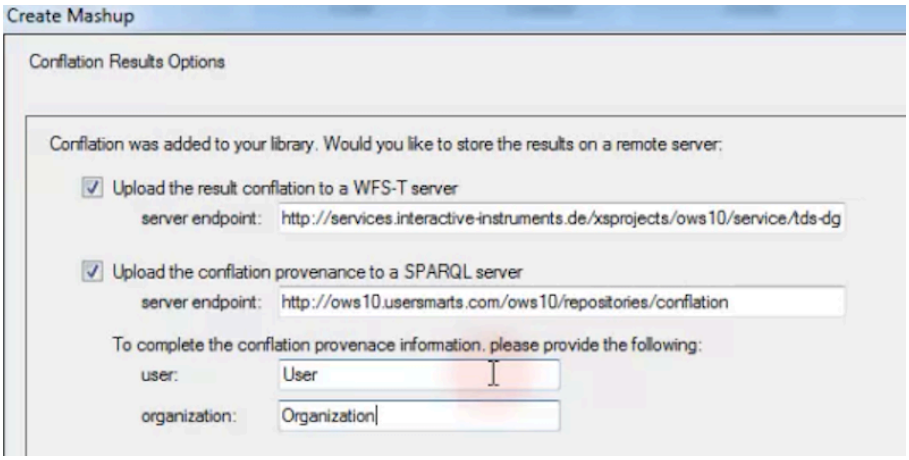

Figure 18 — PYXIS client is ready to send both components of the WPS result to the WFS-T and to the SPARQL service.

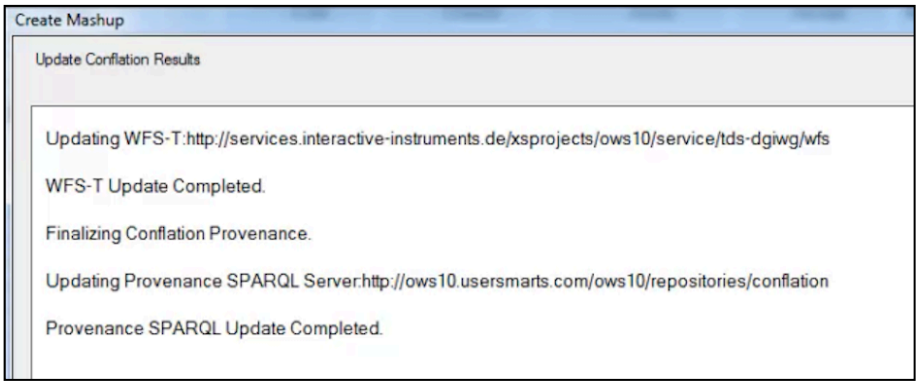

Figure 19 — WFS-T and WPARQL services have completed their tasks and everything is stored

20. The client (PYXIS) determines the new feature id's of the newly created features by requesting them to the WFS.
21. The client adds the equivalences of the feature names used by the WPS in the RDF and the new names registered in the WFS and includes these equivalences as owl:sameAs triples in the SPARQL-server.

8.2 Generating WPS conflation

At least 3 conflation WPS where implemented in the OWS-10 for different purposes. In this section we describe the 52ºNorth implementation in detail.

8.2.1 52North implementation and scenario

In this scenario fire stations from a TDS WFS and a TNM WFS are conflated. All stations from the TNM dataset that do not exist in the TDS one will be added to the resulting conflated dataset. According to the specified rules, attribute values for the new

conflated features will be taken over from the TNM features or the values will be set to a fixed value.

8.2.1.1 User Inputs

The user must enter the following information to start the conflation process:

- **source**
  Source dataset. All features of this dataset will be in the *conflated result* (see Subsection 8.2.1.2). The schema of the data will be used for the features derived from the target dataset. Features are expressed in GML and given to the service as a reference to a WFS.

- **target**
  Target dataset. Features that are non-existing in the source dataset will be added to the conflated result dataset. Features are expressed in GML and given to the service as a reference to a WFS.

- **rules**
  Rules for the conflation process in JSON.

8.2.1.2 Outputs

The following outputs can be requested:

- **conflated_result**
  The resulting conflated dataset that includes all features in the source dataset and some extracted from the target dataset. Features are expressed in GML.

- **provenance**
  Provenance information about the process and involved features and attributes. Provenance is expressed in RDF.

8.2.1.3 Conflation Rules

A JSON-based encoding for conflation rules was developed during OWS-10. With this encoding, some parameters are specified such as: which attribute values are to be taken over from the target features and which attribute values shall be set to fixed values. The structure of the JSON is shown in the following:

```
{ "mappings": {
"attribute-name-target":"attribute-name-source"
},
"fixedAttributeValues": {
"attribute-name-target":"fixed-value"
}
}
```

An example taken from the scenario:

```
{ "mappings": {
"address":"address",
"name":"geoNameCollection.memberGeoName.fullName"
},
"fixedAttributeValues": {
"featureFunction-1":"firefighting",
"restriction.securityAttributesGroup_resClassification" :"U"
}
}
```

The developed concept covers currently only simple rules for conflation scenarios, in which all features from the target dataset that do not exist in the source dataset are added to the result. It can be extended to allow more complex conflation scenarios, e.g. updating of source features/attributes based on rules.

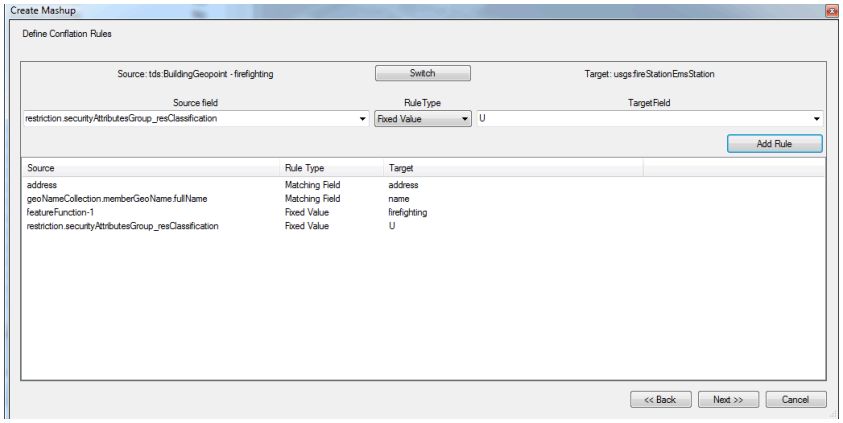

Figure 20 — PYXIS interface to create and edit conflation rules.

8.2.1.4 Example execute request

The following is an example of a Execute request to execute the conflation process. In it you can see the 3 inputs (source, target and rules), and the two elements of the requested response (conflated_result and provenance):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<wps:Execute service="WPS" version="1.0.0"
  xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
  xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsExecute_request.xsd">
  <ows:Identifier>org.n52.wps.server.algorithm.conflation.Kinda_Generic
_ConflationProcess</ows:Identifier>
  <wps:DataInputs>
    <wps:Input>
```



```
      <ows:Identifier>Source</ows:Identifier>
      <wps:Reference mimeType="text/xml"
        schema="http://schemas.opengis.net/gml/3.2.1/gml.xsd"
        xlink:href="
http://services.interactive-
instruments.de/xsprojects/ows10/service/tds-
dgiwg/wfs?service=WFS&amp;version=2.0.0&amp;request=GetFeature&amp;name
spaces=xmlns(tds,http://metadata.dod.mil/mdr/ns/GSIP/3.0/tds/3.0)&amp;t
ypenames=tds:BuildingGeopoint&amp;FILTER=%3CFilter%20xmlns%3D%22http%3A
%2F%2Fwww.opengis.net%2Fogc%22%20xmlns%3Atds%3D%22http%3A%2F%2Fmetadata
.dod.mil%2Fmdr%2Fns%2FGSIP%2F3.0%2Ftds%2F3.0%22%3E%3CPropertyIsEqualTo%
3E%3CValueReference%3Etds%3AfeatureFunction-
1%3C%2FValueReference%3E%3CLiteral%3Efirefighting%3C%2FLiteral%3E%3C%2F
PropertyIsEqualTo%3E%3C%2FFilter%3E">
      </wps:Reference>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>Target</ows:Identifier>
      <wps:Reference mimeType="text/xml"
        schema="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"
        xlink:href="http://portal.cubewerx.com/cubewerx/projects/ows9/c
ubeserv.cgi?service=WFS&amp;datastore=OWS9&amp;request=GetFeature&amp;t
ypename=usgs:fireStationEmsStation&amp;bbox=36.464,-122.003,36.714,-
121.667" />
    </wps:Input>
    <wps:Input>
      <ows:Identifier>Rules</ows:Identifier>
      <wps:Data>
        <wps:LiteralData>{ "mappings": { "address":"address",
"name":"geoNameCollection.memberGeoName.fullName" },
"fixedAttributeValues": { "featureFunction-1":"firefighting",
"restriction.securityAttributesGroup_resClassification" :"U" }
}</wps:LiteralData>
      </wps:Data>
    </wps:Input>
  </wps:DataInputs>
  <wps:ResponseForm>
    <wps:ResponseDocument>
      <wps:Output mimeType="text/xml"
        schema="http://schemas.opengis.net/gml/3.2.1/gml.xsd">
        <ows:Identifier>conflated_result</ows:Identifier>
      </wps:Output>
      <wps:Output mimeType="text/turtle">
        <ows:Identifier>provenance</ows:Identifier>
      </wps:Output>
    </wps:ResponseDocument>
  </wps:ResponseForm>
</wps:Execute>
```

In this execute request, the source and target datasets are specified by WFS HTTP GetFeature requests. Note that a FILTER= parameters is used to only request the fire stations (see Figure 21). The rules are specified in the JSON format described in Subsection 8.2.1.3. As outputs the conflated dataset and the provenance are requested.

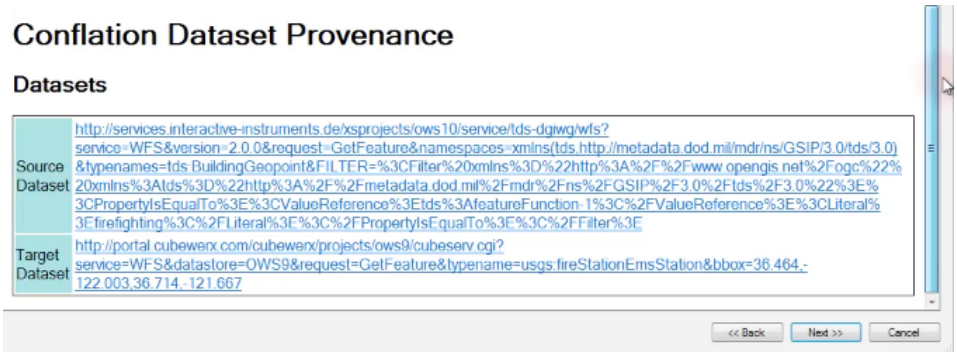

Figure 21 — Source and target to be conflated

8.2.1.5 Process execution

During the process, the execution is iterated over all target features. Whether the feature exists in the source target dataset is checked, by comparing the id/name with the ones from all source features. If the feature does not exist in the source dataset, a new empty feature is created according to the schema of the source dataset. The conflation rules are checked for properties mappings and fixed values and the attribute values of the newly created feature are set accordingly. Attributes that are not mapped and do not have a fixed value assigned have are set to the respective default value. The original id of the target feature is used as the new id. The relationship between the newly created feature and the target feature is preserved by annotating them in the provenance information at feature and attribute level.

8.2.1.6 Example target and result feature

To illustrate the conflation result, an example target feature and the resulting conflated feature will be shown in the following code using their GML representation. Figure 22 shows that this particular execution has found 19 fire stations in the target dataset, those were not in the source dataset.

Target feature:

```
<gml:featureMember>
   <fireStationEmsStation
      gml:id="CWFID.ST_FIRE_STATION.0.7.4D877CAE6A6100961F20020000">
      <geometry>
         <gml:Point srsName="urn:ogc:def:crs:EPSG::4269">
            <gml:pos>36.48157269024836 -121.7315188735717</gml:pos>
         </gml:Point>
      </geometry>
      <permanentIdentifier>1ca8c60b-f636-4437-8daf-
2e81dbe0a21e</permanentIdentifier>
      <sourceFeatureID>10139139</sourceFeatureID>
      <sourceDatasetID>{099A61A7-5FF0-4C55-B6EF-
569B3B790258}</sourceDatasetID>
      <sourceDataDesc>CA fire station update for 29 counties in central
CA,
```



```
            processed by NGTOC August 2010</sourceDataDesc>
         <sourceOriginator>TechniGraphics, Inc.</sourceOriginator>
         <dataSecurity>5</dataSecurity>
         <distributionPolicy>E4</distributionPolicy>
         <loadDate>2010-08-12T00:00:00</loadDate>
         <featureType>740</featureType>
         <featureCode>74026</featureCode>
         <name>Carmel Valley Fire Department</name>
         <isLandmark>0</isLandmark>
         <pointLocationType>4</pointLocationType>
         <adminType>0</adminType>
         <address>26 Via Contenta</address>
         <city>Carmel Valley</city>
         <state>CA</state>
         <zipcode>93924-9566</zipcode>
         <gnisID>2623868</gnisID>
      </fireStationEmsStation>
</gml:featureMember>
```

Conflated feature:

```
<BuildingGeopoint
   gml:id="CWFID.ST_FIRE_STATION.0.7.4D877CAE6A6100961F20020000">
   <gml:metaDataProperty />
   <gml:description>No Information</gml:description>
   <gml:identifier>No Information</gml:identifier>
   <gml:name>No Information</gml:name>
   <gml:boundedBy>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
         <gml:lowerCorner>36.48157269024836 -
121.7315188735717</gml:lowerCorner>
         <gml:upperCorner>36.48157269024836 -
121.7315188735717</gml:upperCorner>
      </gml:Envelope>
   </gml:boundedBy>
   <address>26 Via Contenta</address>
   <aeroObstacleLightPresent>No Information</aeroObstacleLightPresent>
   <angleOfOrientation>-999999.0</angleOfOrientation>
   <area>-999999.0</area>
   <conditionOfFacility>No Information</conditionOfFacility>
   <controllingAuthority>No Information</controllingAuthority>
   <featureFunction-1>No Information</featureFunction-1>
   <featureFunction-2>No Information</featureFunction-2>
   <featureFunction-3>No Information</featureFunction-3>
   <floorCount>-999999</floorCount>
   <geointAssuranceMetadata.processStep.source.resourceContentOrigin>No
Information</geointAssuranceMetadata.processStep.source.resourceContent
Origin>
   <geoNameCollection.memberGeoName.fullName>Carmel Valley Fire
Department</geoNameCollection.memberGeoName.fullName>
   <geoNameCollection.memberGeoName.nameIdentifier>No
Information</geoNameCollection.memberGeoName.nameIdentifier>
   <heightAboveSurfaceLevel>-999999.0</heightAboveSurfaceLevel>
   <highestElevation>-999999.0</highestElevation>
```

```
   <length>-999999.0</length>
   <manufacturingInfo.byProduct-1>No Information
          </manufacturingInfo.byProduct-1>
   <manufacturingInfo.byProduct-2>No Information
          </manufacturingInfo.byProduct-2>
   <manufacturingInfo.byProduct-3>No Information
          </manufacturingInfo.byProduct-3>
   <manufacturingInfo.product-1>No Information
          </manufacturingInfo.product-1>
   <manufacturingInfo.product-2>No Information
          </manufacturingInfo.product-2>
   <manufacturingInfo.product-3>No Information
          </manufacturingInfo.product-3>
   <manufacturingInfo.rawMaterial-1>No Information
          </manufacturingInfo.rawMaterial-1>
   <manufacturingInfo.rawMaterial-2>No Information
          </manufacturingInfo.rawMaterial-2>
   <manufacturingInfo.rawMaterial-3>No Information
          </manufacturingInfo.rawMaterial-3>
   <navigationLandmark>No Information</navigationLandmark>
   <note.memorandum>No Information</note.memorandum>
   <religiousInfo.religiousDesignation>No Information
          </religiousInfo.religiousDesignation>
   <religiousInfo.religiousFacilityType>No Information
          </religiousInfo.religiousFacilityType>
   <restriction.securityAttributesGroup_resClassification>No
Information</restriction.securityAttributesGroup_resClassification>
   <restriction.securityAttributesGroup_resNonIntelComMarkings>No
Information</restriction.securityAttributesGroup_resNonIntelComMarkings
>
   <restriction.securityAttributesGroup_resOwnerProducer>No
Information</restriction.securityAttributesGroup_resOwnerProducer>
   <roofShape-1>No Information</roofShape-1>
   <roofShape-2>No Information</roofShape-2>
   <roofShape-3>No Information</roofShape-3>
   <specifiedEnumerants>No Information</specifiedEnumerants>
   <uniqueEntityIdentifier>No Information</uniqueEntityIdentifier>
   <verticalConstMaterial-1>No Information</verticalConstMaterial-1>
   <verticalConstMaterial-2>No Information</verticalConstMaterial-2>
   <verticalConstMaterial-3>No Information</verticalConstMaterial-3>
   <verticalObstIdentifier>No Information</verticalObstIdentifier>
   <width>-999999.0</width>
   <wirelessTelecomInfo.wirelessTelecomType>No
Information</wirelessTelecomInfo.wirelessTelecomType>
</BuildingGeopoint>
```

The attribute values of the target feature are mapped according to the specified rules. Also the fixed attribute values are set. The ids of the target and conflated feature are the same.

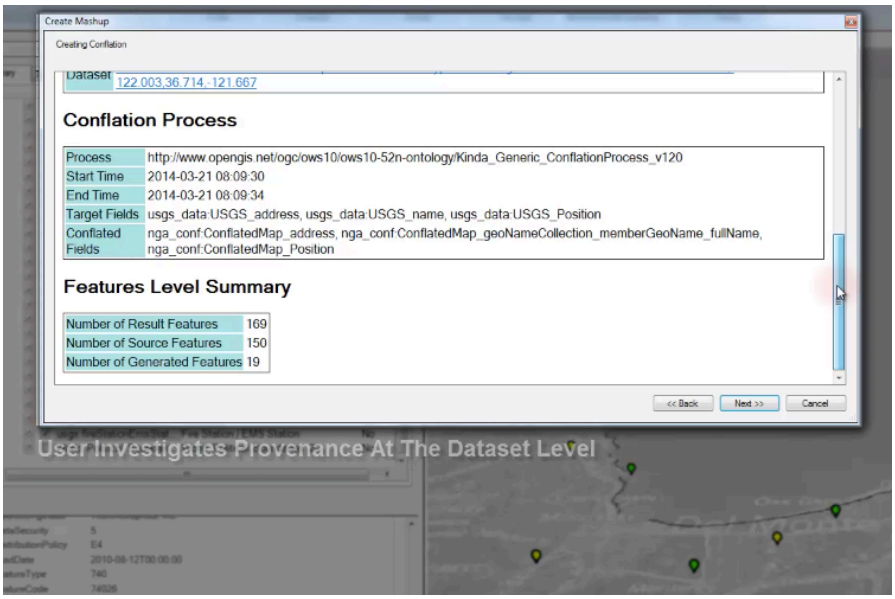

Figure 22 — Some information about the results of the conflation process

8.3 Optimizing the information of the different levels

In Section 6 we have presented a way to encode provenance in W3C PROV RDF in 3 different levels of granularity. It is important to have a set of conventions to minimize the amount of information that has to be stored. This are a small set of conventions suggested so far.

Conventions about inheritance of provenance between levels:

- If all geometrical and non geometrical properties have the same provenance, avoid creating entities for each property, including the hasProperty relation and describing the provenance at the attribute level. Just add the common provenance description at the feature level. This will save a lot of space in the triple store.

- If all features in a feature collection have the same provenance (and do not have specific attribute level provenance, avoid describing the provenance at the feature level (it could be still useful to create entities for each feature, including the hasMember relation to be able to get the name of the feature collection from it). Just add the common provenance description at the feature collection level. This will save a lot of space in the triple store.

Conventions about naming:

- Use a naming convention that takes into consideration the existence of the namespace. For example, each feature name is unique in the namespace of the dataset so there is no need to repeat the name of the dataset in the RDF feature name.

These are just a few examples on how we can make the provenance information more compact. Additional conventions should be explored in future work.

9 Presenting provenance to the user

9.1 Provenance retrieval and presentation

One of the simpler queries that a client can do to is to retrieve the provenance information for a single feature that the user has selected on the screen. This query requires knowing the identifier of the feature in the PROV RDF model. In this case, we can assume that the feature description in GML has been already retrieved and the client already has the gml:id of the feature. Then, the client will query the triple store with the RDF identifier (see Section 6.2 to know more details about how this equivalence is stored). Finally, it will ask for the triples related to the RDF identifier, get the information and present it to the user.

9.1.1 Provenance presentation considering different levels.

In Section 6 we have presented a way to encode provenance in W3C PROV RDF in 3 different levels of granularity and in Subsection 8.3, we have included a set of rules to support provenance inheritance and make the storage of provenance more compact. If the provenance has been stored using these rules the client has to apply them to retrieve the provenance of the different *entities*. In that sense:

- If we query for the provenance of a feature property and we do not find any provenance information at this level we assume that all properties have the same provenance and make a second query to the feature level

- If we query for the provenance of a feature and we do not find any provenance information at this level we assume that all features have the same provenance and make a second query to the dataset level.

9.2 Other queries to provenance

As we discussed, provenance information will be used to represent the origins of geospatial information in conflated datasets at the three different levels. Once the provenance information is represented, specific aspects of provenance can be extracted through the use of queries. For example, a very simple query could be:

- For this feature, show me the provenance information.

The model presented allows to explore more advanced scenarios where users can discover the specific provenance of a feature or an attribute. For example the users can examine the provenance of individual features derived from specific conflation process. So, by keeping provenance records, it will be possible to support requests from users that

require a better understanding of how information was generated. This way, the system can support request such as:

- Show in the map only features or attributes that originated in the USGS dataset.
  - This query refers to *sources* involved in the conflation process.
- Show in the map only features or attributes that originated in government datasets.
  - This query asks about *types of sources* involved in the conflation process
- Show in the map only features or attributes that were conflated by 52N.
  - This query asks about *agents* involved in the conflation process.
- Show in the map only features or attributes that were conflated by a particular conflation algorithm.
  - This query asks about *entities* involved in the conflation process.
- Show in the map only features or attributes that were conflated by a particular conflation rule, like distance threshold.
  - This query asks about *entities* involved in feature and attribute level conflation processes.
- Show in the map only features or attributes that were conflated before Jan 1, 2014.
  - This query asks about *characteristics of the conflation process,* in this case their execution date.
- Show in the map only features or attributes where the original USGS dataset and the OSM dataset were in agreement.
  - This query asks about *information contained in the sources* involved in the conflation process.

A better understanding of the kinds of provenance queries that need to be supported for a given application would determine the appropriate design for a provenance solution, since different solutions have storage and performance tradeoffs as discussed in Section 5.5. Requirements in terms of provenance queries, both technical and user driven, are left for future work.

As an example of how to drive user requirements, a possible demonstration scenario involving provenance would be to show in a user interface a panel the results of a specific set of provenance queries such as the above. For example, a panel with all the original source datasets (dynamically extracted from the provenance records of the conflation processes) and as the user selects one source then all the points in the map that have features/attributes from that data source would be highlighted in green and the points where information from that data source was not selected by the conflation would be highlighted in red. The development and implementation of such scenarios is beyond the scope of the work on OWS-10, and left for future work.

10 Storing provenance using XML

One of the problems that has the implemented approach discussed in Section 7 is the need to store the provenance in one encoding (RDF) and stored in a database (triple store) that is deferent from the encoding (GML) and the database (WFS) used for the geospatial data.

A priori, the use of an XML encoding seems to be more appropriated to generate a more homogeneous environment where all information can be stored in the same format. In the end, we will see that we need 2 independent services anyway and the final architecture is not so different.

This approach was elaborated at the beginning of the OWS 10 activity is included in this document for completeness. It was never implemented (and discarded in favor of the RDF approach previously discussed). We think there is a value on describing the progress made and allow the reader to decide which approach fit better with its use case. Section 11 briefly compares both approaches and the argumentation used to decide in favor of the RDF approach to provenance.

10.1 Alternative encodings

To encode provenance in XML we consider 2 possible alternatives: W3C PROV and ISO19139.

10.1.1 Use of ISO 19139

In ISO realm, the dataset-level of provenance is recorded in the LI_Lineage element of a ISO 19115 metadata record,so all the the metadata is stored together. As presented in Subsection 5.6.1.1, the ISO 19139 provides the XML implementation schema for ISO 19115 specifying the metadata record format. The schema is used to define and validate the geospatial metadata structure prepared in XML. This section provides some guidelines of how to include provenance information in XML metadata records encoded in ISO 19139. In ISO 19115, the provenance information is part of the *DQ_DataQuality* and is contained in the *gmd:lineage* sub-element, which is composed by three subelements: *statement, processSteps and sources*. Here we just depict the most relevant aspects, paying special attention to the lineage parts of a conflated dataset. The complete XML metadata archive can be found in the Annex D.

- An ISO19115 metadata document that describes a *dataset* is identified as *OWS10conflatedmap.*

```
<gmd:fileIdentifier>
        <gco:CharacterString>USGSconflatedMap</gco:CharacterString>
</gmd:fileIdentifier>
```


- The *gmd:statement* is an unstructured text that can describe briefly the sources and process that participated in the elaboration of the dataset. Even if this can be useful for a human, it is almost useless for a machine that will rather work with a more structured representation of the provenance.

```
<gmd:DQ_DataQuality>
    <gmd:lineage>
        <gmd:LI_Lineage>
            <gmd:statement>
                <gco:CharacterString>The dataset is a result of a
conflation WPS instance between an USGS Map Data and Open Street
Map</gco:CharacterString>
            </gmd:statement>
        </gmd:LI_Lineage>
    </gmd:lineage>
</gmd:DQ_DataQuality>
```

- Alternatively or complementary to the *statement*, process steps involved on the creation of the dataset can be enumerated and described. Concretely, this dataset is a result of *52North WPS conflation instance,* the 52N_ConflationExecution20140305. The schema also records the time of execution, the responsible parties, and what is more important, the sources that have been used in this process*, SC_USGS, SC_OSM*.

```
<gmd:DQ_DataQuality>
   <gmd:lineage>
      <gmd:LI_Lineage>
         <gmd:processStep>
            <gmd:LI_ProcessStep id="52N_ConflationExecution20140305">
               <gmd:description>
                  <gco:CharacterString>52North WPS conflation instance
                    </gco:CharacterString>
               </gmd:description>
               <gmd:dateTime>
                  <gco:DateTime>2014-03-05T16:05:00</gco:DateTime>
               </gmd:dateTime>
               <gmd:processor>
                  <gmd:CI_ResponsibleParty id="NGA_davidClient">
                     <gmd:individualName>
                        <gco:CharacterString>David</gco:CharacterString>
                     </gmd:individualName>
                     <gmd:organisationName>
                         <gco:CharacterString>NGA</gco:CharacterString>
                     </gmd:organisationName>
                     <gmd:role>
```

```
                    <gmd:CI_RoleCode
codeList="http://www.ngdc.noaa.gov/metadata/published/xsd/schema/resour
ces/Codelist/gmxCodelists.xml#CI_RoleCode"
codeListValue="processor">processor</gmd:CI_RoleCode>
                  </gmd:role>
                </gmd:CI_ResponsibleParty>
              </gmd:processor>
              <gmd:source xlink:href="#usgs_USGSMap" />
              <gmd:source xlink:href="#osm_OSMMap" />
            </gmd:LI_ProcessStep>
          </gmd:processStep>
        </gmd:LI_Lineage>
    </gmd:lineage>
</gmd:DQ_DataQuality>
```

- Finally, the schema describes the sources used in 52North WPS conflation instance: the Geological Soils Maps of United States, *the* usgs_USGSMap, and the OpenStreetMap, the osm_OSMMap.

```
<gmd:DQ_DataQuality>
   <gmd:lineage>
      <gmd:LI_Lineage>
         <gmd:source>
            <gmd:LI_Source id="usgs_USGSMap">
               <gmd:description>
                  <gco:CharacterString>The U.S.Geological Survey Maps
of United States</gco:CharacterString>
               </gmd:description>
               <gmd:sourceCitation>
                  <gmd:CI_Citation>
                     <gmd:title>
                        <gco:CharacterString>USGS map
data</gco:CharacterString>
                     </gmd:title>
                     <gmd:date/>
                  </gmd:CI_Citation>
               </gmd:sourceCitation>
               <gmd:sourceStep
xlink:href="#52N_ConflationExecution20140305"/>
            </gmd:LI_Source>
            <gmd:LI_Source id="osm_OSMMap">
               <gmd:description>
                  <gco:CharacterString>OpenStreetMap (OSM) is a
collaborative project to create a free editable map of the
world.</gco:CharacterString>
               </gmd:description>
               <gmd:sourceCitation>
                  <gmd:CI_Citation>
                     <gmd:title>
                        <gco:CharacterString>Open Street
Map</gco:CharacterString>
                     </gmd:title>
```



```
                        <gmd:date/>
                     </gmd:CI_Citation>
                  </gmd:sourceCitation>
                  <gmd:sourceStep
xlink:href="#52N_ConflationExecution20140305"/>
               </gmd:LI_Source>
            </gmd:source>
         </gmd:LI_Lineage>
   </gmd:lineage>
</gmd:DQ_DataQuality>
```

10.1.2 Use W3C PROV XML

W3C PROV describes a way to encode provenance using the same data model but instead of encoding it in RDF, encode it in XML. This way, objects and their relations are expressed in a XML notation.

This section does not require much explanation since the conceptual model used will be the same that the one described on the RDF notation part (see Section 6) but with a different notation.

This way the RDF definition of the road_66 as an entity,

```
usgs:road_66 a prov:Entity ;
```

it is encoded it like this in PROV XML:

```
<prov:entity prov:id= "usgs.road_66">
   <prov:label>USGS road 66</prov:label>
</prov:entity>
```

To say that the road_66 was generated by a WPS conflation operation execution, that was previously stated as RDF as:

```
usgs_usgs:road66 prov:wasGeneratedBy f2n:52N_ConflationExecution01 .
```

it is encoded it like this in PROV XML:

```
<prov:wasGeneratedBy>
   <prov:entity prov:ref="usgs.road_66"/>
   <prov:activity prov:ref="f2n.52N_ConflationExecution01"/>
</prov:wasGeneratedBy>
```

It is also possible to define subClassing by using equivalent mechanism in XML.

It is possible to both embed this encoding in GML an also, in theory, story it in an ebRIM CSW catalogue configured to support W3C PROV objects.

10.2 Alternative storage methods

This Subsection will expose 2 different strategies to store provenance information explored in OWS10.

10.2.1 Storing provenance in an independed catalogue

Normally, ISO 19115 metadata documents are stored in CSW catalogues. In particular, catalogues conformal to OGC Catalogue Services Specification 2.0.2 - ISO Metadata Application Profile (1.0.0) are able to return the XML encoded documents.

XML encodings for W3C PROV can be stored in an ebRIM catalogue if this is adapted to support the W3C PROV data model. Nobody was willing to try this approach and we were not able to assess the difficulty of doing this.

10.2.2 Storing provenance information embedded in GML

10.2.2.1 Storing ISO 19115 lineage information embedded in GML

GML offers the possibility to embed an ISO document directly in a feature or a feature collection by defining a complex property type derived from AbstractMetadataPropertyType.

```
<complexType name="SpringISOMetadataPropertyType">
  <complexContent>
    <extension base="gml:AbstractMetadataPropertyType">
      <sequence>
        <element ref="gmd:MD_Metadata" minOccurs="0"/>
      </sequence>
      <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </extension>
  </complexContent>
</complexType>
```

and use it to create of property for a feature or a feature collection

```
<complexType name="SpringType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <element name="metadata"
           type="xmp:SpringISOMetadataPropertyType" minOccurs="0"/>
        <element name="name" type="string" minOccurs="0"
           maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

This way it is possible to embed a metadata document in a GML document:


```
<xmp:Spring>
   <xmp:metadata>
      <gmd:MD_Metadata>...
   </xmp:metadata>
   <xmp:name>Silver spring</xmp:name>
</xmp:Spring>
```

The same trick can be used to embed just the provenance information or just a source or a process step:

```
<complexType name="SpringISOSourcePropertyType">
   <complexContent>
      <extension base="gml:AbstractMetadataPropertyType">
         <sequence>
            <element ref="gmd:LI_Source" minOccurs="0"/>
         </sequence>
         <attributeGroup ref="gml:AssociationAttributeGroup"/>
      </extension>
   </complexContent>
</complexType>
```

So that allows them to enumerate sources and process steps:

```
<xmp:Spring gml:id="GoldSpring">
   <xmp:isoSource>
      <gmd:LI_Source>...
   </xmp:isoSource>
   <xmp:isoProcesStep
      <gmd:LI_Process>...
   </xmp:isoProcesStep
   <xmp:position>...
   <xmp:name>Gold spring</xmp:name>
</xmp:Spring>
```

Nevertheless this strategy can be very verbose, because it forces us to repeat the description of the process (and source) for each feature. Fortunately, there is xlink mechanism not just in GML also in the ISO19139 (even if the later is rarely used). This mechanism allows referencing provenance descriptions in other document by using just an id avoiding unnecessary repetitions.

```
<xmp:Spring gml:id="ID_SPRING_1">
   <xmp:isoSource
xlink:href="https://bob.opengeospatial.org/SpringDatasetISOLineage.xml#
SC_USGS"/>
   <xmp:isoProcesStep
xlink:href="https://bob.opengeospatial.org/SpringDatasetISOLineage.xml#
PS_WPS_1"/>
   <xmp:position>...
   <xmp:name>Gold spring</xmp:name>
</xmp:Spring>
```


10.2.2.2 Storing XML encoded W3C PROV provenance information embedded in GML

As discussed in Subsection 10.1.2, it is possible to encode W3C PROV in XML. In this interoperability experiment we also experimented with the possibility to embed PROV information directly in GML encoded features. GML offers the possibility to embed an XML metadata in a feature or a feature collection by defining a complex property type derived from AbstractMetadataPropertyType.

```
<complexType name="SpringW3CSourcePropertyType">
   <complexContent>
      <extension base="gml:AbstractMetadataPropertyType">
         <sequence>
            <element ref="prov:entity" minOccurs="0"
maxOccurs="unbounded"/>
         </sequence>
         <attributeGroup ref="gml:AssociationAttributeGroup"/>
      </extension>
   </complexContent>
</complexType>
<complexType name="SpringW3CProcessPropertyType">
   <complexContent>
      <extension base="gml:AbstractMetadataPropertyType">
         <sequence>
            <element ref="prov:activity" minOccurs="0"/>
         </sequence>
         <attributeGroup ref="gml:AssociationAttributeGroup"/>
      </extension>
   </complexContent>
</complexType>
```

and use it to create of property for a feature or a feature collection

```
<complexType name="SpringType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <element name="source"
           type="xmp:SpringW3CSourcePropertyType" minOccurs="0"/>
        <element name="name" type="string" minOccurs="0"
           maxOccurs="unbounded"/>
        <element name="processStep"
           type="xmp:SpringW3CProcessPropertyType" minOccurs="0"/>
        <element name="name" type="string" minOccurs="0"
           maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

This way it is possible to embed a metadata document in a GML document:

```
<xmp:Spring>
   <xmp:source>
```



```
        <prov:entity>...
    </xmp:source>
    <xmp:processStep>
        <prov:activity>...
    </xmp:processStep>
    <xmp:name>Platinum spring</xmp:name>
</xmp:Spring>
```

NOTE This encoding does not follow the object-property alternated notation where the objects have names in upper camel case and properties in lower camel case. The reason for this is that W3C PROV XML encoding does not follow this in their application schema.

Again, we can use the xlink mechanism. This mechanism allows referencing provenance descriptions in other document by using just an id avoiding unnecessary repetitions.

```
<xmp:Spring gml:id="ID_SPRING_2">
    <xmp:source
xlink:href="https://bob.opengeospatial.org/SpringDatasetW3CLineage.xml#
SC_USGS"/>
    <xmp:processStep
xlink:href="https://bob.opengeospatial.org/SpringDatasetW3CLineage.xml#
PS_WPS_1"/>
    <xmp:position>...
    <xmp:name>Gold spring</xmp:name>
</xmp:Spring>
```

10.3 Storing provenance at different levels

As presented in the Subsection 5.5, provenance has different levels of granularity and it can be recorded at dataset, feature and attribute levels. In this section, we describe these levels and some strategies to include provenance in each of them. The dataset level includes the process steps involved and acquires the information of the inputs and outputs of the complete conflation WPS process as whole (input and output datasets) following the ISO 19115 standard and encoded in an ISO19139 document related with the dataset as a whole. However some features have different provenance information that is more detailed that the provenance provided at the dataset level. In some cases, feature attributes can have their own specific provenance.

Recording provenance at feature and attribute level requires a high quantity of memory and database space, and the feature and attribute metadata can many times be reiterative, duplicating several sources and processes descriptions of the provenance (and other metadata properties out of scope of this document such as quality indicators). To solve this issue, this proposal exposes a solution to reduce the space required to record the metadata at the feature and attribute level, and diminish redundancies by means of the xlink XML attribute previously presented.

10.3.1 Dataset level

We recommend using a full ISO 19115 metadata record encoded in XML that will contain the provenance information at a dataset level. There are two ways of doing this:

modify the GML application schema or use the deprecated metaDataProperty. In both cases, we can opt to include embed a full metadata record or to use xlink to link to an external metadata document that can be a file or a record in a metadata CSW catalogue.

10.3.1.1 Personalized application schema

To do this, we just need to include a MD_Metadata object as a property of the feature collection.

```
<element name="SpringCollection" type="xmp:SpringCollectionType"
substitutionGroup="gml:AbstractFeature"/>
   <complexType name="SpringCollectionType">
      <complexContent>
         <extension base="gml:AbstractFeatureType">
            <sequence>
               <element name="metadata"
type="xmp:SpringISOMetadataPropertyType" minOccurs="0"/>
               <element name="provenance"
type="xmp:SpringW3CProvPropertyType" minOccurs="0"/>
               <element name="spring" type="xmp:SpringPropertyType"
minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>


<?xml version="1.0" encoding="UTF-8"?>
<xmp:SpringCollection gml:id="OWS-10Conflatedmap">
   <xmp:metadata
xlink:href="https://portal.opengeospatial.org/wiki/pub/OWS10/OWS10CCIPr
ovenance/SpringDatasetISOLineage.xml"/>
   ...
</xmp:SpringCollection>
```

10.3.1.2 Using metaDataProperty

In some cases we will not have the capability to modify the application schema. For instance because we are working with an external WFS or we are using a GML profile (e.g. in the Gazetteer case). In this case there is no other way than use the deprecated metaDataProperty element as a feature collection level.

```
<?xml version="1.0" encoding="UTF-8"?>
<xmp:SpringCollection gml:id="OWS-10Conflatedmap">
   <gml:metaDataProperty
xlink:href="https://portal.opengeospatial.org/wiki/pub/OWS10/OWS10CCIPr
ovenance/SpringDatasetISOLineage.xml">
   ...
</xmp:SpringCollection>
```


10.3.2 Feature level

10.3.2.1 Having controls of the applications schema

If we have control of the application schema, we can personalize it to store provenance at feature level. We include new optional tags for sources and processes that are already described as a feature level so only links to the right elements of the dataset level metadata will suffice.

```
<xmp:Spring gml:id="ID_SPRING_1">
   <xmp:isoSource
xlink:href="https://bob.opengeospatial.org/SpringDatasetISOLineage.xml#
SC_USGS"/>
   <xmp:isoProcesStep
xlink:href="https://bob.opengeospatial.org/SpringDatasetISOLineage.xml#
PS_WPS_1"/>
   <xmp:position>...
   <xmp:name>Gold spring</xmp:name>
</xmp:Spring>
```

10.3.2.2 Reusing a preexisting applications schema

In some cases we will not have the capability to modify the application schema. In this case there is no other way than use the deprecated metaDataProperty element as a feature collection level. But even if we do so, this is a bit trickier because we want to do an xlink to the sources and process steps of the dataset level but the reader can not anticipate the purpose of this links until it reads the actual objects that are linked.

Specifically, we have to make use of the following attributes: "xlink:href", "xlink:role" and the "xlink:arcrole" to fully describe the relation between the Gazetteer features to the provenance elements in the dataset level provenance file (either in ISO or W3C format). (Some clues about the meaning or role and arcrole can be found in the W3C xlink standards but they are not so clear to the authors of this document. This topic is also discussed in section 8.1 "Xlinks — Object associations and remote properties" of GML 3.2.1)

We have to be aware that OGC already discussed encoding property metadata in GML features in 2006/2007 without any emerging best-practice. Three documents in the OGC portal discuss this topic:

- 06-001r1 Use case for GML feature properties metadata (Arliss Whiteside).
    - The document discusses which elements of the ISO19115 are more likely to be useful as "Feature property metadata" (feature level) and "Feature property value types" (attribute level)
    - Table 3 lists the metadata items from [ISO 19115] that they think are likely to be useful for feature level metadata. Interesting to note than most

of the elements of dataQualityInfo are listed there including "lineage" elements such as "processStep" but NOT "source". Table 4 includes the items for attribute level metadata (mainly DQ_Element subclasses.

- 07-069 Property metadata - patterns and encodings (Simon Cox)
    - The document provides rules for using role and arcrole in "Feature property value types" (attribute level). The guidance seems to avoid the need for including the ISO 19139 schema by just referencing the URN of a metadata element to use: <myns:custodian xlink:href="http://www.granddomain.gov/parties/yip99.xml" xlink:arcrole="urn:ogc:def:arcrole:OGC:original" link:role="urn:ogc:def:identifiableObjectType:ISOTC211:CI_ResponsibleParty">Fred. It also suggest mixing the inline and byref properties, a thing that is possible in GML but it has to represent the same thing. this is implied in GML 3.2.1: " GML property elements which follow this pattern may be used to attach values either inline or by-reference". "If both a link and content are present in an instance of a property element, then the object found by traversing the xlink:href link shall be the normative value of the property. The object included as content shall be used by the data recipient only if the remote instance cannot be resolved; this may be considered to be a "cached" version of the object.". In the example given in 07-069 the xlink represents a MD document and the in-line value only a CI_ResponsibleParty so I'm finding this confusing.
- 07-083 Use of xlink in GML – Profile for file-based data content (Andrew Woolf)
    - This document describes how xlink to values that are in external files (such as points that are in a netCDF file). It is using a logic in the arcrole that is similar to what we proposed in the section "Approach that forces the semantic of the role and arcrole" (now abandoned) to reference a property value type from a feature metaDataProperty (even if we used "role" instead of "arcrole"; probably incorrectly).

The next bullets summarize our conclusions about this topic. If this pattern is never used, role and arcrole values would need to be registered with the OGC-NA.

- The xlink:href points to the specific source id in a ISO or W3C remote document that specifies where the feature came from:
    - e.g.: xlink:href="https://portal.opengeospatial.org/wiki/pub/OWS10/OWS10CCIProvenance/SpringDatasetISOLineage.xml#SC_USGS" or xlink:href="https://portal.opengeospatial.org/wiki/pub/OWS10/OWS10CCIProvenance/SpringDataSetW3Cprov.xml#SC_USGS".

- The xlink:role identifier (http URI) of the data type of the resource.
    - Alternative 1: We build it by composing the metadata namespace URI with the name of the target element (just the final element and not the xpath) xlink:arcrole="http://www.isotc211.org/2005/gmd/LI_Source" or xlink:arcrole="http://www.w3.org/ns/prov#entity"
    - Alternative 2: We register a new URI in the OGC-NA: "http://www.opengis.net/def/dataType/iso19115/2003/LI_Source" or "http://www.opengis.net/def/dataType/w3cprov/2012/entity"
- The xlink:arcrole identifier (http URI) of the link relation (also registered in the OGC-NA): e.g.: http://www.opengis.net/def/rel/ogc/source (common for both ISO and W3C? approaches)
- The xlink:title provides a Human readable title
    - e.g. xlink:title="Source for the alternative name".

Finally, for the feature level this will look like this:

```
<gml:metaDataProperty
xlink:href="https://portal.opengeospatial.org/wiki/pub/OWS10/OWS10CCIPr
ovenance/SpringDatasetISOLineage.xml#SC_USGS"
xlink:role="http://www.opengis.net/def/dataType/iso19115/2003/LI_Source
" xlink:arcrole="http://www.opengis.net/def/rel/ogc/source"
xlink:title="USGS map data"/>
```

10.3.2.2.1 Considerations about the use of gml:remoteSchema

At the beginning we incorporated the need to document gml:remoteSchema in the gml:metaDataProperty but we discarded that for the following reasons suggested by Clemens Portele:

- gml:remoteSchema has been deprecated some time ago. In fact gml:metaDataProperty is also deprecated but for a different set of reasons. Since there is the need to link to metadata in the Gazeetter schema, this is almost our only chance to embed it directly in a Gazeetter response.
- remoteSchema was deprecated because it duplicates the semantics of xlink:role. From the GML 3.2.1 release notes: "The 'remoteSchema' attribute has been deprecated as the xlink:role attribute can be used for the same purpose (where needed)."
- We think that gml:remoteSchema value is not needed. If I dereference the href I will see what it is and the associated schema. If the resource referenced by the URI supports content negotiation I may even access both PROV or 19139 representations of the same resource.
- In case we end up using it, the values of gml:remoteSchema should be absolute URIs, e.g. "http://schemas.opengis.net/iso/19139/20070417/gmd/gmd.xsd" or "http://www.w3.org/ns/prov.xsd"

10.3.2.2.2 Considerations about the use of the other attributes of an xlink type=simple

xlink:show and xlink:actuate are just providing clues on how and when the link has to be executed by clients and are of no use here

10.3.3 Attribute level

10.3.3.1 Having controls of the applications schema

At the attribute level, as we did in the feature level, we proposose to change the GML feature model and its schema to store provenance at attribute level. This proposal suggest the incorporation of a new optional tag that relates the atributte level with the ISO data set level.

<xmp:ISOsource
xlink:href="https://portal.opengeospatial.org/wiki/pub/OWS10/OWS10CCIProvenance/StringDatasetISOLineage.xml#SC_USGS

- The same for the W3C PROV model.

<xmp:W3Csource
xlink:href="https://portal.opengeospatial.org/wiki/pub/OWS10/OWS10CCIProvenance/SpringDataSetW3Cprov.xml#SC_USGS"/> for the W3C PROV document.

10.3.3.2 Reusing a preexisting applications schema

In this cases we are forced to use the xlinx in the metaDataProperty of the feature containing the attributes and the specify the scope to the right attribute. The attribute level provenance level is particularly challenging due to the fact that we need to better define the scope of the provenance (the part of the element that is affected by the provenance link). The gml:metaDataProperty adopts the xlink "type=simple" approach for the xlink so it assumes that the source of the link is the current element (the feature instance, in this case) and the scope can not be redefined. If the xlink in the metaDataProperty was and extended one with type="arc", it could be possible to specify a from attribute so it could be used to set the scope to a property of the feature. This is not the case so this alternative is not possible. Another alternative that we experimented with was to force the semantics of the "xlink:role" and the "xlink:arcrole" to use one of them to specify the from and another to specify the to of the relation. With this approaches consensus by the group was not achieved.

We finally found another solution that is more verbose but it is clearer. Use the metaDataProperty of the feature but point to a "gmd:DQ_DataQuality" that contains a

scope element with a subelement gmd:scopeDescription (and a xlink to the attribute name that is affected) in combination with a gmd:source and /or a gmd:processStep:

```
   <gml:metaDataProperty xmlns:gmd="http://www.isotc211.org/2005/gmd">
       <gmd:DQ_DataQuality>
         <gmd:scope>
            <gmd:DQ_Scope>
               <gmd:level>
                  <gmd:MD_ScopeCode
codeList="http://www.isotc211.org/2005/resources/codeList.xml#MD_ScopeC
ode" codeListValue="attribute"/>
               </gmd:level>
               <gmd:levelDescription>
                  <gmd:MD_ScopeDescription>
                     <gmd:attributeInstances
xlink:href="geographicIdentifier"/>
                  </gmd:MD_ScopeDescription>
               </gmd:levelDescription>
            </gmd:DQ_Scope>
         </gmd:scope>
         <gmd:lineage>
            <gmd:LI_Lineage>
               <gmd:source
xlink:href="https://portal.opengeospatial.org/wiki/pub/OWS10/OWS10CCIPr
ovenance/SpringDatasetISOLineage.xml#SC_USGS"
xlink:role="http://www.opengis.net/def/dataType/iso19115/2003/LI_Source
" xlink:arcrole="http://www.opengis.net/def/rel/ogc/source"
xlink:title="USGS map data"/>
            </gmd:LI_Lineage>
         </gmd:lineage>
      </gmd:DQ_DataQuality>
   </gml:metaDataProperty>
```

NOTE: This approach allows specifing more than one link for provenance (more than one metaDataProperty per feature). There is an issue with the WFS-G spec as the element in a transaction does not account for updating properties with multiplicity greater than one.

11 Discussion between W3C PROV and ISO 19115

These are the main reasons why we decided to use W3C PROV instead of ISO Metadata.

- Current implementations and architectures using GML will not be able to incorporate the ISO metadata model without deep modifications.
- Combining the ISO metadata model with GML becomes very verbose and does not completely satisfy the requirements of attribute level provenance.
- Given those difficulties, it is hard to reach a consensus on how to implement attribute level provenance with the ISO metadata model so this was not pursued in the OWS-10 implementations.
- W3C PROV is a conceptual model for provenance that offers an elegant and flexible solution for linking provenance information to geospatial elements with

the necessary semantics. It can be realized in RDF, XML, and text form, giving alternative options for implementing the same model.

- The RDF serialization of W3C PROV enables adding and changing provenance information flexibly as it becomes available over time, which accommodates continuous conflation processes and updates

- Although there is an XML encoding for W3C PROV that was explored in this work, it lacks the flexibility of the RDF serialization and was not pursued in the OWS-10 implementations.

12 Future work

These are aspects that have been identified as requiring future work:

- Exploring alternative approaches for making the provenance information more compact at the feature and attribute levels.

- Develop a set of requirements for provenance based on provenance queries desirable for geospatial applications. These queries should include not only queries to provenance information but also queries that combine provenance with other information in the system.

- Explore different approaches to provenance (storage compression, redundancy) and analyze the impact of performance on provenance queries.

- Address provenance to accommodate the evolution of original data sources over time, which implies re-running the conflation processes and incremental updates (additions, updates, and retractions) in new versions of the map.

- Include considerations about provenance of coverages or coverage elements or region.

- Consider different user requirements and alternative presentations of provenance information in a geospatial interface.

Annex A

Conflation use case provenance diagrams

A.1 General

This annex present the complete W3C PROV diagram developed to encode provenance (presented in Section 6) separately in order to allow a better view of the model. The A.2 subsection contains the legend of the schema. Section A.3 represents the entities, features collection and the activities. Finally the A.4 presents the Agent and Plan Collection part.

A.2 Legend

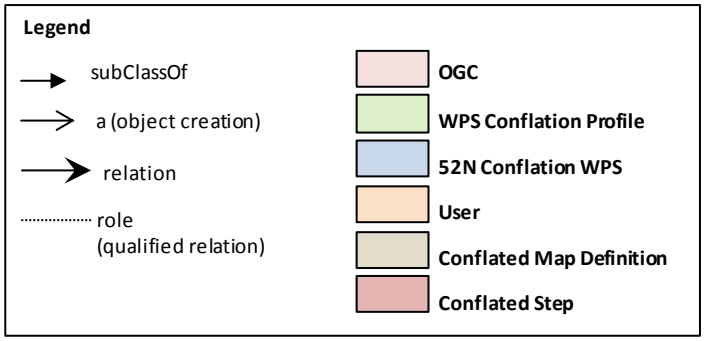

Figure A.2 — Legend of the different levels of the following provenance diagrams

A.3 Conflation process diagram: entities and activities.

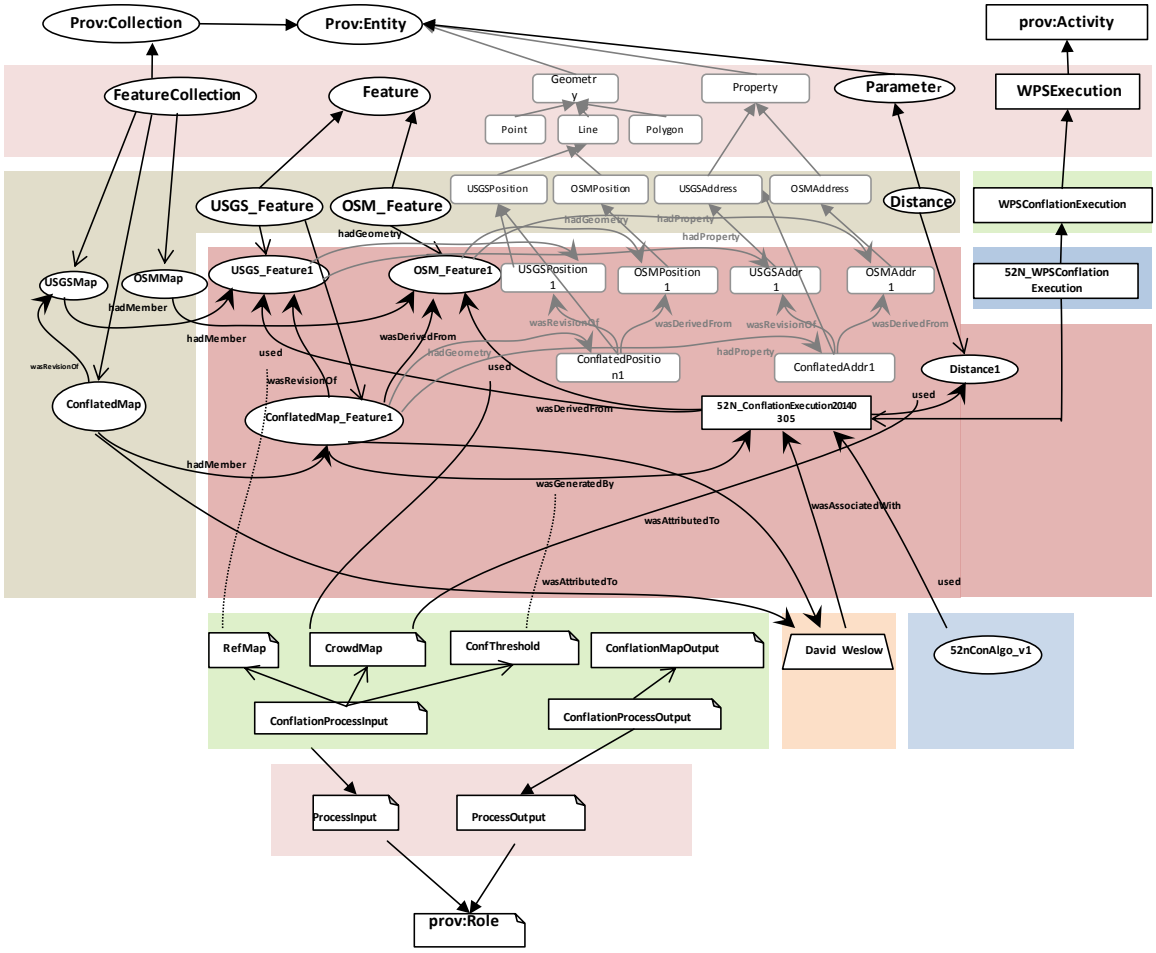

Figure A.3 — Entities and activity provenance diagrams

A.4 Conflation process diagram: Agent and Plan collections

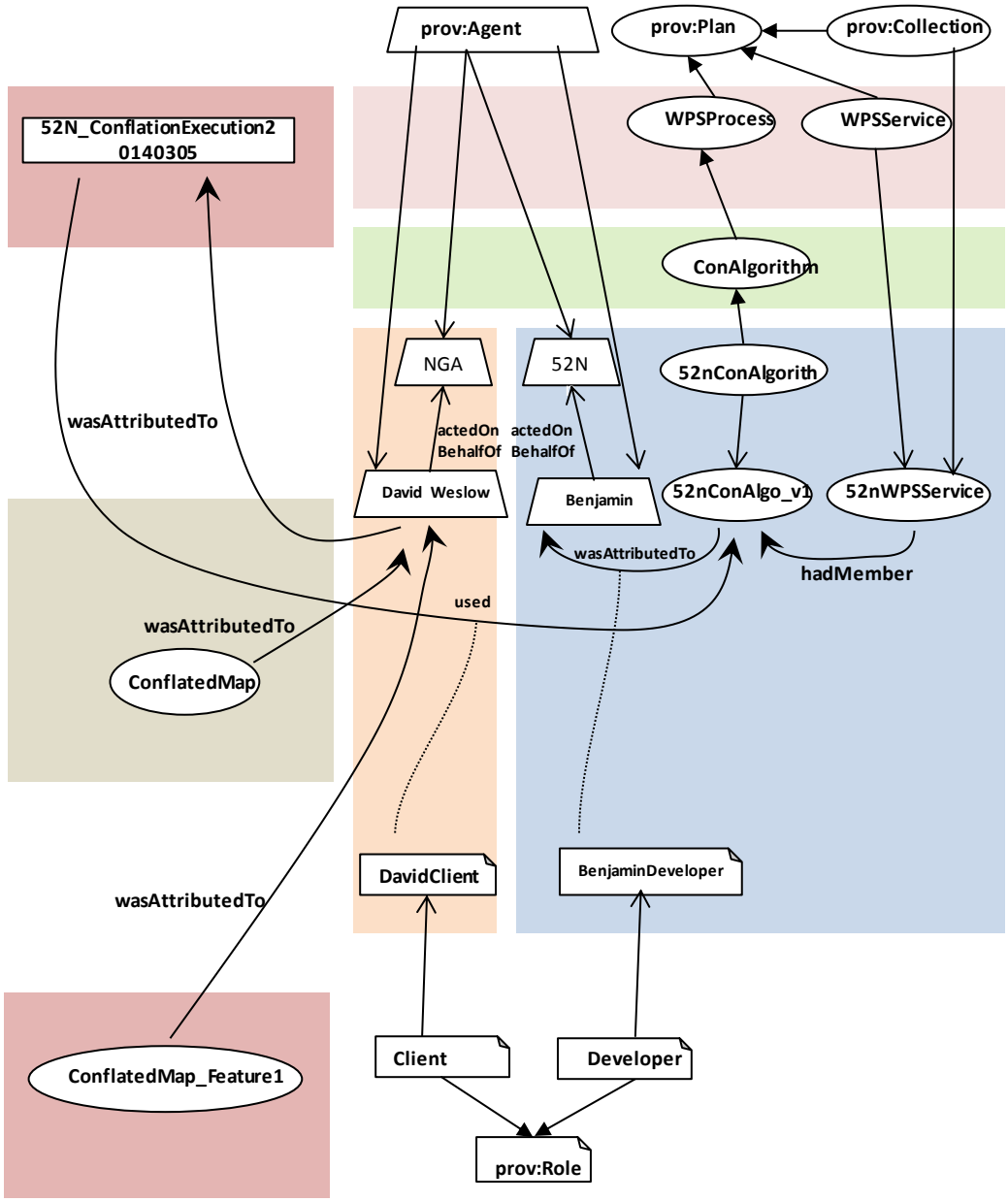

Figure A.4 — Agents, plans and service collections provenance diagram

Annex B

OGC RDF PROV Model

B.1 General

The RDF encoding of PROV are used to express provenance of the geospatial objects, roles and relations. This Annex contains the complete RDF encoding corresponding to the OGC and WPS concepts. The B.2.1 and B.2.2 Subsections have, respectively, the complete RDF encoding of the 6.1.1 and 7.1 sections of the main document.

B.2 RDF encoding

B.2.1 OGC RDF Encoding

0_Prov_OGC_RDF.n3 file content. See Section 6.1.1 for a description.

```
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .

@prefix ows: <http://www.opengis.net/ogc/ows/ows-core-ontology/>



#################################################################
#                               OGC types
#################################################################

# This layer contains all ogc types for WPS services, and for features

# FeatureCollection
# http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-collection
ows:FeatureCollection rdfs:subClassOf prov:collection .

# Feature
ows:Feature rdfs:subClassOf prov:Entity .

# Geometries
ows:Geometry rdfs:subClassOf prov:Entity .
# Geometry
ows:Point rdfs:subClassOf ows:Geometry .
ows:Line rdfs:subClassOf ows:Geometry .
ows:Polygon rdfs:subClassOf ows:Geometry .

# Non Geometrical Property
ows:Property rdfs:subClassOf prov:Entity .
```



```
# The only new relations that we needed to include in W3C prov
ows:hadGeometry a owl:ObjectProperty ;
       rdfs:comment "A geometric property of a resource" ;
       rdfs:domain ows:Feature;
       rdfs:isDefinedBy <http://www.opengis.net/ogc/ows/ows-core-
ontology/> ;
       rdfs:label "hasGeometry" ;
       rdfs:range ows:Geometry .

ows:hadProperty a owl:ObjectProperty ;
       rdfs:comment "A Non geometric property of a resource" ;
       rdfs:domain ows:Feature;
       rdfs:isDefinedBy <http://www.opengis.net/ogc/ows/ows-core-
ontology/> ;
       rdfs:label "hasProperty" ;
       rdfs:range ows:Property .

# WPSExectution (a execution of an individual process in a WPS service)
as a subclass of PROV Activity    .
ows:WPSExecution rdfs:subclassOf prov:Activity .

# A WPS parameter that is not a feature or a feature collection
ows:WPSParameter  rdfs:subclassOf prov:Entity .


# Generic Roles
ows:ProcessInput rdfs:subclassOf prov:Role .
ows:ProcessOutput rdfs:subclassOf prov:Role .
ows:Client rdfs:subclassOf prov:Role .
ows:Developer rdfs:subclassOf prov:Role .

# A WPS service is a plan collection
ows:WPSService  rdfs:subclassOf prov:Plan .

# A WPS process is a plan (the capability of doing something)
ows:WPSProcess  rdfs:subclassOf prov:Plan .
```

B.2.2 WPS conflation profile encoding

1_Prov_Conflation_Profile.n3 file content. See Section 7.1 for a description.

```
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .

@prefix ows: <http://www.opengis.net/ogc/ows/ows-core-ontology/> .
@prefix ows10: <http://www.opengis.net/ogc/ows10/ows10-core-ontology/>
.


#################################################################
```

```
#                 WPS generic Executions in ows10
#####################################################################

# This is intruducing the idea of "WPS profiles"
ows10:WPSConflationExecution rdfs:SubclassOf ows:WPSExecution .

#####################################################################
#                     Roles in ows10
#####################################################################

ows10:ConflationProcessInput rdfs:subclassOf ows:ProcessInput .
ows10:ConflationProcessOutput rdfs:subclassOf ows:ProcessInput .

ows10:referenceMapSource     a ows10:ConflationProcessInput .
ows10:crowdsourcedMapSource  a ows10:ConflationProcessInput .
ows10:conflationThreshold    a ows10:ConflationProcessInput .
ows10:conflatedMapOutput     a ows10:ConflationProcessOutput .

#####################################################################
#                     The Process Service and Operation (Plan)
#####################################################################

# The conflation process implements an algorithm, defined as a subclass
of PROV Plan

ows10:ConflationAlgorithm  rdfs:subclassOf ows:WPSProcess.
```


Annex C

Conflation example in RDF

C.1 General

The RDF encoding has been used to express provenance of the geospatial objects, roles and relations. This Annex contains all the RDF encoding derived from the conflation process example. The C.2.1, C.2.2, C.2.3 and C.2.4 Subsections have, respectively, the complete RDF encoding of the 7.2, 7.3, 7.4 and 7.5 sections of the main document.

C.2 RDF encoding

C.2.1 52North conflation process encoded in RDF

2_52N_Conflation_WPS.n3 file content. See Section 7.2 for a description.

```
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .

@prefix ows: <http://www.opengis.net/ogc/ows/ows-core-ontology/> .
@prefix ows10: <http://www.opengis.net/ogc/ows10/ows10-core-ontology/>
.
@prefix f2n: <http://www.opengis.net/ogc/ows10/ows10-f2n-ontology/> .
@prefix wenwen: <http://www.opengis.net/ogc/ows10/ows10-wenwen-
ontology/> .


#####################################################################
#                 WPS generic Executions in ows10
#####################################################################
# 52N_WPSConflationExecution (the implementation of 52North) as a
subclass of ows10:WPSConflationExecution
f2n:52N_WPSConflationExecution rdfs:subclassOf
ows10:WPSConflationExecution .

wenwen:WenWen_WPSConflationExecution rdfs:subclassOf
ows10:WPSConflationExecution .

#####################################################################
#                      People from 52 north
#####################################################################

# 52north and its affiliates (benjamin) are agents associated with the
52N conflation execution
f2n:52north a prov:Agent ;
            a prov:Organization ;
```

```
             foaf:name "52°North Initiative for Geospatial Open Source
Software GmbH" .


# f2n:benjamin is the developer
f2n:benjamin a prov:Agent ;
            a prov:Person ;
            foaf:givenName "Benjamin Pross"^^xsd:string ;
            foaf:mbox      <mailto:b.pross@52north.org> .

f2n:benjamin prov:actedOnBehalfOf f2n:52north .

#Role of Benjamin as developer
f2n:benjaminDeveloper       a ows:Developer .

#####################################################################
#                    The Process Service and Operation (Plan)
#####################################################################

f2n:52N_WPSService  a ows:WPSService;
            a prov:collection;
          owl:sameAs
<http://ows.dev.52north.org:8080/wps/WebProcessingService?service=wps&r
equest=getcapabilities&version=1.0.0> .

# 52N uses a particular conflation algorithm
f2n:52N_ConflationAlgorithm rdfs:subclassOf ows10:ConflationAlgorithm .

f2n:52N_ConflationAlgorithm_v1 a f2n:52N_ConflationAlgorithm ;
             prov:generatedAtTime "2014-03-05T12:00:00"^^xsd:dateTime ;
             owl:sameAs
<http://ows.dev.52north.org:8080/wps/WebProcessingService?service=wps&v
ersion=1.0.0&request=DescribeProcess&Identifier=Conflation> .

f2n:52N_WPSService prov:hadMember f2n:52N_ConflationAlgorithm_v1 .

# The 52N conflation algorithm can have different versions over time,
each defined as an instance of the algorithm and therefore as an
instance of PROV Plan

#####################################################################
#              Attribution the process (who did it?)
#####################################################################

f2n:52N_ConflationAlgorithm_v1 prov:wasAttributedTo f2n:benjamin .

f2n:52N_ConflationAlgorithm_v1 prov:qualifiedAttribution [
                    a  prov:Attribution ;
                       prov:agent    f2n:benjamin ;
                       prov:hadRole  f2n:benjaminDeveloper ;
  ] .
```


C.2.2 The user that executes the process steps encoded in RDF

3_User.n3 file content. See Section 7.3 for a description.

```
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .

@prefix ows: <http://www.opengis.net/ogc/ows/ows-core-ontology/> .
# <http://www.opengis.net/ogc/usgs_conf-dataset/> is a generic URL.
Please use a URL that represents the OSM dataset instead.
@prefix f2n: <http://www.opengis.net/ogc/ows10/ows10-f2n-ontology/> .
@prefix nga: <http://www.opengis.net/ogc/ows10/ows10-nga-ontology/> .

###################################################################
#                     People from NGA
###################################################################

# NGA and its affiliates (david) are agents associated with the
execution of a 52N conflation processes and the creation of conflated
map

nga:NGA a prov:Agent ;
              a prov:Organization ;
              foaf:name "National Geospatial Agency OWS-10
contribution" .

# f2n:david is the operator of the process execution
nga:david a prov:Agent ;
            a prov:Person ;
              foaf:givenName "David Weslow"^^xsd:string ;
              foaf:mbox      <mailto:david.weslow@nga.mil> .

nga:david prov:actedOnBehalfOf nga:NGA .
```

C.2.3 Dataset level provenance encoded in RDF

4_Conflated_Map.n3 file content. See Section 7.4 for a description.

```
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .

@prefix ows: <http://www.opengis.net/ogc/ows/ows-core-ontology/> .
# Dataset namespaces
# <http://www.opengis.net/ogc/ows10/usgs_dataset/> is a generic URL.
Please use a URL that represents the USGS dataset instead.
@prefix usgs: <http://www.opengis.net/ogc/ows10/usgs_dataset/> .
# <http://www.opengis.net/ogc/ows10/osm_dataset/> is a generic URL.
Please use a URL that represents the OSM dataset instead.
```

```
@prefix osm: <http://www.opengis.net/ogc/ows10/osm_dataset/> .
# <http://www.opengis.net/ogc/usgs_conf-dataset/> is a generic URL.
Please use a URL that represents the OSM dataset instead.
@prefix usgs_conf: <http://www.opengis.net/ogc/usgs_conf-dataset/> .

@prefix ows10: <http://www.opengis.net/ogc/ows10/ows10-core-ontology/>
.
@prefix nga: <http://www.opengis.net/ogc/ows10/ows10-nga-ontology/> .

#####################################################################
#     FeatureCollections ("Datasets") involved in the conflation
process
#####################################################################

# The two OWS-10 data sources defined as instances of PROV Collection

#owl:sameAs is decribed here: <http://xmlns.com/foaf/spec/>

usgs:USGSMap a ows:FeatureCollection;
        owl:sameAs
<http://portal.cubewerx.com/cubewerx/projects/ows9/cubeserv.cgi?service
=WFS&amp;datastore=OWS9&amp;request=GetFeature&amp;typename=usgs:fireSt
ationEmsStation>.
osm:OSMMap a ows:FeatureCollection;
        owl:sameAs <http://services.interactive-
instruments.de/xsprojects/ows10/service/tds-dgiwg/wfs/...>.

# The conflated map is a class, subclass of PROV Collection
usgs_conf:conflatedMap a ows:FeatureCollection .

# A new conflated map instance is a revision of a previously generated
one
usgs_conf:conflatedMap prov:wasRevisionOf usgs:USGSMap .

usgs_conf:conflatedMap prov:generatedAtTime "2014-03-
05T10:20:05"^^xsd:dateTime .

#####################################################################
#    Attribution the execution (who pressed the button it?)
#####################################################################
usgs_conf:conflatedMap prov:wasAttributedTo nga:david .

#####################################################################
#     FeatureTypes involved in the conflation process
#####################################################################
usgs:USGS_Feature rdfs:subClassOf ows:Feature .
osm:USGS_Feature rdfs:subClassOf ows:Feature .

#####################################################################
#     AttributeTypes involved in the conflation process
#####################################################################
usgs:USGS_Position rdfs:subClassOf ows:Line .
osm:OSM_Position rdfs:subClassOf ows:Line .
usgs:USGS_Address rdfs:subClassOf ows:Property .
osm:OSM_Address rdfs:subClassOf ows:Property .
```



```
#####################################################################
#                Other parameters used in the execution
#####################################################################

ows10:Distance rdfs:subClassOf ows:WPSParameter .

#####################################################################
#     How Individual conflated feature NON geometrical properties
relate to sources
#####################################################################

# This are just examples of diferent posibilities that NON geometrical
properties can be inhereted or created

# If the NON geometrical property was not affected by the conflation
execution
usgs_conf:conflated_Address1 owl:sameAs usgs:USGS_Address1 .

# If the NON geometrical property was revised by the conflation
execution
usgs_conf:conflated_Address2 prov:wasRevisionOf usgs:USGS_Address2 .

# If the NON geometrical property was created from a OSM feature
usgs_conf:conflated_Address2 prov:wasDerivedFrom osm:OSM_Address2 .
#... 3, 4, 5...
```

C.2.4 Feature and attribute level provenance encoded in RDF

5_Conflation_Step.n3 file content. See Section 7.5 for a description.

```
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .

@prefix ows: <http://www.opengis.net/ogc/ows/ows-core-ontology/> .
# Dataset namespaces
# <http://www.opengis.net/ogc/ows10/usgs_dataset/> is a generic URL.
Please use a URL that represents

the USGS dataset instead.
@prefix usgs: <http://www.opengis.net/ogc/ows10/usgs_dataset/> .
# <http://www.opengis.net/ogc/ows10/osm_dataset/> is a generic URL.
Please use a URL that represents

the OSM dataset instead.
@prefix osm: <http://www.opengis.net/ogc/ows10/osm_dataset/> .
# <http://www.opengis.net/ogc/usgs_conf-dataset/> is a generic URL.
Please use a URL that represents

the OSM dataset instead.
@prefix usgs_conf: <http://www.opengis.net/ogc/usgs_conf-dataset/> .
```



```
@prefix ows10: <http://www.opengis.net/ogc/ows10/ows10-core-ontology/>
.
@prefix f2n: <http://www.opengis.net/ogc/ows10/ows10-f2n-ontology/> .
@prefix wenwen: <http://www.opengis.net/ogc/ows10/ows10-wenwen-
ontology/> .
@prefix nga: <http://www.opengis.net/ogc/ows10/ows10-nga-ontology/> .


#####################################################################
#     Individual feature properties of the source "datasets"
#####################################################################

usgs:USGS_Position1 a usgs:USGS_Position .
usgs:USGS_Position2 a usgs:USGS_Position .
usgs:USGS_Address1 a usgs:USGS_Address .
usgs:USGS_Address2 a usgs:USGS_Address .
usgs:OSM_Position1 a osm:OSM_Position .
usgs:OSM_Position2 a osm:OSM_Position .
osm:OSM_Address1 a osm:OSM_Address .
osm:OSM_Address2 a osm:OSM_Address .
usgs_conf:conflated_Position1 a usgs:USGS_Position .
usgs_conf:conflated_Position2 a usgs:USGS_Position .
usgs_conf:conflated_Address1 a usgs:USGS_Address .
usgs_conf:conflted_Address2 a usgs:USGS_Address .

#####################################################################
#     Individual features of the source "datasets"
#####################################################################

# USGS_Feature1, ... are just generic names. Please use the gml:id
values as names usgs namespace

usgs:USGS_Feature1 a usgs:USGS_Feature;
                      ows:hadGeometry usgs:USGS_Position1;
                   ows:hadProperty usgs:USGS_Address1 .
usgs:USGS_Feature2 a usgs:USGS_Feature;
                   ows:hadGeometry usgs:USGS_Position2;
                      ows:hadProperty usgs:USGS_Address2 .
# ...3, 4, 5...

# OSM_Feature1, ... are just generic names. Please use the gml:id
values as names usgs namespace
osm:OSM_Feature1 a osm:OSM_Feature;
                    ows:hadGeometry usgs:OSM_Position1;
                    ows:hadProperty osm:OSM_Address1 .
osm:OSM_Feature2 a osm:OSM_Feature;
                      ows:hadGeometry osm:OSM_Position2;
             ows:hadProperty osm:OSM_Address2 .
# ...3, 4, 5...

# The two OWS-10 data sources are formed by member entities
# http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-membership
usgs:USGSMap prov:hadMember usgs:USGS_Feature1;
              prov:hadMember usgs:USGS_Feature2 .
```



```
# ... 3, 4, 5...

osm:OSMMap   prov:hadMember osm:OSM_Feature1;
             prov:hadMember osm:OSM_Feature2 .
# ... 3, 4, 5...

# Please note that the confation results are defined as
usgs:USGS_Feature type deliverately.
# this because the conflate results has the same data structure than
the USGS dataset.
usgs_conf:conflated_Feature1 a usgs:USGS_Feature;
                     ows:hadGeometry usgs_conf:conflated_Position1;
                   ows:hadProperty usgs_conf:conflated_Address1 .
usgs_conf:conflated_Feature2 a usgs:USGS_Feature;
                     ows:hadGeometry usgs_conf:conflated_Position2;
                   ows:hadProperty usgs_conf:conflated_Address2 .
# ... 3, 4, 5...

usgs_conf:conflatedMap prov:hadMember usgs_conf:conflated_Feature1;
                    prov:hadMember usgs_conf:conflated_Feature2 .
# ... 3, 4, 5...

#####################################################################
#     How Individual conflated features relate to sources
#####################################################################

# This are just examples of diferent posibilities that Features can be
inhereted or created

# If the feature was not affected by the conflation execution
usgs_conf:conflated_Feature1 owl:sameAs usgs:USGS_Feature1 .

# If the feature was revised by the conflation execution
usgs_conf:conflated_Feature2 prov:wasRevisionOf usgs:USGS_Feature2 .

# If the feature was created from a OSM feature
usgs_conf:conflated_Feature2 prov:wasDerivedFrom osm:OSM_Feature2 .
#... 3, 4, 5...

#####################################################################
#     How Individual conflated feature geometries relate to sources
#####################################################################

# This are just examples of diferent posibilities that Geometries can
be inhereted or created

# If the geometry was not affected by the conflation execution
usgs_conf:conflated_Position1 owl:sameAs usgs:USGS_Geometry1 .

# If the geometry was revised by the conflation execution
usgs_conf:conflated_Position2 prov:wasRevisionOf usgs:USGS_Geometry2 .

# If the geometry was created from a OSM feature
usgs_conf:conflated_Position2 prov:wasDerivedFrom osm:OSM_Geometry2 .
#... 3, 4, 5...
```

```
#####################################################################
#                Other parameters used in the execution
#####################################################################

# The threshold distance is an example of parameter.
ows10:distance1 a ows10:Distance;
       prov:value 10 .

f2n:52N_ConflationExecution20140305 prov:used ows10:distance1 .

f2n:52N_ConflationExecution20140305 prov:qualifiedUsage [
                  a prov:Usage ;
                  prov:entity  ows10:distance1 ;
                  prov:hadRole ows10:conflationThreshold
     ] .


#####################################################################
#                       Individual execution
#####################################################################

## Operation 1 runs conflation processes instance number 20140305 (we
use the date as an identifier).
f2n:52N_ConflationExecution20140305 a f2n:52N_WPSConflationExecution .

# The individual execution uses a plan
f2n:52N_ConflationExecution20140305  prov:used
f2n:52N_ConflationAlgorithm_v1 .

f2n:52N_ConflationExecution20140305 prov:qualifiedUsage [
                  a prov:Usage ;
                  prov:hadPlan  f2n:52N_ConflationAlgorithm_v1 ;
                  prov:hadRole nga:client
     ] .

f2n:52N_ConflationExecution20140305  prov:startedAtTime "2014-03-
03T08:10:00"^^xsd:dateTime ;
               prov:endedAtTime   "2014-03-05T10:20:00"^^xsd:dateTime .

#####################################################################
#    Relations between individual features and individual executions
#####################################################################

# Usage and Generation relations between entities and activities
# Each conflation process instance uses the two OWS-10 data sources
f2n:52N_ConflationExecution20140305   prov:used  usgs:USGS_Feature1;
                                      prov:used  usgs:USGS_Feature2;
                                      prov:used  osm:OSM_Feature1;
                                      prov:used  osm:OSM_Feature2 .

# When a conflation process is run, a conflated map instance is
generated by the process
usgs_conf:conflated_Feature1  prov:wasGeneratedBy
f2n:52N_ConflationExecution20140305 .
```



```
usgs_conf:conflated_Feature2  prov:wasGeneratedBy
f2n:52N_ConflationExecution20140305 .
#...

# The provenance of conflated maps and conflation processes can include
time stamps
usgs_conf:conflated_Feature1 prov:generatedAtTime "2014-03-
05T10:20:00"^^xsd:dateTime .
usgs_conf:conflated_Feature2 prov:generatedAtTime "2014-03-
05T10:20:05"^^xsd:dateTime .
#...


#####################################################################
#    Attribution the execution (who pressed the button it?)
#####################################################################

usgs_conf:conflated_Feature1 prov:wasAttributedTo nga:david .
usgs_conf:conflated_Feature2 prov:wasAttributedTo nga:david .
#... 3, 4, 5...

f2n:52N_ConflationExecution20140305 prov:wasAssociatedWith nga:david .

f2n:52N_ConflationExecution20140305 prov:qualifiedAssociation [
        a prov:Association;
        prov:agent   nga:david;
        prov:hadRole nga:client
      ] .



#####################################################################
#    Roles for individual executions and features.
#####################################################################
# Entities and agents have specific roles in conflation processes


f2n:52N_ConflationExecution20140305 prov:qualifiedUsage [
                  a prov:Usage ;
                  prov:entity  usgs:USGS_Feature1 ;
                  prov:hadRole ows10:ReferencedMapSource
     ] .

f2n:52N_ConflationExecution20140305 prov:qualifiedUsage [
                  a prov:Usage ;
                  prov:entity  osm:OSM_Feature1 ;
                  prov:hadRole ows10:crowdsourcedMapSource
     ] .

f2n:52N_ConflationExecution20140305 prov:qualifiedUsage [
                  a prov:Usage ;
                  prov:entity  usgs:USGS_Feature2 ;
                  prov:hadRole ows10:ReferencedMapSource
     ] .
f2n:52N_ConflationExecution20140305 prov:qualifiedUsage [
                  a prov:Usage ;
```

```
                  prov:entity  osm:OSM_Feature2 ;
                  prov:hadRole ows10:crowdsourcedMapSource
     ] .

#... 3, 4, 5...

usgs_conf:conflated_Feature1 prov:qualifiedGeneration [
                     a prov:Generation ;
                     prov:activity  f2n:52N_ConflationExecution20140305
;
                     prov:hadRole   ows10:conflatedMapOutput
  ] .

usgs_conf:conflated_Feature2 prov:qualifiedGeneration [
                     a prov:Generation ;
                     prov:activity  f2n:52N_ConflationExecution20140305
;
                     prov:hadRole   ows10:conflatedMapOutput
  ] .

#... 3, 4, 5...


###################################################################
#                              Bundle
###################################################################


# The provenance of the provenance can also be expressed by defining a
bundle at the beginning of a

file as follows:

prov:Bundle prov:wasAttributedTo f2n:Arne ;
            prov:wasGeneratedAt "2014-03-05T10:22:00"^^xsd:dateTime .
```


Annex D

ISO lineage for feature and attribute level examples

D.1 General

This annex is an example of provenance encoded in ISO 19115 and ISO 19139. Concretely, the document describes a dataset derived from a conflation process of a U.S. Geological Survey map and Open Street Map.

D.2 XML Encoding

D.2.1 Example of provenance in ISO 19115 and ISO 19139

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- <?xml-stylesheet type="text/xsl"
href="http://schemas.geoviqua.org/GVQ/3.0.0/stylesheets/GeoViQuaCompone
nts-HTMLTable.xsl"?>  -->
<gmd:MD_Metadata xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gmd="http://www.isotc211.org/2005/gmd"
xmlns:gco="http://www.isotc211.org/2005/gco"
xmlns:gml="http://www.opengis.net/gml/3.2"
xsi:schemaLocation="http://www.isotc211.org/2005/gmd
../../../iso/19139/20070417/gmd/gmd.xsd" id="OWS-10Conflatedmap">
    <gmd:fileIdentifier>
        <gco:CharacterString>OWS-10Conflatedmap</gco:CharacterString>
    </gmd:fileIdentifier>
    <gmd:language>
        <gco:CharacterString>eng</gco:CharacterString>
    </gmd:language>
    <gmd:characterSet>
        <gmd:MD_CharacterSetCode
codeList="http://www.isotc211.org/2005/resources/codeList.xml#MD_Charac
terSetCode" codeListValue="8859part1"/>
    </gmd:characterSet>
    <gmd:contact>
        <gmd:CI_ResponsibleParty id="CREAF">
            <gmd:organisationName>
                <gco:CharacterString>Centre for Ecological Research and
Forestry Applications</gco:CharacterString>
            </gmd:organisationName>
            <gmd:positionName>
                <gco:CharacterString>Researcher</gco:CharacterString>
            </gmd:positionName>
            <gmd:contactInfo>
                <gmd:CI_Contact>
                </gmd:CI_Contact>
            </gmd:contactInfo>
            <gmd:role>
```

```
                <gmd:CI_RoleCode
codeList="http://www.isotc211.org/2005/resources/codeList.xml#CI_RoleCo
de" codeListValue="author"/>
            </gmd:role>
        </gmd:CI_ResponsibleParty>
    </gmd:contact>
    <gmd:dateStamp>
        <gco:Date>2013-06-14</gco:Date>
    </gmd:dateStamp>
    <gmd:metadataStandardName>
        <gco:CharacterString>ISO 19115:2003/19139</gco:CharacterString>
    </gmd:metadataStandardName>
    <gmd:metadataStandardVersion>
        <gco:CharacterString>1.0</gco:CharacterString>
    </gmd:metadataStandardVersion>
    <gmd:spatialRepresentationInfo>
    </gmd:spatialRepresentationInfo>
    <gmd:referenceSystemInfo>
        <gmd:MD_ReferenceSystem>
            <gmd:referenceSystemIdentifier>
                <gmd:RS_Identifier>
                    <gmd:code>

<gco:CharacterString>EPSG:23031</gco:CharacterString>
                    </gmd:code>
                </gmd:RS_Identifier>
            </gmd:referenceSystemIdentifier>
        </gmd:MD_ReferenceSystem>
    </gmd:referenceSystemInfo>
    <gmd:identificationInfo>
    </gmd:identificationInfo>
    <gmd:dataQualityInfo>
        <gmd:DQ_DataQuality>
            <gmd:scope>
                <gmd:DQ_Scope>
                    <gmd:level>
                        <gmd:MD_ScopeCode
codeList="http://www.isotc211.org/2005/resources/codeList.xml#MD_ScopeC
ode" codeListValue="dataset">dataset</gmd:MD_ScopeCode>
                    </gmd:level>
                </gmd:DQ_Scope>
            </gmd:scope>
            <gmd:lineage>
                <gmd:LI_Lineage>
                    <gmd:statement>
                        <gco:CharacterString>The dataset is a result of
a conflation WPS instance between an USGS Map Data and Open Street
Map</gco:CharacterString>
                    </gmd:statement>
                    <gmd:processStep>
                        <gmd:LI_ProcessStep id="PS_WPS_1">
                            <gmd:description>
                                <gco:CharacterString>52North WPS
conflation instance</gco:CharacterString>
                            </gmd:description>
```



```
                            <gmd:dateTime><gco:DateTime>2013-11-
22T16:05:00</gco:DateTime></gmd:dateTime>
                            <gmd:processor>
                                <gmd:CI_ResponsibleParty
id="JM_CREAF">
                                    <gmd:individualName>
                                        <gco:CharacterString>Joan
Masó</gco:CharacterString>
                                    </gmd:individualName>
                                    <gmd:organisationName>

<gco:CharacterString>CREAF</gco:CharacterString>
                                    </gmd:organisationName>
                                    <gmd:positionName>

<gco:CharacterString>Researcher</gco:CharacterString>
                                    </gmd:positionName>
                                    <gmd:role>
                                        <gmd:CI_RoleCode
codeList="http://www.ngdc.noaa.gov/metadata/published/xsd/schema/resour
ces/Codelist/gmxCodelists.xml#CI_RoleCode"
codeListValue="processor">processor</gmd:CI_RoleCode>
                                    </gmd:role>
                                </gmd:CI_ResponsibleParty>
                            </gmd:processor>
                            <gmd:source xlink:href="#SC_USGS" />
                            <gmd:source xlink:href="#SC_OSM" />
                        </gmd:LI_ProcessStep>
                    </gmd:processStep>
                    <gmd:source>
                        <gmd:LI_Source id="SC_USGS">
                            <gmd:description>
                                <gco:CharacterString>The U.S.Geological
Survey generates the Geological Soils Maps of United
States</gco:CharacterString>
                            </gmd:description>
                            <gmd:sourceCitation>
                                <gmd:CI_Citation>
                                    <gmd:title>
                                        <gco:CharacterString>USGS map
data</gco:CharacterString>
                                    </gmd:title>
                                    <gmd:date/>
                                </gmd:CI_Citation>
                            </gmd:sourceCitation>
                            <gmd:sourceStep xlink:href="#PS_WPS"/>
                        </gmd:LI_Source>
                    </gmd:source>
                    <gmd:source>
                        <gmd:LI_Source id="SC_OSM">
                            <gmd:description>
                                <gco:CharacterString>OpenStreetMap
(OSM) is a collaborative project to create a free editable map of the
world.</gco:CharacterString>
                            </gmd:description>
                            <gmd:sourceCitation>
```



```
                                    <gmd:CI_Citation>
                                        <gmd:title>
                                            <gco:CharacterString>Open
Street Map</gco:CharacterString>
                                        </gmd:title>
                                        <gmd:date/>
                                    </gmd:CI_Citation>
                                </gmd:sourceCitation>
                                <gmd:sourceStep xlink:href="#PS_WPS"/>
                            </gmd:LI_Source>
                        </gmd:source>
                    </gmd:LI_Lineage>
                </gmd:lineage>
            </gmd:DQ_DataQuality>
        </gmd:dataQualityInfo>
</gmd:MD_Metadata>
```


Annex E

Tools used in the Engineering Report

E.1 General

In order to elaborate and validate all the RDF and XML schemas and examples in the Interoperability Experiment that are presented in this engineering report, some editor and validation tools have been used. This information is presented here as an informative reference.

E.2 Altova XMLSpy

XMLSpy is an XML editor and integrated development environment (IDE) developed by Altova. XMLSpy that allows developers to create XML schemas and documents commonly used in OGC and ISO as a validation tool in the Windows operating system

The XMLSpy was used during the elaboration and validation of XML examples encoded in ISO 19139 and W3C PROV XML.

E.3 EulerGUI

The EulerGUI Integrated Development Environment (IDE) is a java application that allows developing and validating documents written in N3 logic language. The sources can be N3, RDF, or OWL, SPARQL queries, UML, eCore, plain XML, XML Schema, files or URL's.

The EulerGUI was used to edit and validate the W3C PROV RDF.

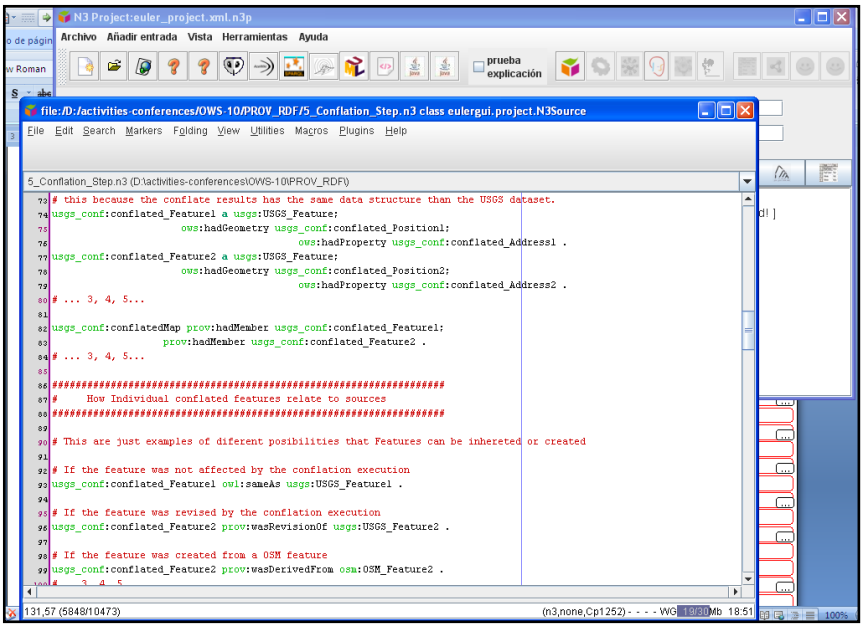

Figure E.1 — Example of EulerGUI use

E.4 Graphviz

Graphviz is a graphical editor that is integrated in the EulerGUI application. This way EulerGUI can use the Dotty graph viewer for showing and RDF document as a graph.. Some figures in Section 7 have been done with this tool.

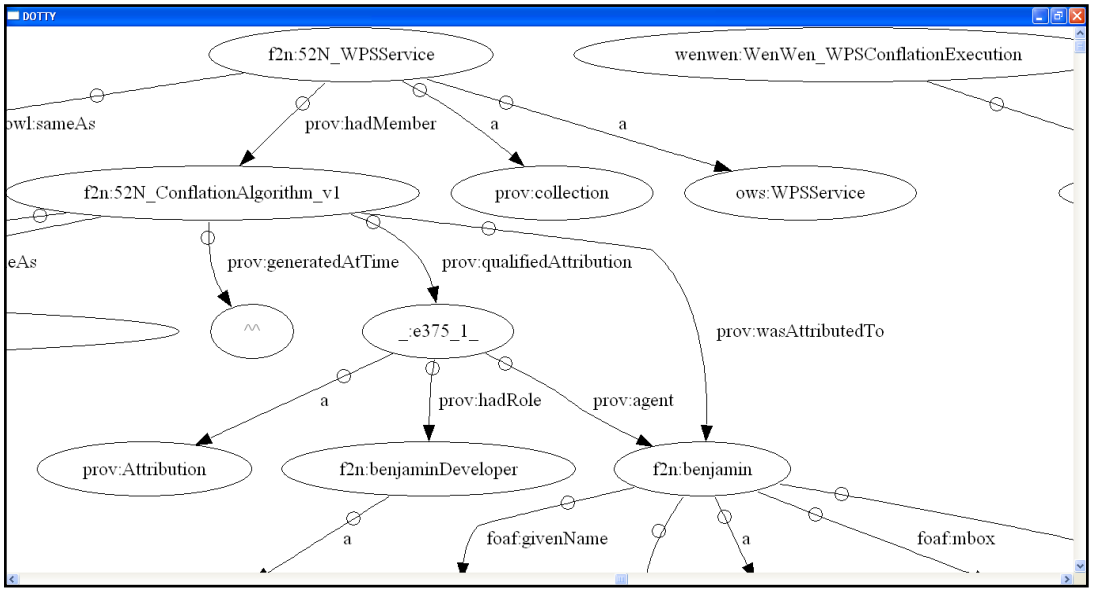

Figure E.2 — Example of Dotty

Bibliography

In addition to the reference documentation cited in Section 2, some other materials have been cited throughout the document.

[OGC 01-101] OGC® Abstract Specification Topic 1: Feature Geometry (ISO 19107 Spatial Schema)

[OGC 02-112] OGC® Abstract Specification Topic 12 OGC Service Architecture Version 4.3

[OGC 05-007r7] Web Processing Service (WPS 1.0.0)

[OGC 06-004r4] OGC® Abstract Specification Topic 18: Geospatial Digital Rights Management Reference Model GeoDRM RM

[OGC 10-059r2] OWS-7 WPS Profiling Engineering Report

[OGC 12-159] OWS-9 CCI Conflation with Provenance Engineering Report