Open Geospatial Consortium

Approval Date: 2013-01-18

Posted Date: 2012-12-21

Publication Date: 2013-06-18

Reference number of this document: OGC 12-105

Reference URL for this document: http://www.opengisl.net/def/doc-type/per/ows9-ows-context

Category: Engineering Report

Editor: Joan Masó

OGC® OWS-9 - OWS Context evaluation IP Engineering Report

Warning

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements.

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

Abstract

This OGC Engineering Report describes the results of the OWS-9 IP on OWS Context 1.0. OWS Context is a draft OGC candidate standard. The OWS Context activity tested and evaluated the relative benefits of different encoding methods prior to finalization of the candidate standard. OWS Context has been proposed with an Atom encoding, a JSON encoding and an HTML5 encoding. The encoding requirement seeks to understand the level of mass-market acceptance of these different encoding options and their ability to support mash-ups. Each encoding should be evaluated, including examples and recommendations to move forward. Recommendations should enable the OWS Context capability for OGC services while remaining cognizant of implementations using mass-market technologies.

Keywords

ogcdoc, OGC document, context, ows context, json, atom, wmc, wmts

What is OGC Web Services 9 (OWS-9)?

OWS-9 builds on the outcomes of prior OGC interoperability initiatives and is organized around the following threads:

- **Aviation**: Develop and demonstrate the use of the Aeronautical Information Exchange Model (AIXM) and the Weather Exchange Model (WXXM) in an OGC Web Services environment, focusing on support for several Single European Sky ATM Research (SESAR) project requirements as well as FAA (US Federal Aviation Administration) Aeronautical Information Management (AIM) and Aircraft Access to SWIM (System Wide Information Management) (AAtS) requirements.

- **Cross-Community Interoperability (CCI)**: Build on the CCI work accomplished in OWS–8 by increasing interoperability within communities sharing geospatial data, focusing on semantic mediation, query results delivery, data provenance and quality and Single Point of Entry Global Gazetteer.

- **Security and Services Interoperability (SSI)**: Investigate 5 main activities: Security Management, OGC Geography Markup Language (GML) Encoding Standard Application Schema UGAS (UML to GML Application Schema) Updates, Web Services Façade, Reference Architecture Profiling, and Bulk Data Transfer.

- **OWS Innovations**: Explore topics that represent either new areas of work for the Consortium (such as GPS and Mobile Applications), a desire for new approaches to existing technologies to solve new challenges (such as the OGC Web Coverage Service (WCS) work), or some combination of the two.

- **Compliance & Interoperability Testing & Evaluation (CITE)**: Develop a suite of compliance test scripts for testing and validation of products with interfaces

implementing the following OGC standards: Web Map Service (WMS) 1.3 Interface Standard, Web Feature Service (WFS) 2.0 Interface Standard, Geography Markup Language (GML) 3.2.1 Encoding Standard, OWS Context 1.0 (candidate encoding standard), Sensor Web Enablement (SWE) standards, Web Coverage Service for Earth Observation (WCS-EO) 1.0 Interface Standard, and TEAM (Test, Evaluation, And Measurement) Engine Capabilities.

The OWS-9 sponsors are: AGC (Army Geospatial Center, US Army Corps of Engineers), CREAF-GeoViQua-EC, EUROCONTROL, FAA (US Federal Aviation Administration), GeoConnections - Natural Resources Canada, Lockheed Martin Corporation, NASA (US National Aeronautics and Space Administration), NGA (US National Geospatial-Intelligence Agency), USGS (US Geological Survey), UK DSTL (UK MoD Defence Science and Technology Laboratory).

License Agreement

Contents

Page

Figures

Page

Tables

Page

OGC® OWS-9 - OWS Context evaluation IP Engineering Report

1 Introduction

1.1 Scope

This OGC® document gives guidelines for using different encodings for OWS Context 1.0. It also summarizes lessons learned in the OGC OWS-9 when working with an OWS Context document. It covers 2 encodings i.e. ATOM and JSON, and it discusses how OWS Context can be used in an HTML5 application.

This OGC® document is applicable to clients and servers that are capable of reading or writing an OWS Context.

1.2 Document contributor contact points

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

Name	Organization
Joan Masó	CREAF

1.3 Revision history

Date	Release	Editor	Primary clauses modified	Description
2012-08-25	1	Joan Masó		First draft to present in OWS-9
2013-01-07	2	Joan Masó		First official version presented on pending documents.

1.4 Future work

Clause *10, Future work,* describes the open issues discovered during the elaboration of this ER and during OWS-9 concerning OWS Context.

The main issue is about having a clear recommendation on how to include annotations in OWS Context and fix some schema validation problems that appear when trying to use GML profiles such as GeoRSS simultaneously with WFS.

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.

OGC 12-080 *OGC® OGC OWS Context Standard. Conceptual Model*

OGC 12-084 *OGC® OWS Context Standard. ATOM Encoding*

In addition to this document, this report includes several XML Schema Document files as specified in Annex A.

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], the OGC OWS Context Conceptual Model [OGC 12-080] and the OGC OWS Context ATOM Encoding Specification [OGC 12-084] shall apply. In addition, the following terms and definitions apply.

3.1
annotation
Any marking on illustrative material for the purpose of clarification.
[ISO19117:2005 "Protrayal"]

An association between an annotation entity (e.g. a text label) and an image or some geometric “region” within the image.
[OGC 05-047r3 GMLJP2]

An illustrative feature overlapped to a map and anchored to a geospatial feature, a part of it or a spatial region for the purpose of clarification.

3.2 nominal scale denominator

Ratio between the size of a feature in CRS coordinates and the size of a feature in a screen that has a pixel size of 0.28mm

4 Conventions

4.1 Abbreviated terms

Some frequently used abbreviated terms:

JSON: JavaScript Object Notation

OWS: Open Geospatial Web Services

OWC: OWS Context

4.2 XML schema location

Many XML fragments are included in this document and sometimes XML schema locations are referenced. For convenience we work with local copies of the schemas instead of pointing to the official OGC Schema repository (http://schemas.opengis.net/) using full URLs (and avoid relative referencing between different namespaces). By doing this we are ignoring an OGC recommendation. The reasons for doing so are that there is no official OWS Context in the repository and that it is then possible to work offline. Please, note that if you plan to copy some of the code included in this document you have to modify the XML schema locations and point it to the official OGC repository.

4.3 Used parts of other documents

This document uses significant parts of document [OGC 12-080]. To reduce the need to refer to that document, this document copies some of those parts with small modifications. To indicate those parts to readers of this document, the largely copied parts are shown with a light grey background (15%).

4.4 Data dictionary tables

The UML model data dictionary is specified herein in a series of tables. The contents of the columns in these tables are described in Table 1.

Table 1 — Contents of data dictionary tables

Column title	Column contents
Names (left column)	Two names for each included parameter or association (or data structure). The first name is the UML model attribute or association role name. The second name uses the XML encoding capitalization specified in Subclause 11.6.2 of [OGC 06-121r3]. The name capitalization rules used are specified in Subclause 11.6.2 of [OGC 06-121r3]. Some names in the tables may appear to contain spaces, but no names contain spaces.
Definition (second column)	Specifies the definition of this parameter (omitting un-necessary words such as "a", "the", and "is"). If the parameter value is the identifier of something, not a description or a definition, the definition of this parameter should read something like "Identifier of TBD".
Data type and value (third column) or Data type (if are no second items are included in rows of table)	Normally contains two items: The mandatory first item is often the data type used for this parameter, using data types appropriate in a UML model, in which this parameter is a named attribute of a UML class. Alternately, the first item can identify the data structure (or class) referenced by this association, and references a separate table used to specify the contents of that class (or data structure). The optional second item in the third column of each table should indicate the source of values for this parameter, the alternative values, or other value information, unless the values are quite clear from other listed information.
Multiplicity and use (right or fourth column) or Multiplicity (if are no second items are included in rows of table)	Normally contains two items: The mandatory first item specifies the multiplicity and optionality of this parameter in this data structure, either "One (mandatory)", "One or more (mandatory)", "Zero or one (optional)", or "Zero or more (optional)". The second item in the right column of each table should specify how any multiplicity other than "One (mandatory)" shall be used. If that parameter is optional, under what condition(s) shall that parameter be included or not included? If that parameter can be repeated, for what is that parameter repeated?

When the data type used for this parameter, in the third column of such a table, is an enumeration or code list, all the values specified shall be listed, together with the meaning of each value. When this information is extensive, these values and meanings should be specified in a separate table that is referenced in the third column of this table row.

The data type of many parameters, in the third table column, is specified as "Character String type, not empty". In the XML Schema Documents specified herein, these parameters are encoded with the xsd:string type, which does NOT require that these strings not be empty.

5 OWS context evaluation ER overview

The specified OWS context evaluation ER addresses the OWS Context draft standard and studies possible encodings, validation techniques and practical implementations tested in OWS 9.

In 2005 OGC released a standard for sharing map compositions that linked to WMS maps called Web Map Context (WMC). This standard defines an XML encoding for saving the status (the context) of a WMS client, which later can be used to recover the saved status. It is able to save WMS URL entry points, the parameters needed to obtain each layer on a map composition as well as the size of the screen in pixels among other metadata. The standard also suggested the possibility of saving references to these XML files in another file that acts as a very simple catalogue of WMC documents.

Since the first version of the WMS, OGC Web Services (OWS) have been diversified and several service standards are now available for different tasks. Some integrated clients are able to dialog with some of these services and to present the data to the user. The possibility to extent WMC to other services was noted while elaborating the WMC specification and, in fact, some work was already done by some authors. WMC was combined with Web Processing Services and Map Services, furthermore WMC was extended to support WFS services in a previous OWS interoperability experiment.

Clause 6 of this document analyzes the conceptual model of the new OWS context and summarizes the practical implications of some decisions took by the Standards Working Group. Clause 7 describes the encodings for the OWS Context, paying particular attention to the Atom encoding as a way to include geospatial content in an Atom file. In Clause 8, it can be found the description of Atom encoding implementation in integrated clients. Clause 9 describes the Atom encoding in services. Finally Clause 10 describes future work.

6 Context Model

6.1 WMC Conceptual Model

There is no conceptual model described in the WMC 05-005 but we can easily generate it from the XML encoding description, which is well defined in the WMC standard.

Table 2 — WMC General model

Name	Definition	Data type and value	Multiplicity and use
version	The published specification version number, contains three positive integers, separated by decimal points, in the form “x.y.z”	Character String type, not empty	One (mandatory)

Name	Definition	Data type and value	Multiplicity and use
id	Unique Identifier assigned to the Context document	URI	One (mandatory)
title	A human readable title for the Context document	Character String type, not empty	One (mandatory)
abstract	Description of the Context document purpose or content	Character String type, not empty	Zero or one (optional)
contactInformation	Contact information of the creator of the Context document	ContactInformationType type	Zero or one (optional)
boundingBox	It represents the geographic extent that should be presented by the client	GM_Envelope, it includes the CRS	One (mandatory)
witdh	The size in pixels of the map the Context document describes	integer	Zero or one (optional)
height	The size in pixels of the map the Context document describes	integer	Zero or one (optional)
keyword	Keyword related to this context document. Shall support an optional codelist parameter	Character String type, not empty	Zero or more (optional)
layer	The description of a resource and its access parameters and configuration [d]	Character String type, not empty	One or more (mandatory)
logoURL	A reference to an image that might be attached to the Context document.	URI and image sizes	Zero or more (optional)
descriptionURL	A URL reference to a webpage which contains relevant information to the view.	URI	Zero or more (optional)
extension	Any other element [e]	n/a	Zero or more (optional)

Table 3 – WMC:Layer

Name	Definition	Data type and values	Multiplicity and use

Name	Definition	Data type and values	Multiplicity and use
serverURL	The link to online resource endpoint of the service	URI	One (mandatory)
serviceType	The type of the service according to OGC interfaces	Character String type, not empty. Examples are WMS, WFS.	One (mandatory)
serviceVersion	Version number of the OGC interface specification which corresponds to the service	Character String type, not empty	One (mandatory)
serviceTitle	The title of the service (extracted from the Capabilities by the Context document creator)	Character String type, not empty	Zero or one (optional)
name	The name of the selected layer (extracted from Capabilities by the Context document creator).	Character String type, not empty	One (mandatory)
title	The title of the selected layer (extracted from Capabilities by the Context document creator). This element is required.	Character String type, not empty	One (mandatory)
abstract	The abstract of the selected layer (extracted from Capabilities by the Context document creator). This element is optional.	Character String type, not empty.	One (mandatory)
SRS	Available SRS for the enclosing layer.	Character String type, not empty [a]	Zero or more (optional)
dimension	This element is defined in the Web Map Service specification [WMS]. One more attribute is defined to specify the current Dimension constraint. The attribute “current” defines the Dimension that **must** be used to request this layer.	Dimension type	Zero or more (optional)

Name	Definition	Data type and values	Multiplicity and use
dataURL	A link to an online resource where data corresponding to the layer can be found. This element is optional.	URI	Zero or one (optional)
format	Describe output image formats for the Layer.	Character String type, not empty Format attribute is "current" that contains 1 if the current image format is selected.	One or more (mandatory)
hidden	Indicates if the layer should be hidden in the client result map.	Boolean	One (mandatory)
queryable	Indicates if the layer is set as queryable	Boolean	One (mandatory)
style	Available styles for this layer using named style or SLD	Style type	Zero or more (optional)
minScaleDenominator	The minimum scale from which this layer should be displayed by a client application.	Double	Zero or one (optional)
maxScaleDenominator	The minimum scale from which this layer should be displayed by a client application.	Double	Zero or one (optional)
metadataURL	A link to an online resource where descriptive metadata corresponding to the layer can be found.	URI	Zero or more (optional) [d]
extension	Any other element	n/a	Zero or more (optional)
[a] One of the listed SRS's must be the SRS mentioned in the ViewerContext/General/BoundingBox@SRS element.			

Additionally, the WMC defines a ViewContextCollection that represent a list of context documents. Context collections could be used in several ways:

- A particular Viewer Client could use a Collection to construct a menu of default start-up views.
- A Collection of related contexts could serve as a script for a demonstration.
- A user could create a Collection to "bookmark" public or user-specific contexts. The creation of such a Collection might be managed by the Viewer Client itself.

It has a version number and a list of ViewContextReference each one with its corresponding id, title and contextURL to a context document.

6.2 OWS Context Conceptual Model

This clause discusses the conceptual model for OWS Context as it was at the moment of writing this document. It is included here because it is foreseen that the model may undergo some change in the future, during the later stages of the standards elaboration process in OGC.

Mainly an OWC:Context element is a collection of general metadata for the context document and a list of resources.

Table 4 — OWC:Context

Name	Definition	Data type and value	Multiplicity and use
specReference	Specification Reference (requirements class) identifying that it is an OWC Context document and its version.	URI Value SHALL be "http://www.opengis.net/spec/owc/1.0/req/atom" in this version	One (mandatory)
language	Language used in the OWC Context document	Character String type, not empty	One (mandatory)
id	Unique Identifier assigned to the OWS Context document	URI	One (mandatory)
title	A human readable title for the OWS Context Document	Character String type, not empty	One (mandatory)
abstract	Description of the Context document purpose or content	Character String type, not empty	Zero or one (optional)
updateDate	A date of a creation or update of the Context document	Date type	One (mandatory)
author	An entity primarily responsible for making the Context document	Character String type, not empty	Zero or one (optional)
publisher	Identifier for the publisher of the Context document	Character String type, not empty	Zero or one (optional)
creatorApplication	The tool/application used to create the Context document	Character String type, not empty	Zero or one (optional)
rights	Information about rights held in and over the Context document	Character String type, not empty	Zero or one (optional)

Name	Definition	Data type and value	Multiplicity and use
areaOfInterest	Geographic Area of interest of the users of the Context document [a] [b]	GM_Envelope	Zero or one (optional)
timeIntervalOfInterest	A date or range of dates relevant to the resource	TM_GeometricPrimitive	Zero or one (optional)
keyword	Keyword related to this context document. Shall support an optional codelist parameter	Character String type, not empty	Zero or more (optional)
resource	The description of a resource and its access parameters and configuration [d]	owc:Resource (as defined in Table 5)	Zero or more (optional)
contextMetadata	Additional metadata describing the context document itself. The format recommended is ISO19115 complaint metadata. The metadata standard used should be specified	n/a	Zero or more (optional) [c]
extension	Any other element [e]	n/a	Zero or more (optional)

[a] These properties define the geographic area of interest and date/time interval of interest to the context user. They do not define the bounding extent (either in geographic area or time) of the referenced resources. The intention is not to provide the overall bounds or clipping extent, but simply to indicate the accessor the expected view of the information in area and time.

[b] The Coordinate Reference System shall be unambiguous. It should either be implicitly indicated in the choice of encoding of AOI or explicitly defined.

[c] The rights described apply to the context document itself not to any of its contents.

[d] Resources are ordered. Clients would normally interpret this in terms of display order. How the encoding defines the order of layers in relation to the display shall be defined in the encoding specification.

[e] Any encoding should allow the user to extend the Context content to include custom items.

Mainly a OWC:Resource element is a collection of general metadata for the resource and a list of offerings.

Table 5 — OWC:Resource

Name	Definition	Data type and values	Multiplicity and use
id	Unique Identifier assigned to the OWS Resource. Used to reference to a resource from other resources.	Character String type, not empty	One (mandatory)

Name	Definition	Data type and values	Multiplicity and use
title	A human readable title for the OWC Resource.	Character String type, not empty	One (mandatory)
abstract	Description of the OWC Resource Purpose or Content.	Character String type, not empty.	One (mandatory)
updateDate	Date when the resource definition was last updated.	Date type	One (mandatory)
author	Identifier of the author of the resource definition.	Character String type, not empty	Zero or more (optional) [a]
publisher	Identifier of the publisher of the ows:resource.	Character String type, not empty	Zero or one (optional)
rights	Information about rights held in and over the ows:resource. [a]	Character String type, not empty	Zero or one (optional)
geospatialExtent	The spatial extent or scope of the content of the ows:resource. [b]	GM_Envelope	Zero or one (optional)
temporalExtent	A date or range of dates relevant to the ows:resource.	TM_GeometricPrimitive	Zero or more (optional)
preview	URI pointing to a quick-look or browse image representing the ows:resource.	URI	Zero or more (optional).
contentByRef	A URI identifying a service which will return an immediately exploitable result by simply sending requests based on the URI for the ows:resource. The expectation is that the return type of this call will be in a well-known format.	URI	Zero or more (optional)
offering	Service or inline content offered by the ows:resource.	owc:OfferingType, see Table 6	Zero or more (optional)
active	This flag indicates the state of the resource within the context document. It can be interpreted by the caller as required (this may be defined in a profile or in the specific service extensions). [c]	Boolean Default value is TRUE	Zero or one (optional)
keyword	Keyword related to this ows:resource definition. Shall support an optional codelist parameter.	Character String type, not empty.	Zero or more (optional)

Name	Definition	Data type and values	Multiplicity and use
minScaleDenominator	Minimum scale for the display of the ows:resource. [d]	Double	Zero or one (optional)
maxScaleDenominator	Maximum scale for the display of the ows:resource. [d]	Double	Zero or one (optional)
resourceMetadata	Metadata about the ows:resource itself.	n/a	Zero or more (optional) [d]
folder	Definition of the folder structure in which the resource is placed.	CharacterString	Zero or one (optional)
extension	Any other element	n/a	Zero or more (optional)

[a] The semantics of rights is not defined here and needs to be defined in extension packages.
[b] The geospatial extent indicates a client that data intersecting with this area needs to be retrieved and ,if relevant portrayed. There is no specific requirement to hard clip the data to this boundary.
[c] The temporal extent indicates a client that data intersecting with this time interval needs to be retrieved and if relevant portrayed. There is no specific requirement to hard clip the data to this boundary.
[d] The scale denominator is defined with respect to a "standardized rendering pixel size" of 0.28 mm × 0.28 mm (millimeters). The definition is the same used in WMS 1.3.0 [OGC 06-042] and in Symbology Encoding Implementation Specification 1.1.0 [05-077r4]. Frequently, the true pixel size is unknown and 0.28 mm is a common actual size for current displays.

An OWC:Offering element can be a list mixing: OWS service operations, inline content and “by reference” content.

Table 6 — Definitions of owc:Offering elements

Name	Definition	Data type and values	Multiplicity and use
code	Code identifying the type of offering. [a]	URI	One (mandatory)
operation	Operations used to invoke the service. [b]	owc:OperationType, see Table 7	Zero or more (optional)
content	The offering content (inline or byRef).	owc:ContentType, see Table 8	Zero or more (optional)
styleSet	Style sets to style the content.	owc:StyleSetType,	Zero or more (optional)
extension	Any other element.	n/a	Zero or more (optional)

[a] Operations of an specific service request are defined in a separate extension of this document. Additionally, custom additions are supported on an ad-hoc basis without changing the core service offering type. Any modification of the parameter file types or semantics would require a new service offering code value.

An OWS:Operation contains both how to request information to the OWS service and eventually holds a container for the result of the operation.

Table 7 — Definitions of owc:Operation elements

Name	Definition	Data type and values	Multiplicity and use
code	Code identifying the type of Operation.	CharacterString [a]	One (mandatory)
method	Code identifying the verb type of Operation.	Character String type, not empty. Possible values are GET and POST. Default value is GET.	Zero or one (optional)
type	MIME type of the expected results.	Character String type, not empty	Zero or one (optional)
requestURL	Service Request URL. [b]	URL	One (mandatory)
payload	Payload content for non GET HTTP operations. [c]	owc:ContentType, see Table 5	Zero or one (optional)
result	Result of the operation.	owc:ContentType, see Table 5	Zero or one (optional)
extension	Any other element.	n/a	Zero or more (optional)

[a] Typically the OGC Service request type, e.g. "GetCapabilities" or "GetMap".
[b] Full request URL for an HTTP GET, and request URL for HTTP POST.
[c] POST and SOAP Payloads. Note: not necessarily XML as the content is defined by MIME-type.

The OWS:Content element is a container for direct content or OWS service results that supports both inline or “by reference” content.

Table 8 — Definitions of owc:Content elements

Name	Definition	Data type and values	Multiplicity and use
type	MIME type of the Content.	CharacterString not empty	One (mandatory)
URL	Referenced Content.	URL	Zero or one (optional) [a]
content	Inline content in the Content element.	Any	Zero or one (optional) [a]
extension	Any other element	n/a	One (mandatory)

[a] URL and content elements are mutually exclusive. One and only one should be populated in a specific content element.

6.3 Comparison between WMC and OWS Context

The following table compares the general information that is present in the OWS context document. When a cell in the first column is blank, it means that no correspondence has been found in OWC for an element present in WMC. When a cell in the second column is blank, it means that no correspondence has been found in WMC for an element present in OWC.

Table 9 — Comparison between OWC:Context and WMC general information

Name in OWC	Name in WMC	Differences
specReference	version	In OWC it is a URI and in WMC it is just the version numbers.
language		
id	id	
title	title	
abstract	abstract	
updateDate		
author	contactInformation	WMC presents complete structure based on WMS
publisher		
creatorApplication		
rights		
areaOfInterest	boundingBox	Optional in OWS, mandatory in WMC
creatorDisplay/ pixelWidh	width	
creatorDisplay/ pixelHeight	height	
creatorDisplay/ mmPerPixel		
timeIntervalOfInterest		
keyword	keyword	
resource	layer	Optional in OWC and mandatory in WMC. See the Table 5 for details.
contextMetadata	descriptionURL	DescriptionURL is not structured

Name in OWC	Name in WMC	Differences
		information.
extension	extension	

WMC was designed to support WMS services, even when the aim was to leave it open to other standards in the future. The introduction of OGC 05-005 states that "Presently, context documents are primarily designed for WMS bindings. However, extensibility is envisioned for binding to other services". That is not the case of the OWS Context, which is designed to support any OWS service and from its origins it is meant to describe how to deal with several services in small extensions.

It is not clear if the WMC was originally designed with other that KVP binding in mind. Even if it does not talk about bindings, the fact that provides all the needed information to build a request to a server (except the binding) without having to read the Capabilities document, it suggest that the authors supposed a KVP binding. On the contrary, OWS Context is designed to support KVP, XML SOAP binding and even some REST interfaces (at least the WMTS one).

An important difference between WMC and OWC is that WMC provides a set of parameter values and expects that the service will be able to construct the WMS request in the following form:

{serverURL}?SERVICE={serviceType}&VERSION={serviceVersion}&LAYERS={name}&SRS={SRS}&BBOX={BoundingBox}&{DimensionName}={DimensionCurrentValue}...&FORMAT={currentFormat}&STYLES={StyleName}

You can read more about this topic in subclause *6.4 Preformatted service requests*

Another important addition to OWC is the possibility to include geospatial content (data) that can be embedded directly inline or referenced in a URL. You can read more about this topic in subclause *6.5 Inclusion of direct content*

The following table compares the specific information of each resource in OWS context document and WMC.

Table 10 — Comparison between OWC Resource and WMC layer information

OWC Reosurce	WMC layer	Observations
id		
title	title	
abstract	abstract	
updateDate		
author		
publisher		
rights		

OWC Reosurce	WMC layer	Observations
geospatialExtent	Assumed to be the general BoundingBox	
temporalExtent		
preview		
`atom:entry/ atom:link[@rel="alternate"]`	DataURL	OWC conceptual model do not mention the possibility but Atom encoding support it natively
contentByRef		
offering	Composed using serverURL, serviceType, serviceVersion, name, BoundingBox, SRS, Dimension, format, Style	An offering has to be build in WMC combining information.
active	*Not* hidden	In OWS is optional and default is TRUE. In WMC is mandatory
Existence of an offering with and operation that has a code GetFeatureInfo	queryable	
Inside the requestURL or the payload of an operation (in an offering element) that has a code GetMap	name	in OWC it has to be parsed from a requestURL layers= or in a XML payload
Inside the requestURL of an operation (in an offering element) that has a GetMap code.	serverURL	
Offering code	serviceType	In OWC it is a URI and in WMC is a name such as “WMS”.
Inside the requestURL or the payload of an operation (in an offering element) that has a GetMap code.	SRS	In WMC it is a list of possible SRS’s.
Inside the requestURL or the payload of an operation (in an offering element) that has a GetMap code.	Dimension	In WMC it is a list of a complex type describing the dimensions and their possible parameters.
The type or the operation (in an offering element) that has a GetMap code.	Format	In WMC it is a list of a complex type describing the possible format parameters.

OWC Reosurce	WMC layer	Observations
keyword		
minScaleDenominator	Minimum scale for the display of the ows:resource d.	
maxScaleDenominator	Maximum scale for the display of the ows:resource d.	
resourceMetadata	metadataURL	
extension	extension	

OWC does not define anything like a ViewContextCollection but nothing prevents an application to provide a list of context document as a web page or as any other means.

6.4 Preformatted service requests

There are 2 possible alternatives for referencing an OGC service from the OWS Context document:

a) Reference the endpoint as an entry and some parameters as other elements, leaving some things open for the client to set.

b) Reference a complete service request (full URL) in a way that the client only has to sent it and wait for the data.

The alternative “a” was the one used in WMC. It assumes that the client knows how to build a service request from its pieces and eventually add what it is missing (such as the bounding box). It also assumes that the client will allow some “dynamic” interaction with the user that will be able to change the state of the view. This implies that the client will be able to modify the needed parameters and resubmit queries. The alternative “b” assumes that some clients will not be knowledgeable enough to formulate a request. This is a reasonable assumption if you are targeting more standard services types than WMC.

OWS Context standard uses alternative ‘b’ (which provides the exact URL of a complete request inside the requestURL or the payload of an operation (in an offering element). This decision has two main advantages:

- Simple clients are able to requests the offerings URL, then they just wait for the results and later present them, all of these without needing any knowledge on how to build requests (nor the details of each service and version) for all the possible web services included in the atom document. Just knowing the MIMEtype that will be returned is enough as long as the client knows how to show this MIMEType in the view. For “dynamic” geospatial clients, this complicates a bit the implementation, because the client has to know the way to parse the request (decompose it in its parts) and replace the parts that have to be changed to reflect the actions of the use.

- If the client knows how to deal with embedded GML, it automatically supports WFS requests and WPS requests that return GML.

In fact there is an alternative “c” that is between both alternatives “a” and “b”.

c) Reference a service request that still has some parameters to resolve in the form of a URL template. The client needs to substitute the still-to-resolve variables in the URL template to be able to get the information from the server, so the client becomes more dependent on particularities of each service supported.

The URL template language proposed here is standardized in the RFC6570 standard.

6.4.1 The case of the WMTS standard

As previously exposed, the alternative “a” is used in the WMC standard. Unfortunately, this standard only deals with WMS services (and it was developed before the WMTS was designed). WMC has an extensibility mechanism that could be used to adapt WMC to any other OGC standard. This could be particularly easy with WMTS due to that WMS and WMTS share many parameter names and meanings. This will not be explained because it is out of the scope of this deliverable.

As mentioned earlier, the alternative “b” requires to provide all the requests needed to fill a bounding box of a chosen scale. This is an example that shows how a WMTS offering could look like:

```
<feed xmlns="http://www.w3.org/2005/Atom"
              xml:lang='en'>
        ...
        <entry>
                <owc:offering
code="http://www.opengis.net/spec/owc/1.0/conf/atom/wmts">
                        <owc:operation code="GetCapabilities" method="GET"
type="application/xml"
href="http://www.opengis.uab.es/SITiled/ICC/1.0.0/WMTSCapabilities.xml"/>
                        <owc:operation code="GetTile" method="GET"
type="image/jpeg"
href="http://www.opengis.uab.es/SITiled/ICC/Topo250k_Vers5_ICC/default/Cat_topo
250k_v5_EPSG23031/200m/1/0.jpg"/>
                        <owc:operation code="GetTile" method="GET"
type="image/jpeg"
href="http://www.opengis.uab.es/SITiled/ICC/Topo250k_Vers5_ICC/default/Cat_topo
250k_v5_EPSG23031/200m/1/1.jpg"/>
                        <owc:operation code="GetTile" method="GET"
type="image/jpeg"
href="http://www.opengis.uab.es/SITiled/ICC/Topo250k_Vers5_ICC/default/Cat_topo
250k_v5_EPSG23031/200m/2/0.jpg"/>
```



```
                        <owc:operation code="GetTile" method="GET"
type="image/jpeg"
href="http://www.opengis.uab.es/SITiled/ICC/Topo250k_Vers5_ICC/default/Cat_topo
250k_v5_EPSG23031/200m/2/1.jpg"/>
                </owc:offering>
        <entry>
```

This approach has several problems.

- The context document is supposed to be client independent and thus it can't be tied to a particular width and height of the screen. The issue is that the client is supposing some screen size when it chooses the tiles that are going to be included in the context documents.

- We are providing several URLs for tiles without any information about how they have to be arranged in the space. This implies that the client has to read the TileMatrixSet structure stored in the GetCapabilities document, and then interpreted the URLs to finally associate a bounding box to each tile. The client will then calculate the needed tile position in the screen as well as the cutting needed in the borders of the viewport. This is as complicated as creating the URLs based on the service entry point in the client side.

There are two alternatives to solve this:

- Alternative b.1: The context document includes some information that explains how to distribute each tile in the screen (in screen coordinates) to the client. We propose to have top and left parameters that include (the screen coordinates, in pixels) the origin of the top/left corner of the image in the viewport. If the numbers are negative, it means that the top/left corner of the image starts outside of the viewport and should be cut by the client. By also providing the width and height of the image, the client can know if the bottom/right corner is also outside of the viewport and then cut the part that lies outside.

```
<feed xmlns="http://www.w3.org/2005/Atom"
              xml:lang='en'>
        ...
        <entry>
                <owc:offering
code="http://www.opengis.net/spec/owc/1.0/conf/atom/wmts">
                        <owc:operation code="GetCapabilities" method="GET"
type="application/xml"
href="http://www.opengis.uab.es/SITiled/ICC/1.0.0/WMTSCapabilities.xml"/>
                        <owc:operation code="GetTile" method="GET"
type="image/jpeg" width="640" height="480" top="-185" left="-485"
href="http://www.opengis.uab.es/SITiled/ICC/Topo250k_Vers5_ICC/default/Cat_topo
250k_v5_EPSG23031/200m/1/0.jpg"/>
```

```
                        <owc:operation code="GetTile" method="GET"
type="image/jpeg" width="640" height="480" top="-185" left="155"
href="http://www.opengis.uab.es/SITiled/ICC/Topo250k_Vers5_ICC/default/Cat_topo
250k_v5_EPSG23031/200m/1/1.jpg"/>
                        <owc:operation code="GetTile" method="GET"
type="image/jpeg" width="640" height="480" top="295" left="-485"
href="http://www.opengis.uab.es/SITiled/ICC/Topo250k_Vers5_ICC/default/Cat_topo
250k_v5_EPSG23031/200m/2/0.jpg"/>
                        <owc:operation code="GetTile" method="GET"
type="image/jpeg" width="640" height="480" top="295" left="155"
href="http://www.opengis.uab.es/SITiled/ICC/Topo250k_Vers5_ICC/default/Cat_topo
250k_v5_EPSG23031/200m/2/1.jpg"/>
                </owc:offering>
        <entry>
```

- Alternative b.2: The context document includes information that explains how to distribute each tile in the screen in CRS coordinates to the client. We propose to have a top , left, bottom and right parameters (or a bbox parameter) that include the CRS coordinates of the origin of the top/left and the bottom/right corners of the image. By combining this information with the nominal scaleDenominator of the view and the bounding box of the view, the client can calculate the exact position of each tile in the viewport. No pixel coordinate parameters are needed in this approach.

```
<feed xmlns="http://www.w3.org/2005/Atom"
              xml:lang='en'>
        ...
        <entry>
                <owc:offering
code="http://www.opengis.net/spec/owc/1.0/conf/atom/wmts">
                        <owc:operation code="GetCapabilities" method="GET"
type="application/xml"
href="http://www.opengis.uab.es/SITiled/ICC/1.0.0/WMTSCapabilities.xml"/>
                        <owc:operation code="GetTile" method="GET"
type="image/jpeg" left="258007" top="4655992" right="386007" bottom="4559992"
href="http://www.opengis.uab.es/SITiled/ICC/Topo250k_Vers5_ICC/default/Cat_topo
250k_v5_EPSG23031/200m/1/0.jpg"/>
                        <owc:operation code="GetTile" method="GET"
type="image/jpeg" left="386007" top="4655992" right="514007" bottom="4559992"
href="http://www.opengis.uab.es/SITiled/ICC/Topo250k_Vers5_ICC/default/Cat_topo
250k_v5_EPSG23031/200m/1/1.jpg"/>
                        <owc:operation code="GetTile" method="GET"
type="image/jpeg" left="258007" top="4559992" right="386007" bottom="4463992"
```

```
href="http://www.opengis.uab.es/SITiled/ICC/Topo250k_Vers5_ICC/default/Cat_topo
250k_v5_EPSG23031/200m/2/0.jpg"/>
                        <owc:operation code="GetTile" method="GET"
type="image/jpeg" left="386007" top="4559992" right="514007" bottom="4463992"
href="http://www.opengis.uab.es/SITiled/ICC/Topo250k_Vers5_ICC/default/Cat_topo
250k_v5_EPSG23031/200m/2/1.jpg"/>
                </owc:offering>
        <entry>
```

In the alternative “c”, we are not going to provide a list of complete tile URLs but a URL template for the WMTS requests, this will leave the client to find how many URLs he needs to fill the display window and where it has to put it. This excludes simple clients that are geo-enable but do not know about the particularities of each standard. It is suggested to use the URL template approach described in the RESTful approach of WMTS but replacing all the variables by their concrete values except the {TileRow} and {TileCol} ones.

```
<feed xmlns="http://www.w3.org/2005/Atom"
              xml:lang='en'>
        ...
        <entry>
                <owc:offering
code="http://www.opengis.net/spec/owc/1.0/conf/atom/wmts">
                        <owc:operation code="GetCapabilities" method="GET"
type="application/xml"
href="http://www.opengis.uab.es/SITiled/ICC/1.0.0/WMTSCapabilities.xml"/>
                        <owc:operation code="GetTile" method="GET"
type="image/jpeg"
href="http://www.opengis.uab.es/SITiled/ICC/Topo250k_Vers5_ICC/default/Cat_topo
250k_v5_EPSG23031/200m/{TileRow}/{TileCol}.jpg"/>
                </owc:offering>
        <entry>
```

Note that even if the URL template has not been described for the KVP syntax, it is still possible to express it as KVP syntax in a URL template (as it is done in RESTful syntax) as shown in the following example:

```
<feed xmlns="http://www.w3.org/2005/Atom"
              xml:lang='en'>
        ...
        <entry>
                <owc:offering
code="http://www.opengis.net/spec/owc/1.0/conf/atom/wmts">
```

```
                        <owc:operation code="GetCapabilities" method="GET"
type="application/xml"
href="http://www.opengis.uab.es/SITiled/ICC/1.0.0/WMTSCapabilities.xml"/>
                        <owc:operation code="GetTile" method="GET"
type="image/jpeg" href=”http://www.opengis.uab.es/cgi-
bin/MiraMon.cgi?service=WMTS&request=GetTile&version=1.0.0&layer=Topo250k_Vers5
_ICC2&style=default&format=image/jpeg&TileMatrixSet=Cat_topo250k_v5_EPSG23031&T
ileMatrix=200m&TileRow={TileRow}&TileCol={TileCol}"/>
                </owc:offering>
        <entry>
```

A similar template trick could be applied to generalize the template solution with the SOAP approach.

The authors of this ER recommend the alternative “b.1”.

6.5 Inclusion of direct content

One of the benefits of the OWS Context is the capability to also to connect to direct content, i.e. data files.

The mechanism is implemented by the current ows:content element that allows both in-line (embedded) content and by-reference content (linked content) but not both (see Table 8).

Example of in-line content:

```
<owc:content type="application/gml+xml">
   <my_srf:RoadCollection xsi:schemaLocation="http://www.opengis.net/gml/3.2
http://www.opengis.net/gml/3.2.1/gml.xsd
http://www.opengis.net/owc/1.0/examples/example1 SpringRoadField.xsd"
           xmlns:gml=" http://www.opengis.net/gml/3.2 "
           xmlns:my_srf=" http://www.opengis.net/owc/1.0/examples/example1 "
       gml:id="ID_ROADS1">
      <my_srf:road>
         <my_srf:Road gml:id="ID_ROAD1">
            <my_srf:position
               <gml:LineString gml:id="ID_LINEROAD1">
                  <gml:pos>300 200</gml:pos>
                  <gml:pos>350 222</gml:pos>
               </gml:LineString
            </my_srf:position>
            <my_srf:width>4.1</my_srf:width>
            <my_srf:name>M30</my_srf:name>
         </my_srf:Road
```

```
      </my_srf:RoadCollection>
</owc:content>
```

Example of by-reference content:

```
<owc:content type="application/gml+xml" href="SpringRoadField.gml"/>
```

This element is used in 4 places:

- An offering can have **direct content** instead of the definition of an operation to get it.
- An operation that uses PUT or POST methods needs a URL and some **content payload**. In this case, it is expected that the content is include in-line because, in general, the payload of this requests is not very long.
- An operation can have a **content result** of a previous execution that can be useful when working offline. In this case, it is expected that the content is an in-line kind because, the operation URL acts as a reference for the data and to provide a second reference for the same data seems confusing and unnecessary.
- A StyleSet can be expressed as a list of **styles content** encoded in a style description language.

Allowing in-line content in an OWS Context document is a risky decision because it opens the door to a new format for multipart data[1]. This is not the intention of OWS Context authors.

The general idea is to provide an easy and simple way to transport simple geospatial objects and annotations, while long datasets are expected to be included as references to external files. Nevertheless this is just a recommendation and the standard does not impose explicit size limits on in-line content. Please note that, in OWS 9 a new standard candidate called GeoPackage is intended to be able to carry long geospatial datasets in a single SQLite database file. The relation between OWS Context and GeoPackage manifest will be discussed later in subclause *10.3 GeoPackage manifest and OWS Context*.

6.5.1 Inclusion of direct content in the OWS Context scenarios

In OWS-9 scenarios the inclusion of direct content was used to transport dynamic information created, which might not be easy to register and keep up to date in a

[1] WCS GetCverage response already uses a multipart format based on RFC2387 MIME Multipart/Related Content-type.

catalogue. Examples of features stored are the trajectory of a hurricane or an escape route mainly included directly as KML information. This information was combined with more static base maps and dynamic raster information of the area of interest, all served by WMTS, WCS and WFS services and recorded as offerings in OWS context.

6.6 Pixel coordinates versus CRS coordinates

In WMC 05-005 it was clear that there were 2 coordinate systems in a context document: the CRS coordinates and the client view window coordinates. A WMC can contain the dimensions of the screen in a BoundingBox CRS coordinates (mandatory) and the window size in pixel coordinates (optional).

"The General element provides layer independent context information. It states the bounding box in units of a particular Spatial Reference System that represent the geographic extent of the map and a dimension as a pair of integers that represents the suggested pixel size of the map" [OGC 05-005]

The presence of both sizes allows the implementers to know the scale of the map at the time it was stored in the context document. Nevertheless, the client that reads the document can have a completely different window size and proportions, and there is no recommendation on how to support that. On the contrary, the document states that clients can do what they please:

"Negotiation between Context defined aspect ratio and typical client aspect ratio (according to the client's vendor) is left to the client." [OGC 05-005]

In OWS context, the presence of both CRS is even more critical. OWS Context has chosen full URLs (see subclause *6.4 Preformatted service requests*). While a WFS full URL leads to a GML response in CRS coordinates, a WMS full URL leads to an image response in pixel coordinates. In-line content can also be done in CRS coordinates (e.g. in GML) or in pixel coordinates (e.g. SVG). This can occur within the same file, so it is of upmost importance to be able to reconcile both coordinate spaces. This is particularly important for WMTS where not all pixel sizes are available from the server and where sometimes it is needed to be able to have elements directly in pixel coordinates.

This section supposes that we are dealing with an OWS Context document where all resources represent the same CRS (even if some are just images in a pixel space), which can be shown together in a single view (overlaid) (this is a requirement in WMC but it is not in OWC). In this case, CRS coordinate resources (such GML) can be converted into pixel space resources (such as map images) by using a linear equation that is common to all resources in the OWS Context document. Furthermore, it can be applied by a client that does not have deep knowledge of the OGC services involved and only knows how to render MIME types.

In a general use case, an OWS Context document can contain resources in several different CRSs representing regions of the space that do not overlap. Clients can still manage to generate representations of them by understanding how each OGC service instance works and handles geometry.

6.6.1 Width and height

As previously discussed, having a bounding box in CRS coordinates, and a width and height of the window in pixel coordinates is useful to know the relation between both coordinate systems and for the correct overlap of resources in both systems.

We have developed an XSL document to transform an OWS context document into an HTML map representation. In this language we have calculated the transformation between both coordinate systems.

```
<xsl:variable name="mapwidth" select="atom:feed/owc:width"/>
<xsl:variable name="mapheight" select="atom:feed/owc:height"/>

<xsl:variable name="inv_pixelsize_x" select="$mapwidth div (number(substring-
before(atom:feed/georss:where/gml:Envelope/gml:upperCorner, ' ')) -
number(substring-before(atom:feed/georss:where/gml:Envelope/gml:lowerCorner, '
')))"/>
<xsl:variable name="inv_pixelsize_y" select="$mapheight div (number(substring-
after(atom:feed/georss:where/gml:Envelope/gml:lowerCorner, ' ')) -
number(substring-after(atom:feed/georss:where/gml:Envelope/gml:upperCorner, '
')))"/>

<xsl:variable name="offset_pixel_x" select="-$inv_pixelsize_x *
number(substring-before(atom:feed/georss:where/gml:Envelope/gml:lowerCorner, '
'))"/>
<xsl:variable name="offset_pixel_y" select="-$inv_pixelsize_y *
number(substring-after(atom:feed/georss:where/gml:Envelope/gml:upperCorner, '
'))"/>

<!-- How to do the conversion from CSR coordinates to pixel coordinates
<xsl:value-of select="$inv_pixelsize_x*X_in_CSR_coords+$offset_pixel_x">
<xsl:value-of select="$inv_pixelsize_y*Y_in_CSR_coords+$offset_pixel_y">
-->
```

One of the main reasons for objects having width and height in the context document is that current visualization devices come with many sizes and screen proportions (e.g. big computer screens, small smartphones and medium size tablets). The OWS Context standard does not want to give the impression that an OWS Context document is tied to a particular pixel size. In fact, when a client interprets the context document, it is free to adapt the full service URLs to the ones needed to fill the screen in a way that tries to keep either the bounding box or the scale. When the context document points to a WMTS service, it is better to keep the original scale of the map because trying to adapt to the bounding box can easily end in a scale not available in the server side.

6.6.2 scaleDenominator

An alternative to provide the width and height is to just provide a nominal scale denominator. In fact this is another way to reconcile the pixel space and the CRS.

This approach avoids mentioning the size of the screen by just providing an initial scale for the context document. An integrated client that is initially empty, can start with that scale denominator and if there is a WMTS description the client can start requesting with a list of tiles. When the client view is too small, the client can simply ignore the unneeded tiles and when the view is too big, the client (if knowledgeable enough) can generate new tile URLs for the missing tiles (or else leave them blank). A client is still able to present the information, make overlaps and transform between pixel space and CRS coordinates using the initial scale, it is also able to perform very simple mathematics even if the client does not parse/understand the OGC requests.

6.7 Resource ordering and grouping

6.7.1 Resource displaying ordering

The use of Atom encoding is not giving the entries any particular ordering. Experience says that some clients may choose to order the entries in reverse chronological order via: atom:published or atom:updated. (That is, the most recently published or updated entries are shown at the top of > the list of syndication "headlines.")

After some discussions, it has been agreed that the order of the resources in the context document will be the order of the screen presentation, assuming that the first resource will be presented on top of the others.

6.7.2 Resource grouping

We have been speculating about different alternatives to group resources in a tree structure. At the time of writing this report, it seems that the Resource folder alternative will be finally adopted.

6.7.2.1 Resource tree

An obvious way to do this is by allowing the resources to be contained within another resource in a recursive way. The parent resource acts as a folder that can have several children. This conceptual model is the one suggested in OWS Common. Unfortunately, the main expected encoding (XML Atom) does not allow nesting entries. This is the reason why we discarded it.

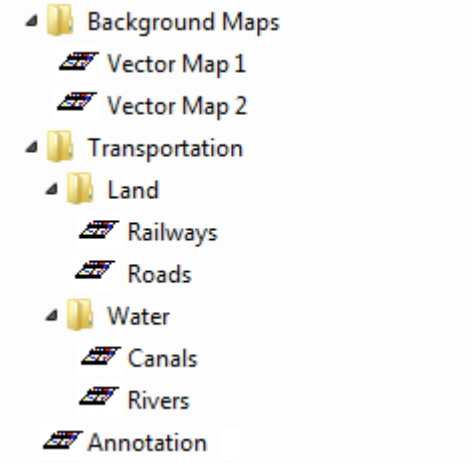
6.7.2.2 Resource folder

Add a “folder” property to a resource. A “/” delimits a folder path for the resource. If a folder is not specified or empty the layer is not placed in a folder.

The folder specifies a default 'tree' organization for a resource; this is intended to allow resources to be grouped. If the folder is empty then the resource exists at the root level. If a folder is specified the resource is placed in the folder. For example:

Resource: title: "VectorMap1", Folder: "Background Maps"
Resource: title: "VectorMap2", Folder: "Background Maps"
Resource: title: "Roads", Folder: "Transportation/Land"
Resource: title: "Railways", Folder: "Transportation/Land"
Resource: title "Rivers", Folder: "Transportation/Water"
Resource: title "Canals", Folder: "Transportation/Water"
Resource: title "Annotation", Folder: ""

This will be interpreted by the client as follows:



Note, in order to comply with the ordering rules for resources, all resources with a given folder must be grouped together. Hence the following is invalid:

Resource: title: "VectorMap1", Folder: "Background Maps"
Resource: title: "Roads", Folder: "Transportation/Land"
Resource: title: "VectorMap2", Folder: "Background Maps"

This is because you cannot put Roads between Vector Map 1 and Vector Map 2 without it being in the same folder.

Recent debates indicate that this could be the final selected option.

6.7.2.3 Resource themes

Reuse the "themes" approach that was included in WMTS. It allowed to add a tree structure to the flat layer structure, it also allowed a layer to be in more than one branch and even to have more than one classification tree.

In OWS context the same advantages are provided. It can be adapted to the Atom encoding in a simpler encoding than that of the WMTS by almost using the same conceptual model.

```
<themes>
    <id>Thematic</id>
    <theme>
       <id>Background Maps</id>
       <resourceRef>VectorMap1</resourceRef>
       <resourceRef>VectorMap2</resourceRef>
    </theme>
    <theme>
       <id>Transportation</id>
       <theme>
          <id>Land</id>
          <resourceRef>Railways</resourceRef>
          <resourceRef>Roads</resourceRef>
       </theme>
       <theme>
          <id>Water</id>
          <resourceRef>Rivers</resourceRef>
          <resourceRef>Canals</resourceRef>
       </theme>
       <resourceRef>Annotation</resourceRef>
   </theme>
</themes>
```

In recent debates it has been widely felt that the plasticity of this model is not needed. Particularly, some people think that to have to theme trees in the same document is redundant with the possibility of generating different OWC Context document for the same resources (In fact, OWS Context mainly contains just references to resources).

7 OWS Context Encodings

This clause discusses different OWS Context encodings. The most developed one is the ATOM encoding, and, in fact, the JSON encoding is planned to be generated from it by an XSLT which has not been discussed in OWS-9. This chapter is going to focus on the Atom encoding.

7.1 ATOM encoding

The name Atom applies to a pair of related standards. The Atom Syndication Format is an XML language used for web feeds (published as an IETF proposed standard in RFC 4287 in December 2005), while the Atom Publishing Protocol (AtomPub or APP) is a

simple HTTP-based protocol for creating and updating web resources (published as RFC 5023 in October 2007). Here we are mainly going to focus on the first one.

A **feed** contains basic metadata fields that are generic plus a list of entries, which may be headlines, full-text articles, excerpts, summaries, and/or links to content on a website, along with various metadata.

The Atom format was developed as an alternative to RSS. Ben Trott, an advocate of the new format that became Atom, believed that RSS had limitations and flaws. The main important advantage of Atom is that it is better defined and more clearly extensible the RSS.

Web feeds allow software programs to check for content and to update publications on a website. By providing a web feed, a site owner publishes a list (or "feed") of links to recent articles or content in a standardized, machine-readable format with some structured metadata. The feed can be downloaded by programs that use it, like websites that syndicate content from the feed, or by feed reader programs that allow Internet users to subscribe to feeds as well as view their content. The way Atom can be used here has been extended by the OWS Context standards that consider it an exchange format that can be generated on the flight by data catalogues, exchanged by email or used as a “project file” in GIS desktop applications.

7.1.1 ATOM encoding mapping

The OWS Context conceptual model can be easily mapped into the Atom elements. This mapping will not be discussed here because it has already been done in extent in OGC 12-084. Nevertheless, it is worth to discuss how Atom has been extended to include the geospatial properties needed to encode a geospatial context. OWS Context Atom encoding makes use of the extensibility mechanism to add the needed elements and attributes to the basic Atom specification.

Since OWS Context documents deeply rely on Atom, each OWS Context has to include the Atom namespace. Even if it is not required, defining the Atom encoding as the default namespace makes the encoding clearer:

```
<feed xmlns="http://www.w3.org/2005/Atom"
          xml:lang="en">
```

7.1.1.1 Adding geospatial extent to Atom

GeoRSS is a standard for encoding location as part of a Web feed. The name "GeoRSS" is derived from RSS, the most known web feed and syndication format, but it can be equally applied to Atom.

In GeoRSS, location content consists of geographical points, lines, and polygons of interest and related feature descriptions. GeoRSS feeds are designed to be consumed by

geographic software such as map generators. This way, Google Maps and Bing Maps are able to represent GeoRSS feeds on a map.

Currently, there are two levels of GeoRSS. The GeoRSS-Simple is a very lightweight format that supports basic geometries (point, line, box, polygon) and covers the typical simple use cases when encoding locations. GeoRSS GML is a formal Open Geospatial Consortium (OGC) GML Application Profile, and supports a greater range of features than GeoRSS-Simple.

OWS Context includes a geoss:where element in the general feed and in the atom:entry to define the extent of the feed or the entry. Please note that it is not intended to fully encode the geometry of the geospatial content of an atom:entry. The way this is done will be described in the next section.

It is expected that most of the applications will use the element gml:Envelope to describe the bounding box of an entry but it could also be done by a more complex element gml:Polygon.

```
<georss:where>
        <gml:Envelope srsName="EPSG:23031" srsDimension="2">
                <gml:lowerCorner>355000 4539000</gml:lowerCorner>
                <gml:upperCorner>475000 4619000</gml:upperCorner>
        </gml:Envelope>
</georss:where>
```

The use of GeoRSS in OWS Context documents forces to include both the georss namespace and the gml namespace used in GeoRSS.

```
<feed xmlns="http://www.w3.org/2005/Atom"
          xmlns:georss="http://www.georss.org/georss"
          xmlns:gml="http://www.opengis.net/gml"
          xml:lang="en">
```

Even if the abbreviated name for a namespace could be personalized in each document, we have seen that using abbreviated namespaces different of “georss” and “gml” makes some implementations of GeoRSS enhanced Atom readers to ignore the GeoRSS section (e.g. current versions of Google Maps), so changing the abbreviation is not recommended.

7.1.1.2 Adding geospatial content to Atom

The purpose of the atom:content element is to embed content in the Atom entries. This opens 2 options for OWS context:

- Include the geospatial content in the atom:content entry.
- Include the geospatial content in a new section and reserve that atom:content to provide more “textual” content.

After some tests, we saw that including geospatial content in atom:content confused some Atom readers or sometimes they simply chose to ignore it. We did not find any Atom reader that was able to make use of geospatial content in the atom:content element.

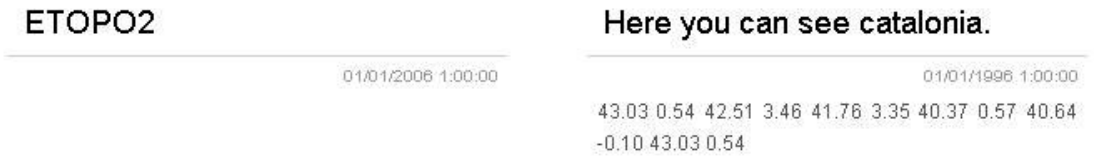


Figure 1 — Interpretation of a feed reader with geospatial information in content by the Opera Web Browser

For that reason, the SWG decided to go for the second option and create a new element called owc:offering that will be able to include any geospatial content in an independent complex element that will be ignored by normal atom readers. The OWC context standard recommends populating the atom:content with a textual content that will help atom readers to better present information for the entry

```
<feed xmlns="http://www.w3.org/2005/Atom"
              xml:lang='en'>
          ...
          <entry>
              ...

            <content type="html">
                  ENVISAT MERIS Level 1 Reduced Resolution &lt;br/&gt;
                           &lt;b&gt; Start :&lt;/b&gt; 2011-01-04T08:41:51.000Z
                           &lt;b&gt; End :&lt;/b&gt; 2011-01-04T09:25:21.000Z
                           &lt;br/&gt;&lt;b&gt;Acquisition Station :&lt;/b&gt;
PDHS-K
                           &lt;b&gt;Processing Center  :&lt;/b&gt; PDHS-K
                           &lt;br/&gt;&lt;b&gt; Orbit :&lt;/b&gt; 46257
              </content>

          </entry>
          ...
   </feed>
```

Include an owc:offering in an OWS Context document also forces to include the owc namespace in them.

```
<feed xmlns="http://www.w3.org/2005/Atom"
          xmlns:owc="http://www.opengis.net/owc/1.0"
          xml:lang="en">
```

7.1.1.3 Dublin Core namespace in Atom

Two attributes of the generic properties of the conceptual model, publisher and timeIntervalOfInterest, are mapped to the Dublin Core metadata elements dc:publisher and dc:date respectively. Two attributes of the resource properties of the conceptual model, publisher and temporalExtent, are also mapped to the Dublin Core metadata elements dc:publisher and dc:date respectively.

The use of these elements in an OWS Context document also forces to include the dc namespace in them:

```
<feed xmlns="http://www.w3.org/2005/Atom"
          xmlns:dc="http://purl.org/dc/elements/1.1/"
          xml:lang="en">
```

Please not the dc URI namespace name is the only one that ends with a slash character.

7.1.1.4 Other namespaces included in ATOM

Several other namespaces shall be used when including in-line content, post requests or offering responses encoded in XML. This namespaces depends on the version of the standards used and cannot be fully listed here.

Note that OWC was built in a way that does not depend on OWS Common elements or namespace. Nevertheless the OWS Common namespace could be necessary if imported by some of the services included in the offerings.

7.1.2 ATOM encoding validation

7.1.2.1 ATOM encoding validation using RelaxNG

The only schemas provided in the Atom specification RFC 4287 are in RelaxNG. They are not considered normative. Following this direction, the OWC.SWG has decided to publish a RelaxNG version of the schemas. The authors of this ER assume this decision but want to express their concerns about diversifying the number of validation schemas in OGC, which may force to adopt different sets of tools for schema validation. Also, there is an important argument in favor of trying to go for a W3C schema validation: OWS

Context will include content coming from other current and future standards (e.g. KML or GML in-line, WFS requests encoded in POST, etc.). The OWC.SWG decision is simply to not validate these fragments since they are not part of the OWC standard. The authors of this ER consider that it is always better to validate everything when possible, but in order to validate different fragments of the OWS Context documents it will force to use several validation tools, and this will only generate dificulties.

7.1.2.1.1 ATOM XML RelaxNG validation inclusion in a OWS Context document

There are two syntaxes for RelaxNG. One is called “compact” and it is not written in XML, while the other is called “regular” and is written in XML.

Below it is shown an example on how a context document can refer to a validation schema in RelaxNG compact, possibly it will be the one included in the OGC schema repository:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="../atom/2005/atom.rng" type="application/relax-ng-compact-
syntax"?>
<feed xmlns="http://www.w3.org/2005/Atom" ...
```

If you want to include the XML version of the RelaxNG notation you can do it by including the following line:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="../atom/2005/atom.rng" type="application/relax-ng-regular-
syntax"?>
<feed xmlns="http://www.w3.org/2005/Atom" ...
```

7.1.2.2 ATOM encoding validation using W3C schemas and Schematron

NOTE: This is not the validation schema finally adopted by the OWC.SWG (which provides RelaxNG validation schemas). Nevertheless, it can be used as a redundant validation of the RelaxNG one, additionally it is an interesting exercise on schema integration.

7.1.2.2.1 Pure ATOM validation using W3C schemas and Schematron

Currently, there are some implementations to validate Atom files with W3C schemas (XSD) that are available on the Internet: http://www.kbcafe.com/rss/atom.xsd.xml and http://tools.oasis-open.org/version-control/browse/wsvn/cmis/trunk/SchemaProject/schema/ATOM.xsd.

In fact, there is a previous effort to generate partial schemas for Atom which is already in the OGC schema repository: http://schemas.opengis.net\kml\2.2.0\atom-author-link.xsd.

It contains part of the Atom schema. The schema say: "*There is no official Atom XSD. This XSD is created based on: http://atompub.org/2005/08/17/atom.rnc. A subset of Atom as used in the ogckml22.xsd is defined here*."

After analyzing them, we use http://www.kbcafe.com/rss/atom.xsd, keeping it almost intact.

Two things are a bit surprising when using XSD in Atom coming from the OGC "tradition": element ordering and mixed elements. In the Atom specification elements there is no ordering in atom:feed and atom:entry and thus, it can appear in any ordering. This is implemented in XSD by using <choice> instead of <sequence>:

```
    <xs:complexType name="feedType">
        <xs:choice minOccurs="3" maxOccurs="unbounded">
            <xs:element name="author" type="atom:personType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="category" type="atom:categoryType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="contributor" type="atom:personType"
minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="generator" type="atom:generatorType"
minOccurs="0"/>
            <xs:element name="icon" type="atom:iconType" minOccurs="0"/>
            <xs:element name="id" type="atom:idType"/>
            <xs:element name="link" type="atom:linkType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="logo" type="atom:logoType" minOccurs="0"/>
            <xs:element name="rights" type="atom:textType" minOccurs="0"/>
            <xs:element name="subtitle" type="atom:textType" minOccurs="0"/>
            <xs:element name="title" type="atom:textType"/>
            <xs:element name="updated" type="atom:dateTimeType"/>
            <xs:element name="entry" type="atom:entryType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
        </xs:choice>
        <xs:attributeGroup ref="atom:commonAttributes"/>
    </xs:complexType>
```

Another surprising thing is the use of elements that can contain text as well as xml elements. This is possible by declaring those elements "mixed".

```
    <xs:complexType name="contentType" mixed="true">
        <xs:sequence>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
```

```
        <xs:attribute name="type" type="atom:ContentTypeType"/>
        <xs:attribute name="src" type=" xs:anyURI"/>
        <xs:attributeGroup ref="atom:commonAttributes"/>
    </xs:complexType>
```

This means that content can have either text or xml.

Many requirements in RFC 4287 cannot be implemented with W3C schemas alone. In this case, the classical approach in OGC is to use Schematron to impose some rules to XPath expressions. In fact, Schematron concept is very easy to use and has been directly adopted in RelaxNG XML encoding. The authors of this document were not able to find any previous Schematron validation document in Internet, so the creation of a new one was necessary. This is how our Schematron for Atom looks like:

```
<?xml version="1.0" encoding="UTF-8"?>
<iso:schema
    xmlns:iso="http://purl.oclc.org/dsdl/schematron"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    schemaVersion="ISO19757-3"
    xsi:schemaLocation="http://purl.oclc.org/dsdl/schematron
../../../../sch/2006/schematron.xsd">
    <iso:title>Schematron validation for atom</iso:title>
    <iso:ns prefix="atom" uri="http://www.w3.org/2005/Atom"/>
    <iso:pattern name="atom validation" id="atom">
        <!-- ATOM rules from RFC4287: -->
        <iso:rule context="/">
              <iso:assert test="atom:feed">Root element shall be
'feed'.</iso:assert>
        </iso:rule>
        <iso:rule context="atom:feed">
            <!-- ATOM rules from RFC4287: -->
            <iso:assert test="count(atom:icon)&lt;2">ATOM rule: Only one
element 'icon' is allowed in a 'feed'</iso:assert>
            <iso:assert test="count(atom:logo)&lt;2">ATOM rule: Only one
element 'logo' is allowed in a 'feed'</iso:assert>
            <iso:assert test="count(atom:id)=1">ATOM rule: Element 'id' is
mandatory for a 'feed'. Only one 'id' is allowed</iso:assert>
            <iso:assert test="count(atom:title)=1">ATOM rule: Element 'title'
is mandatory for a 'feed'. Only one 'title' is allowed</iso:assert>
            <iso:assert test="count(atom:subtitle)&lt;2">ATOM rule: Only one
element 'subtitle' is allowed in a OWSContext 'feed'</iso:assert>
            <iso:assert test="count(atom:updated)=1">ATOM rule: Only one
element 'updated' is allowed in a OWSContext 'feed'</iso:assert>
            <iso:assert test="count(atom:generator)&lt;2">ATOM rule: Only one
element 'generator' is allowed in a 'feed'</iso:assert>
```

```
            <iso:assert test="count(atom:rights)&lt;2">ATOM rule: Only one
element 'rights' is allowed in a 'feed'</iso:assert>
             <iso:assert test="atom:author or
not(atom:entry[not(atom:author)])">ATOM rule: An atom:feed must have an
atom:author unless all of its atom:entry children have an
atom:author.</iso:assert>
        </iso:rule>
        <iso:rule context="atom:entry">
            <!-- ATOM rules from RFC4287: -->
            <iso:assert test="atom:link[@rel='alternate'] or
atom:link[not(@rel)] or atom:content">ATOM rule: The entry with id="<iso:value-
of select="atom:id" />" must have at least one atom:link element with a rel
attribute of 'alternate' or an atom:content.</iso:assert>
            <iso:assert test="atom:author or ../atom:author or
atom:source/atom:author">ATOM rule: The entry with id="<iso:value-of
select="atom:id" />" must have an atom:author if its feed does
not.</iso:assert>
            <iso:assert test="count(atom:published)&lt;2">ATOM rule: The entry
with id="<iso:value-of select="atom:id" />" can only have one element
'published'.</iso:assert>
            <iso:assert test="count(atom:source)&lt;2">ATOM rule: The entry
with id="<iso:value-of select="atom:id" />" can only have one element
'source'.</iso:assert>
            <iso:assert test="not(atom:content) or (atom:content[@src] and
not(atom:content[text()])) or not(atom:content[@src])">ATOM rule: If the 'src'
attribute is present, 'content' be empty in entry with id="<iso:value-of
select="atom:id" />".</iso:assert>
            <iso:assert test="not(atom:content) or not(atom:content[@src]) or
atom:summary">ATOM rule: In entry with id="<iso:value-of select="atom:id" />",
'summary' is mandatory if the entry contains a 'content' that has a 'src'
attribute.</iso:assert>
            <iso:assert test="not(atom:content) or not(atom:content[@type]) or
not(contains(atom:content/@type,'/')) or
substring(atom:content/@type,1,5)='text/' or (string-
length(atom:content/@type)&gt;5 and (substring(atom:content/@type,string-
length(atom:content/@type)-3,4)='/xml' or substring(atom:content/@type,string-
length(atom:content/@type)-3,4)='+xml')) or atom:summary">ATOM rule: In entry
with id="<iso:value-of select="atom:id" />", 'summary' is mandatory if the
entry contains content that is encoded in Base64; i.e., the 'type' attribute of
the 'content' is a MIME media type, but is not an XML media type, does not
begin with "text/", and does not end with "/xml" or "+xml".</iso:assert>
             <iso:assert test="count(atom:content)&lt;2">ATOM rule: The entry
with id="<iso:value-of select="atom:id" />" can only have one element
'content'. In OWS Context you can use 'AlternateContent' for more
'content'</iso:assert>
            <iso:assert test="count(atom:id)=1">ATOM rule: Element 'id' is
mandatory for an 'entry'. Only one 'id' is allowed</iso:assert>
```

```
            <iso:assert test="count(atom:title)=1">ATOM rule: Element 'title'
is mandatory for an 'entry'. Only one 'title' is allowed in the entry with
id="<iso:value-of select="atom:id" />".</iso:assert>
            <iso:assert test="count(atom:updated)=1">ATOM rule: Element
'updated' is mandatory for an 'entry'. Only one 'updated' is allowed in the
entry with id="<iso:value-of select="atom:id" />".</iso:assert>
            <iso:assert test="count(atom:summary)&lt;2">ATOM rule: The entry
with id="<iso:value-of select="atom:id" />" can only have one element
'.</iso:assert>
            <iso:assert test="count(atom:rights)&lt;2">ATOM rule: The entry
with id="<iso:value-of select="atom:id" />" can only have one element
'rights'.</iso:assert>
            <iso:report test="atom:link[@rel='enclosure'] and
not(atom:link[@rel='enclosure']/@length)">ATOM NOTE: A 'length' attribute is
recommended for an contentByRef (atom:link with 'rel' attribute = "enclosure"]
) in the entry with id="<iso:value-of select="atom:id" />".</iso:report>
             <iso:report test="atom:link[@rel='enclosure'] and
not(atom:link[@rel='enclosure']/@type)">ATOM NOTE: A 'type' attribute is
recommended for an contentByRef (atom:link with 'rel' attribute = "enclosure"]
) in the entry with id="<iso:value-of select="atom:id" />".</iso:report>
        </iso:rule>
        <iso:rule context="atom:feed/atom:author">
            <iso:assert test="count(atom:name)=1">ATOM rule: Element 'name' is
mandatory in the 'author' of the 'feed'. Only one 'name' is
allowed.</iso:assert>
            <iso:assert test="count(atom:uri)&lt;2">ATOM rule: Only one element
'uri' is allowed in a 'feed'</iso:assert>
            <iso:assert test="count(atom:email)&lt;2">ATOM rule: Only one
element 'email' is allowed in a 'feed'</iso:assert>
        </iso:rule>
        <iso:rule context="atom:entry/atom:author">
            <iso:assert test="count(atom:name)=1">ATOM rule: Element 'name' is
mandatory in the 'author' in the entry with id="<iso:value-of
select="../atom:id" />". Only one 'name' is allowed.</iso:assert>
            <iso:assert test="count(atom:uri)&lt;2">ATOM rule: Only one element
'uri' is allowed in the entry with id="<iso:value-of select="../atom:id"
/>.</iso:assert>
            <iso:assert test="count(atom:email)&lt;2">ATOM rule: Only one
element 'email' is allowed in the entry with id="<iso:value-of
select="../atom:id" />.</iso:assert>
        </iso:rule>
    </iso:pattern>
</iso:schema>
```

We have not been able to find any popular way to include a .sch file in an XML document for default validation. However, a plug-in for XmlSpy that we use called ValidatorBuddy uses the following non standard way to declare it:

```
<?xml version="1.0" encoding="UTF-8"?>
<?valbuddy_schematron ../atom.sch?>
<feed xmlns="http://www.w3.org/2005/Atom"
```

Having together an XSD and an SCH file for Atom allows us to validate the generic rules for Atom. OWS Context adds more requirements that have to be encoded in validation files.

7.1.2.2.2 Dublin Core XML schema validation

The dc XML schemas were already created by the CSW editors some time ago and are stored in the official OGC repository. The main file can be found here:

http://schemas.opengis.net/csw/2.0.2/rec-dcmes.xsd

There is also extended terms:

http://schemas.opengis.net/csw/2.0.2/rec-dcterms.xsd

We use the first one without modification and the second one is not needed.

7.1.2.2.3 GEORSS XML schema validation

Some schemas for GeosRSS are stored in http://georss.org/xml/1.1. We particularly need georss.xsd and gmlgeorss311.xsd. There is a problem with the second one that uses the URI namespaces http://www.opengis.net/gml, since it is the same namespace used by the full version of GML 3.1.1, also used in WFS 1.1. XMLSpy validation detects collisions in the definitions of GML elements that are difficult to avoid. This is the main reason why OGC has abandoned the generation of schemas for GML profiles and instead recommends using the full XSD schemas and encodes the profile rules as Schematron rules. The Schematron for GeoRSS was not generated but there is a need to do so in order to validate WFS interactions in OWS Context.

7.1.2.2.4 OWS Context validation using W3C schemas and Schematron

Finally, there was the need for defining new elements in a new OWS Context XSD schema. There is also a need for defining new complex data types as well as root elements, which can then be used as new elements in the Atom document. These elements have to be defined as global elements.

Next we see how we define the new elements that can be included as feed child elements:


```
    <!-- Feed extensions for dealing with scale interval -->
    <element name="minScaleDenominator" type="double">
        <annotation>
            <documentation>Minimum scale for the display of the resource. This
element is an element extension for entry</documentation>
        </annotation>
    </element>
    <element name="maxScaleDenominator" type="double">
        <annotation>
            <documentation>Minimum scale for the display of the resource. This
element is an element extension for entry</documentation>
        </annotation>
    </element>

    <!-- Feed extensions for display sizes -->
    <element name="display" type="owc:DisplayType">
        <annotation>
            <documentation>Service or inline content offering for the resource
targetted at OGC compliant clients.</documentation>
        </annotation>
    </element>

    <complexType name="DisplayType">
        <annotation>
            <documentation>Information related to the display area used in the
creator application when the OWS Context document was produced. This set of
properties only applies to creator applications which are using a geographic
display and is supporting information to the exploiter of the OWS Context
document. Note the elements within creator display are intended as supporting
information (metadata) for clients and not properties which should control the
display size of the client opening the document.</documentation>
        </annotation>
        <choice maxOccurs="unbounded">
            <element name="pixelWidth" type="nonNegativeInteger"
minOccurs="0">
                <annotation><documentation>Width measured in pixels of the
display specified by Area of Interest</documentation></annotation>
            </element>
            <element name="pixelHeight" type="nonNegativeInteger"
minOccurs="0">
                <annotation><documentation>Height measured in pixels of the
display specified by Area of Interest.</documentation></annotation>
            </element>
            <element name="mmPerPixel" type="double" minOccurs="0">
                <annotation><documentation>The number of mm per pixel for the
display (allowing the real display size to be
calculated).</documentation></annotation>
            </element>
```

```
            <any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </choice>
    </complexType><feed xmlns="http://www.w3.org/2005/Atom"
```

One interesting feature that we are using here is the extendable code list. This is very useful to generate a list with some recommended items, but it can still be extended by the user without the need to modify the schemas. This trick consists of creating a data type which is an enumeration, and later create the code list type as a union of the enumeration and the string type.

```
    <!-- -Entry extensions for dealing with Offerings -->
    <simpleType name="KnownOfferingTypeCodeType">
        <restriction base="string">
            <enumeration
value="http://www.opengis.net/spec/owc/1.0/req/atom/csw"/>
            <enumeration
value="http://www.opengis.net/spec/owc/1.0/req/atom/wcs"/>
            <enumeration
value="http://www.opengis.net/spec/owc/1.0/req/atom/wfs"/>
            <enumeration
value="http://www.opengis.net/spec/owc/1.0/req/atom/wms"/>
            <enumeration
value="http://www.opengis.net/spec/owc/1.0/req/atom/wmts"/>
            <enumeration
value="http://www.opengis.net/spec/owc/1.0/req/atom/wps"/>
            <enumeration
value="http://www.opengis.net/spec/owc/1.0/req/atom/gml"/>
            <enumeration
value="http://www.opengis.net/spec/owc/1.0/req/atom/kml"/>
            <enumeration
value="http://www.opengis.net/spec/owc/1.0/req/atom/geotiff"/>
            <enumeration
value="http://www.opengis.net/spec/owc/1.0/req/atom/gmljp2"/>
            <enumeration
value="http://www.opengis.net/spec/owc/1.0/req/atom/gmlcov"/>
        </restriction>
    </simpleType>
    <simpleType name="OfferingTypeCodeType">
        <union memberTypes="owc:KnownOfferingTypeCodeType string"/>
    </simpleType>
    <simpleType name="MethodCodeType">
        <restriction base="string">
            <enumeration value="GET"/>
            <enumeration value="POST"/>
```



```
            <enumeration value="PUT"/>
            <enumeration value="DELETE"/>
            <enumeration value="HEAD"/>
            <enumeration value="OPTIONS"/>
        </restriction>
    </simpleType>
    <!--simpleType name="KnownOperationTypeCodeType">
        <restriction base="string">
            <enumeration value="capabilities"/>
            <enumeration value="describe"/>
            <enumeration value="data_access"/>
            <enumeration value="data_transaction"/>
            <enumeration value="status"/>
        </restriction>
    </simpleType>
    <simpleType name="OperationTypeCodeType">
        <union memberTypes="owc:KnownOperationTypeCodeType string"/>
    </simpleType-->
    <simpleType name="KnownOperationCodeType">
        <restriction base="string">
            <enumeration value="GetCapabilities"/>
            <enumeration value="DescribeFeature"/>
            <enumeration value="DescribeCoverage"/>
            <enumeration value="GetMap"/>
            <enumeration value="GetTile"/>
            <enumeration value="GetFeature"/>
            <enumeration value="GetFeatureInfo"/>
            <enumeration value="GetCoverage"/>
            <enumeration value="GetRecords"/>
            <enumeration value="Execute"/>
            <enumeration value="Transaction"/>
        </restriction>
    </simpleType>
    <simpleType name="OperationCodeType">
        <union memberTypes="owc:KnownOperationCodeType string"/>
    </simpleType>
```

Next we see how we define the new elements that can be included as entry child elements.

```
    <element name="offering" type="owc:OfferingType">
        <annotation>
            <documentation>Service or inline content offering for the resource
targetted at OGC compliant clients.</documentation>
        </annotation>
```

```
    </element>
    <complexType name="OfferingType">
        <annotation>
            <documentation>Service or inline content offering for the resource
targetted at OGC compliant clients.</documentation>
        </annotation>
        <choice maxOccurs="unbounded">
            <element name="operation" type="owc:OperationType" minOccurs="0"
maxOccurs="unbounded"/>
            <element name="content" type="owc:ContentType" minOccurs="0"
maxOccurs="unbounded"/>
            <element name="styleSet" type="owc:StyleSetType" minOccurs="0"
maxOccurs="unbounded"/>
            <any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </choice>
        <attribute name="code" type="owc:OfferingTypeCodeType" use="required">
            <annotation>
                <documentation>Code identifying the type of
offering.</documentation>
            </annotation>
        </attribute>
    </complexType>
    <complexType name="OperationType">
        <annotation>
            <documentation>
                Service operations useful to retrieve the information.
            </documentation>
        </annotation>
        <choice maxOccurs="unbounded">
            <element name="payload" type="owc:ContentType" minOccurs="0">
                <annotation>
                        <documentation>For POST and SOAP Requests, a payload
is required. Note: not necessarily XML as the content is defined by MIME-type.
If the content is text/xml or application/xml+* it MUST be present as a XML
fragment (without the xml... header) and the encoding MUST be the same as the
feed. </documentation>
                </annotation>
            </element>
            <element name="result" type="owc:ContentType" minOccurs="0">
                <annotation>
                        <documentation>Result is an optional parameter that
captures the result of an operation, in the form it was returned from the
server. This can be defined inline or as a reference. When the result content
is inline XML it should be as a XML fragment (without the xml... header) and
the encoding MUST be the same as the feed. </documentation>
                </annotation>
```

```
            </element>
            <any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </choice>
        <attribute name="method" type="owc:MethodCodeType" default="GET">
            <annotation>
                <documentation>This identifies the method (verb) of the
operation. Possible values are GET, POST... Default value is
GET.</documentation>
            </annotation>
        </attribute>
        <attribute name="code" type="owc:OperationCodeType" use="required">
            <annotation>
                <documentation>Code identifying the type of Operation.
Typically the OGC Service request type, e.g. "GetCapabilities" or "GetMap".
</documentation>
            </annotation>
        </attribute>
        <attribute name="href" type="anyURI" use="required">
            <annotation>
                <documentation>For HTTP GET the serviceURL item is used to
capture the entire request. For POST (and SOAP) requests, the serviceURL is
used to capture the address, and in addition a payload is required. See payload
below for an example of the POST request.</documentation>
            </annotation>
        </attribute>
        <attribute name="type" type="atom:ContentTypeType"/>
        <anyAttribute namespace="##other" processContents="lax"/>
    </complexType>
    <complexType name="ContentType" mixed="true">
        <sequence>
            <any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
        <attribute name="type" type="atom:ContentTypeType" use="required"/>
        <attribute name="href" type="anyURI"/>
    </complexType>

    <!-- Entry extension for folder -->
    <element name="forder" type="string">
        <annotation>
            <documentation>Definition of the folder structure in which the
resource is placed.</documentation>
        </annotation>
    </element>

    <!-- Feed extensions for dealing with symbology styles -->
    <!-- simpleType name="KnownStyleSetCodeType">
```

```
        <restriction base="string">
            <enumeration value="named"/>
            <enumeration value="full"/>
        </restriction>
    </simpleType>
    <simpleType name="StyleSetCodeType">
        <union memberTypes="owc:KnownStyleSetCodeType string"/>
    </simpleType -->
    <complexType name="StyleSetType">
        <annotation>
            <documentation>Defines a portrayal style for a resource
content.</documentation>
        </annotation>
        <choice maxOccurs="unbounded">
            <element name="name" type="string">
                <annotation>
                        <documentation>Unique name of the styleSet within a
given service </documentation>
                </annotation>
            </element>
            <element name="title" type="string">
                <annotation>
                        <documentation>A Human Readable Title for the
style</documentation>
                </annotation>
            </element>
            <element name="abstract" type="string" minOccurs="0">
                <annotation>
                        <documentation>Abstract or description of the
Style</documentation>
                </annotation>
            </element>
            <element name="legendURL" type="atom:linkType" minOccurs="0">
                <annotation>
                        <documentation>URL of a legend image for the
style</documentation>
                </annotation>
            </element>
            <element name="content" type="owc:ContentType" minOccurs="0">
                <annotation>
                        <documentation>
            The inline or a external reference to the style definition
                        </documentation>
                </annotation>
            </element>
            <any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
```

```
        </choice>
        <attribute name="default" type="boolean" default="false">
            <annotation>
                <documentation>Specifies the style to be applied when the
service is invoked (other styles are there as alternatives).</documentation>
            </annotation>
        </attribute>
    </complexType>
```

7.1.2.2.5 Combining validation schemas

Core Schematron rules can be inherited by a Schematron extension by using the <include> element. Namespaces are defined using the <ns> element.

```
<?xml version="1.0" encoding="UTF-8"?>
<iso:schema
    xmlns:iso="http://purl.oclc.org/dsdl/schematron"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    schemaVersion="ISO19757-3"
    xsi:schemaLocation="http://purl.oclc.org/dsdl/schematron
../../sch/2006/schematron.xsd">
    <iso:title>Schematron validation for OWS context</iso:title>
    <iso:ns prefix="atom" uri="http://www.w3.org/2005/Atom"/>
    <iso:ns prefix="dc" uri="http://purl.org/dc/elements/1.1/"/>
    <iso:ns prefix="georss" uri="http://www.georss.org/georss"/>
    <iso:ns prefix="gml" uri="http://www.opengis.net/gml"/>
    <iso:ns prefix="owc" uri="http://www.opengis.net/owc/1.0"/>

    <iso:include href="atom/2005/atom.sch#atom"/>
```

On the other hand, each XSD file has a single target namespace. To allow extensibility, the core is in one namespace and each extension has another namespace. This makes the XSD validation declaration a bit more complicated. Please, note the inclusion of both the Schematron for the OWS Context document and the RelaxNG Atom validation.

```
<?xml version="1.0" encoding="UTF-8"?>
<?valbuddy_schematron ../OWSContextCore.sch?>
<?xml-model href="../atom/2005/atom.rng" type="application/relax-ng-regular-
syntax"?>
<?xml-stylesheet href="owc2html.xsl" type="text/xsl"?>
<feed xmlns="http://www.w3.org/2005/Atom"
          xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:georss="http://www.georss.org/georss"
          xmlns:gml="http://www.opengis.net/gml"
          xmlns:owc="http://www.opengis.net/owc/1.0"
```

```
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.w3.org/2005/Atom
                         ../atom/2005/atom.xsd
                    http://purl.org/dc/elements/1.1/
                         http://schemas.opengis.net/csw/2.0.2/rec-dcmes.xsd
                    http://www.georss.org/georss
                         http://georss.org/xml/1.1/georss.xsd
                    http://www.opengis.net/gml
                         http://georss.org/xml/1.1/gmlgeorss311.xsd
                    http://www.opengis.net/owc/1.0
                         ../OWSContextCore.xsd"
          xml:lang="en">
```

7.1.3 ATOM examples

The following sections describe individual <ows:content> that can be combined in an Atom feed document.

7.1.3.1 ATOM offering content example

This is an example of GML embedded content in an <owc:offering> element.

```
<?xml version="1.0" encoding="UTF-8"?>
<?valbuddy_schematron ../OWSContextCore.sch?>
<?xml-model href="../atom/2005/atom.rng" type="application/relax-ng-regular-
syntax"?>
<feed xmlns="http://www.w3.org/2005/Atom"
              xmlns:dc="http://purl.org/dc/elements/1.1/"
              xmlns:georss="http://www.georss.org/georss"
              xmlns:gml="http://www.opengis.net/gml"
              xmlns:owc="http://www.opengis.net/owc/1.0"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation=
         "http://www.w3.org/2005/Atom ../atom/2005/atom.xsd
         http://purl.org/dc/elements/1.1/ ../../../csw/2.0.2/rec-dcmes.xsd
         http://www.georss.org/georss ../georss/1.1/georss.xsd
         http://www.opengis.net/gml ../georss/1.1/gmlgeorss311.xsd
         http://www.opengis.net/owc/1.0 ../OWSContextCore.xsd" xml:lang="en">

    <category scheme="http://www.opengis.net/spec/owc/specReference"
term="http://www.opengis.net/spec/owc/1.0/req/atom" label="This file is
compliant with version 1.0 of OGC Context"/>
    <id>https://portal.opengeospatial.org/twiki/bin/view/OWSContextswg/SpecAtomE
ncoding#1</id>
    <title>Context Example :: Examples of Services</title>
```



```
    <subtitle type="html">
    Example of context document with Every Service Included &lt;br/&gt;
    </subtitle>
    <updated>2012-02-21T11:58:23Z</updated>
    <author>
    <name>John Doe</name>
    <email>JohnDoe@example.com</email>
    <uri>http://example.com/~johndoe</uri>
    </author>
    <dc:publisher>ACME Project</dc:publisher>
    <generator uri="http://mysite.com/mycontext.php" version="1.0">
    ACME OWS Context Server
    </generator>
    <rights>
    Copyright (c) 2012.  Some rights reserved.  This feed
    licensed under a Creative Commons Attribution 3.0 License.
    </rights>
    <georss:where>
         <gml:Envelope srsName="EPSG:4326">
              <gml:lowerCorner>-90.0 -180.0</gml:lowerCorner>
              <gml:upperCorner>90.0 180.0</gml:upperCorner>
         </gml:Envelope>
    </georss:where>
    <dc:date>2009-01-23T09:08:56.000Z/2009-01-23T09:14:08.000Z</dc:date>
    <link rel='via' type='application/xml'
href='http://www.acme.com/collections/xxx.xml' title='ex XML metadata'/>
    <category scheme="http://www.acme.com/category"
                term="Common Operating Picture"
                label="COP"/>

    <!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->
    <!-- Resource (Inline GML offerings) -->
    <!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->

    <entry>
         <id>http://www.someurl.com/annotation1</id>
         <title>Some Useful Annotation</title>
         <updated>2012-05-10T14:35:00.400Z</updated>
         <content type="html">
           Some Useful Annotation&lt;br/&gt;
         </content>

         <owc:offering code="http://www.opengis.net/spec/owc/1.0/conf/atom/gml">
            <owc:content type="application/gml+xml">
              <my_srf:RoadCollection gml:id="ID_ROADS1"
xmlns:my_srf="http://www.opengis.net/owc/1.0/examples/example1"
                   xmlns:gml="http://www.opengis.net/gml/3.2"
```

```
                    xsi:schemaLocation="http://www.opengis.net/gml/3.2
../../../gml/3.2.1/gml.xsd

    http://www.opengis.net/owc/1.0/examples/example1 SpringRoadField.xsd">
              <my_srf:road>
                <my_srf:Road gml:id="ID_ROAD1">
                  <my_srf:position>
                    <gml:LineString  gml:id="ID_LINEROAD1">
                      <gml:pos>300 200</gml:pos>
                      <gml:pos>350 222</gml:pos>
                    </gml:LineString>
                  </my_srf:position>
                  <my_srf:width>4.1</my_srf:width>
                  <my_srf:name>M30</my_srf:name>
                </my_srf:Road>
              </my_srf:road>
            </my_srf:RoadCollection>
          </owc:content>
        </owc:offering>
    </entry>

</feed>
```

That uses the following GML application schema

```
<?xml version="1.0" encoding="windows-1252"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xmp="http://www.opengis.net/owc/1.0/examples/example1"
targetNamespace="http://www.opengis.net/owc/1.0/examples/example1"
elementFormDefault="qualified" version="1.0">
    <import namespace="http://www.opengis.net/gml/3.2"
schemaLocation="../../../gml/3.2.1/gml.xsd"/>
    <!--XML Schema document created by ShapeChange-->
    <element name="RoadCollection" type="xmp:RoadCollectionType"
substitutionGroup="gml:AbstractFeature"/>
    <complexType name="RoadCollectionType">
        <complexContent>
            <extension base="gml:AbstractFeatureType">
                <sequence>
                    <element name="road" type="xmp:RoadPropertyType"
minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
```

```
    </complexType>
    <complexType name="RoadCollectionPropertyType">
        <sequence minOccurs="0">
            <element ref="xmp:RoadCollection"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    </complexType>
    <element name="Road" type="xmp:RoadType"
substitutionGroup="gml:AbstractFeature"/>
    <complexType name="RoadType">
        <complexContent>
            <extension base="gml:AbstractFeatureType">
                <sequence>
                    <element name="position" type="gml:CurvePropertyType"/>
                    <element name="width" type="double"/>
                    <element name="name" type="string"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <complexType name="RoadPropertyType">
        <sequence minOccurs="0">
            <element ref="xmp:Road"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    </complexType>
</schema>
```

7.1.3.2 ATOM WMS offerings

This is an example for a WMS service that is referenced in an offering:

```
<?xml version="1.0" encoding="UTF-8"?>
<?valbuddy_schematron ../OWSContextCore.sch?>
<?xml-model href="../atom/2005/atom.rng" type="application/relax-ng-regular-
syntax"?>
<feed xmlns="http://www.w3.org/2005/Atom"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:owc="http://www.opengis.net/owc/1.0"
          xsi:schemaLocation="http://www.w3.org/2005/Atom ../atom/2005/atom.xsd
             http://www.opengis.net/owc/1.0 ../OWSContextCore.xsd"
        xml:lang="en">
```

```
    <category scheme="http://www.opengis.net/spec/owc/specReference"
term="http://www.opengis.net/spec/owc/1.0/req/atom" label="This file is
compliant with version 1.0 of OGC Context"/>
    <id>http://www.opengis.net/owc/1.0/examples/simples-atom-in-the-world</id>
    <title>The Simples atom In The World</title>
    <updated>2012-02-21T11:58:23Z</updated>
        <author>
            <name>J</name>
        </author>
    <entry>
        <id>http://www.opengis.net/owc/1.0/examples/simples-atom-in-the-
world/1</id>
        <title>The Simples atom In The World</title>
        <updated>2012-02-21T11:58:23Z</updated>
        <content>The Simples atom In The World</content>
        <owc:offering code="http://www.opengis.net/spec/OWC/1.0/conf/atom/wms">
            <owc:operation code="GetCapabilities" href="http://b-
maps.com/map.cgi?REQUEST=GetCapabilities&amp;SERVICE=WMS&amp;VERSION=1.3.0"/>
            <owc:operation code="GetMap" href="http://b-
maps.com/map.cgi?VERSION=1.3.0&amp;REQUEST=GetMap&amp;CRS=CRS:84&amp;BBOX=-
97.105,24.913,-
78.794,36.358&amp;WIDTH=560&amp;HEIGHT=350&amp;LAYERS=BUILTUPA_1M,COASTL_1M,POL
BNDL_1M&amp;STYLES=0XFF8080,0X101040,BLACK&amp;FORMAT=image/png&amp;BGCOLOR=0xF
FFFFF&amp;TRANSPARENT=TRUE&amp;EXCEPTIONS=INIMAGE"/>
        </owc:offering>
    </entry>
</feed>
```

7.2 JSON encoding

Not much work has been done in OWS9 to develop this encoding. One of the ideas was to generate a JSON encoding from the XML encoding using an XSLT. The OWS context SWG has produced some examples on how this can be done, but it is still in progress.

The generation of a JSON encoding seems to be adequate for the use on JavaScript clients. Nevertheless, 2 difficulties are foreseen:

- Some OWS service requests need to be produced in XML encoding such as HTTP POST and HTTP SOAP. Sometimes there is no equivalent in KVP. This operation requires an XML payload that does not currently have JSON encoding. Even if we can generate one, the actual request to the servers will require a conversion back to XML.

- The main in-line content examples that have been produced so far are in GML or KML. Both are XML encodings with no direct equivalence in JSON. The use of

GeoJSON can be considered. Furthermore, in-line content needs to be presented in the browser screen. HTML4 and predecessors do not include any native vector renderization operation. HTML5 includes two approaches: SVG in-line and canvas. None of them are JSON. The first one is an XML notation that can be produced by an XSLT transformation from GML of KML, and the second one is a JavaScript API.

In the opinion of the authors of this ER, one of the main success factors in adopting JSON is that no cross domain vulnerability was imposed on it. XML AJAX had the imposition of the cross domain vulnerability restriction long time ago. This makes the use of a JSON document that comes from another server possible, even if it is different from the current page server. In ordinary circumstances this is not possible in AJAX. Avoiding XML AJAX cross domain vulnerability restriction using Cross Origin Resource Sharing (CORS). There is an alternative to avoid the cross domain vulnerability restriction in AJAX, but unfortunately requires server provider collaboration.

Let's say your application lives on example.com and you want to pull data from www.example2.com. Normally if you tried to make this type of AJAX call, the request would fail and the browser would throw an origin mismatch error. With CORS, www.example2.com can choose to allow requests from example.com by simply adding a HTTP header:

```
Access-Control-Allow-Origin: http://example.com
```

Access-Control-Allow-Origin can be added to a single resource under a site or across the entire domain. To allow any domain to make a request to you set in the HTTP headers:

```
Access-Control-Allow-Origin: *
```

As a recommendation, any geospatial web service that generates GML, KML or any other kind of XML encoding file should add the latter line to the HTTP headers on all responses.

7.2.1 JSON encoding schemas

An important aspect about JSON encoding is producing schemas. There is also a schema language for JSON (http://davidwalsh.name/json-validation) used by the RESTful API. The opinion of the authors of this ER is that it could be used to normalize this standard extension.

7.3 HTML5 encoding

HTML5 is a W3C Candidate Recommendation for HTML that introduces several important modifications to solve historical problems in HTML. We have not defined any

complete encoding for OWS Context in HTML5, but parts of HTML5 could be included in an OWC Context encoded in XML. Particularly interesting for presenting geospatial features on a web browser are the “canvas” and the “SVG inline”.

7.3.1 HTML5 canvas encoding

The HTML5 element is used to draw graphics, on the fly, via scripting (usually JavaScript). The element is only a container for graphics; you must use a script to actually draw the graphics. A canvas is a drawable region defined in HTML code with height and width attributes. Canvas has several methods for drawing paths, boxes, circles, characters, and adding images.

This means that an OWS Context document could contain vector features expressed in canvas language as a direct content that can be later used by a JavaScript engine to render the objects.

```
    <owc:offering>
        <owc:content type="text/javascript">
<![CDATA[
var c=document.getElementById("myMap");
var ctx=c.getContext("2d");
ctx.fillStyle="#FF0000";
ctx.fillRect(0,0,150,75);
]]>
        </owc:content>
    </owc:offering>
```

7.3.2 HTML5 in-line SVG encoding

HTML5 natively supports in-line SVG, thus eliminating the need for an <object> element and allowing HTML5 web browsers to render SVG without the need of external (Adobe) plugins. This means that an OWS Context document could contain vector features expressed in SVG as a direct content that can be later shown in a web browser.

```
    <entry>
        <svg xmlns="http://www.w3.org/2000/svg" version="1.1" width="600"
height="400">
          <defs>
             <marker id="TriangleYellow"
          viewBox="0 0 10 10" refX="0" refY="5"
          markerUnits="strokeWidth"
          markerWidth="4" markerHeight="3"
          orient="auto">
          <path d="M 0 0 L 10 5 L 0 10 Z"
             stroke-width="0" fill="yellow"/>
```

```
            </marker>
          </defs>
          <text x="30" y="80" font-family="Verdana" font-size="30"
fill="yellow">Barcelona</text>
          <path d="M 200 75 L 250 75 L 340 150"
            fill="none" stroke="yellow" stroke-width="7"
            marker-end="url(#TriangleYellow)"/>
        </svg>
    </entry>
```

7.3.3 Geolocation API

HTML5 includes a Geolocation API that can be used to get the geographical position of the user. Since this can compromise user privacy, the position is not available unless the user approves it. Geolocation is much more accurate for devices with GPS.

The getCurrentPosition() method to get the user's position is:

```
<script>
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.getCurrentPosition(showPosition);
    }
  else{alert("Geolocation is not supported by this browser.");}
  }
function showPosition(position)
  {
  alert("Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude)
  }
</script>
```

This functionality allows us to develop a web browser that moves us to our current position after uploading an OWS Context document. In fact, during OWS-9 we incorporated this code in our MiraMon Web Map client.

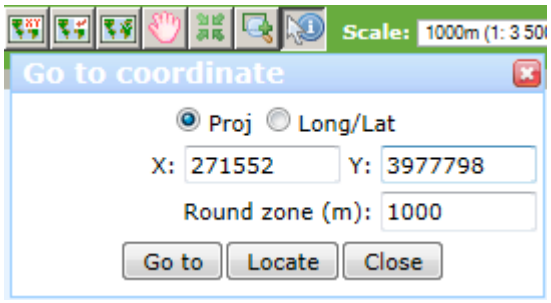


Figure 2 — Button "Locate" implements the javascript getCurrentPosition() function and return a value in the Barcelona Region

7.4 How to develop extensions

The Atom encoding has its own method for extensibility based on the <any> element so it is possible to extent either <atom:feed> or <atom:entry>. Also <owc:display> <owc:offering>, <owc:operation> and <owc:styleSet> have <any> elements as extension points.

In the XSD schemas, to extent an element you simply have to define another XSD with a new URI namespace and declare root elements and attributes that can later be used in the <any> elements. These elements can be simple types or complex types, in the case of complex types with more than one element, the Atom RFC explicitly says: "the structure of a structured extension element, including the order of its child elements, could be significant". Nevertheless, we recommend defining complex elements where order is not significant as in Atom elements. Please note that in some cases the definition of new elements could not be necessary if some namespace are already defined and can provide the required elements.

To control the father complex types where these elements are valid a Schematron rule can also be created in another sch file for the same namespace. This step is required even if the namespace was already defined by others.

7.4.1 OWS Context package example

OWS Context documents are basically a collection of data and services references with some additional metadata. This is good in a distributed environment and for devices that are permanently connected to the Internet, but it is not convenient for mobile devices that in some situations need to be autonomous such as when there is no Internet coverage.

In OWS9- Innovation, GeoPackage is being developed to deal with this problem. In practice, a GeoPackage is a way to encapsulate operation responses from OGC services on an SQLite file. GeoPackage can contain a manifest file (it was even speculated that it could be a database table) that describes the internal resources connected to the original services (as well as information on how they were obtained). This topic is still under discussion, but it is obvious that there are many conceptual similarities between both

purposes, as well as regarding the information that both have to contain. In the case that the manifest is encoded in XML (the decision is not still clear), the manifest could be defined as an extension of OWS Context. Here we are going to present the work done to illustrate how this can be done, also serving as an example of how OWS Context can be extended.

Please NOTE that the following approach has been tested in OWS 9 but has NOT been approved by the GeoPackage SWG.

This is the GeoPackageContext.xsd document that imports OWS Context and adds some data types, elements and attributes. Note that it also imports gmd to illustrate in-line ISO metadata inclusion.

```
<schema xmlns:gpkg="http://www.opengis.net/gpkg/1.0"
                xmlns="http://www.w3.org/2001/XMLSchema"
                xmlns:gmd="http://www.isotc211.org/2005/gmd"
                targetNamespace="http://www.opengis.net/gpkg/1.0"
elementFormDefault="qualified" version="1.0.0" xml:lang="en">
    <!-- 2012-09-04 -->
    <annotation>
        <documentation>Manifest schema for OGC GeoPackage as OWS context / Atom
extension</documentation>
    </annotation>
    <!-- ========================================================== -->
    <import namespace="http://www.isotc211.org/2005/gmd"
schemaLocation="../../iso/19139/20070417/gmd/gmd.xsd"/>
    <!-- ==========================================================
               Types, elements, and attributes
            ========================================================== -->
    <attribute name="area" type="gpkg:GeoPackageAreaNameType">
        <annotation>
            <documentation>Applies to Atom link elements in Atom feed element
but not in Atom entry elements. </documentation>
        </annotation>
    </attribute>
    <!-- ========================================================== -->
    <simpleType name="GeoPackageAreaNameType">
        <annotation>
            <documentation>Names of relative GeoPackage adjacent areas of data
coverage.</documentation>
        </annotation>
        <restriction base="string">
            <enumeration value="Here"/>
            <enumeration value="North"/>
            <enumeration value="NorthEast"/>
            <enumeration value="East"/>
            <enumeration value="SouthEast"/>
            <enumeration value="South"/>
```

```
            <enumeration value="SouthWest"/>
            <enumeration value="West"/>
            <enumeration value="NorthWest"/>
        </restriction>
    </simpleType>
    <!-- ========================================================== -->
    <attribute name="tableType" type="gpkg:TableTypeNameType">
        <annotation>
            <documentation>Applies to Atom link elements with rel="self" in
Atom entry elements.</documentation>
        </annotation>
    </attribute>
    <!-- ========================================================== -->
    <simpleType name="TableTypeNameType">
        <annotation>
            <documentation>Names of the types GeoPackage tables that contain
geospatial data content.</documentation>
        </annotation>
        <restriction base="string">
            <enumeration value="features">
                <annotation>
                    <documentation>Vector features with or without geometry
attribute(s), but no raster attribute(s).</documentation>
                </annotation>
            </enumeration>
            <enumeration value="rasters">
                <annotation>
                    <documentation>Rasters table with id primary key and
raster attribute(s), but no geometry attribute(s).</documentation>
                </annotation>
            </enumeration>
            <enumeration value="featuresWithRasters">
                <annotation>
                    <documentation>Vector features with id primary key,
geometry attribute(s) and raster attribute(s). </documentation>
                </annotation>
            </enumeration>
            <enumeration value="tiles">
                <annotation>
                    <documentation>Tile matrix set of tile matrices with
rasters at different zoom levels.</documentation>
                </annotation>
            </enumeration>
        </restriction>
    </simpleType>
    <!-- ========================================================== -->
```

```
    <!-- <attribute name="dataVersion" type="gpkg:VersionType"/> replaced by
atom:updated -->
    <annotation>
        <documentation>The following five attributes apply to Atom link
elements with rel="self" in Atom entry elements.</documentation>
    </annotation>
    <attribute name="geopackageVersion" type="gpkg:GeoPackageVersionType"/>
    <attribute name="sqliteVersion" type="gpkg:GeoPackageVersionType"/>
    <attribute name="spatialiteVersion" type="gpkg:GeoPackageVersionType"/>
    <attribute name="projVersion" type="gpkg:GeoPackageVersionType"/>
    <attribute name="geosVersion" type="gpkg:VersionType"/>
    <!-- ========================================================== -->
    <simpleType name="GeoPackageVersionType">
        <annotation>
            <documentation>Version type restricted to a string of the form
1.2.3, with an optional suffix, such as "a", "b", "-RC1"</documentation>
        </annotation>
        <restriction base="string">
            <pattern value="[1-9][0-9]*\.[0-9]+\.[0-9]+[a-z,A-Z,0-9,\-]*"/>
        </restriction>
    </simpleType>
    <!-- ========================================================== -->
    <simpleType name="VersionType">
        <annotation>
            <documentation>Version type specified by any string, to allow
convention used by any data provider.</documentation>
        </annotation>
        <restriction base="string"/>
    </simpleType>
    <!-- ========================================================== -->
    <attribute name="mdScope" type="gpkg:MetadataScopeType">
        <annotation>
            <documentation>Applies to gpkg:Metadata element.</documentation>
        </annotation>
    </attribute>
    <!-- ========================================================== -->
    <simpleType name="MetadataScopeType">
        <annotation>
            <documentation>Names of metadata scopes from ISO 19115 B.5.25
MD_ScopeCode code list, with extensions</documentation>
        </annotation>
        <restriction base="string">
            <enumeration value="undefined"/>
            <enumeration value="fieldSession"/>
            <enumeration value="collectionSession"/>
            <enumeration value="series"/>
            <enumeration value="dataset"/>
            <enumeration value="featureType"/>
```

```
            <enumeration value="feature"/>
            <enumeration value="attributeType"/>
            <enumeration value="attribute"/>
            <enumeration value="tile"/>
            <enumeration value="model"/>
            <enumeration value="catalogue"/>
            <enumeration value="schema"/>
            <enumeration value="taxonomy"/>
            <enumeration value="software"/>
            <enumeration value="service"/>
            <enumeration value="collectionHardware"/>
            <enumeration value="nonGeographicDataset"/>
            <enumeration value="dimensionGroup"/>
        </restriction>
    </simpleType>
    <!-- ========================================================== -->
    <attribute name="authority" type="anyURI">
        <annotation>
            <documentation>Authority for metadata model structure, e.g. ISO,
DC, etc.  Applies to gpkg:Metadata element.</documentation>
        </annotation>
    </attribute>
    <!-- ========================================================== -->
    <attribute name="columnName" type="string">
        <annotation>
            <documentation>Column name to identify the attribute or tile type
described by attributeType metadata, or combined with rowid, the attribute or
tile instance described by metadata.  Applies to gpkg:Metadata
element.</documentation>
        </annotation>
    </attribute>
    <!-- ========================================================== -->
    <attribute name="rowid" type="integer">
        <annotation>
            <documentation>ROWID to identify the row described by feature
instance metadata, or combined with columnName, the attribute or tile instance
described by metadata.  Applies to gpkg:Metadata element.</documentation>
        </annotation>
    </attribute>
    <!-- ========================================================== -->
    <attribute name="boundsRef" type="anyURI">
        <annotation>
            <documentation>Reference to current 2-D bounding polygons as well
as future 3-D bounding geometric solids.  Although OWS:Context allows
gml:Envelope here, GeoPackage requires gml:Polygon to carry gml:id attribute to
support gpkg:boundsRef attribute reference from GeoPackage extended Atom link
element with rel="self" attribute.</documentation>
```

```
        </annotation>
    </attribute>
    <!-- ============================================================ -->
    <attribute name="href" type="anyURI">
        <annotation>
            <documentation>Online source of metadata.  Applies to
gpkg:Metadata element.</documentation>
        </annotation>
    </attribute>
    <!-- ============================================================ -->
    <element name="Metadata" type="gpkg:MetadataWithAuthorityReferenceType">
        <annotation>
            <documentation>May be child of Atom feed element describing a
GeoPackage, or Atom entry element describing a GeoPackage data table. When it
appears as child of the Atom feed element, it implicitly applies to all
GeoPackage data tables. </documentation>
        </annotation>
    </element>
    <!-- ============================================================ -->
    <complexType name="MetadataWithAuthorityReferenceType" mixed="true">
        <sequence>
            <any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
        <attribute ref="gpkg:authority" use="required"/>
        <attribute ref="gpkg:mdScope" use="required"/>
        <attribute ref="gpkg:columnName"/>
        <attribute ref="gpkg:rowid"/>
        <attribute ref="gpkg:href"/>
    </complexType>
    <!-- ============================================================ -->
</schema>
```

This is the Schematron file (geoPackageContext.sch) that essentially defines the need for a category element that marks the Context document as a GeoPackage manifest type and which includes some rules to limit the parent names of the root elements defined in the XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<iso:schema
    xmlns:iso="http://purl.oclc.org/dsdl/schematron"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    schemaVersion="ISO19757-3"
    xsi:schemaLocation="http://purl.oclc.org/dsdl/schematron
../../sch/2006/schematron.xsd"><!-- 2012-09-04 Incomplete!-->
    <iso:title>Test ISO schematron file. Introduction mode</iso:title>
```

```
    <iso:ns prefix="atom" uri="http://www.w3.org/2005/Atom"/>
    <iso:ns prefix="owc" uri="http://www.opengis.net/owc/1.0"/>
    <iso:ns prefix="gpkg" uri="http://www.opengis.net/gpkg/1.0"/>
    <iso:include href="../../../../owc/1.0/OWSContextCore.sch#owc"/>
    <iso:pattern name="GeoPackage OWSContext validation" id="gpkg">
        <iso:rule context="atom:feed">
             <iso:assert
test="count(atom:category[@scheme='http://www.opengis.net/spec/gpkg/specReferen
ce'])=1 and
atom:category[@scheme='http://www.opengis.net/spec/gpkg/specReference']/@term='
http://www.opengis.net/spec/gpkg/1.0/req/atom'">A GeoPackage OWSContext
document shall have one 'specReference' element in the 'feed' element, encoded
as an atom:category with an attribute
scheme="http://www.opengis.net/spec/gpkg/specReference" an attribute
term="http://www.opengis.net/spec/gpkg/1.0/req/atom".</iso:assert>
        </iso:rule>
        <iso:rule context="gpkg:Metadata">
             <iso:assert test="parent::atom:feed | parent::atom:entry">An
'gpkg:Metadata' shall be a child of 'feed' or 'entry.</iso:assert>
        </iso:rule>
        <iso:rule context="@gpkg:area">
             <iso:assert test="parent::atom:link">An 'gpkg:area' shall be an
attribute of 'link'.</iso:assert>
        </iso:rule>
        <iso:rule context="@gpkg:boundsRef">
             <iso:assert test="parent::atom:link">An 'gpkg:boundsRef' shall be
an attribute of 'link'.</iso:assert>
        </iso:rule>
        <iso:rule context="@gpkg:sqliteVersion">
             <iso:assert test="parent::atom:link">An 'gpkg:sqliteVersion' shall
be an attribute of 'link'.</iso:assert>
        </iso:rule>
        <iso:rule context="@gpkg:spatialiteVersion">
             <iso:assert test="parent::atom:link">An 'gpkg:spatialiteVersion'
shall be an attribute of 'link'.</iso:assert>
        </iso:rule>
        <iso:rule context="@gpkg:geosVersion">
             <iso:assert test="parent::atom:link">An 'gpkg:geosVersion' shall
be an attribute of 'link'.</iso:assert>
        </iso:rule>
        <iso:rule context="@gpkg:projVersion">
             <iso:assert test="parent::atom:link">An 'gpkg:projVersion' shall
be an attribute of 'link'.</iso:assert>
        </iso:rule>
        <iso:rule context="@gpkg:geopackageVersion">
             <iso:assert test="parent::atom:link">An 'gpkg:geopackageVersion'
shall be an attribute of 'link'.</iso:assert>
```

```
        </iso:rule>
        <iso:rule context="@gpkg:tableType">
             <iso:assert test="parent::atom:link">An 'gpkg:tableType' shall be
an attribute of 'link'.</iso:assert>
        </iso:rule>
        <iso:rule context="@gpkg:authority">
             <iso:assert test="parent::gpkg:Metadata">An 'gpkg:authority' shall
be an attribute of 'gpkg:Metadata'.</iso:assert>
        </iso:rule>
        <iso:rule context="@gpkg:mdScope">
             <iso:assert test="parent::gpkg:Metadata">An 'gpkg:mdScope' shall
be an attribute of 'gpkg:Metadata'.</iso:assert>
        </iso:rule>
        <iso:rule context="@gpkg:columnName">
             <iso:assert test="parent::gpkg:Metadata">An 'gpkg:columnName'
shall be an attribute of 'gpkg:Metadata'.</iso:assert>
        </iso:rule>
        <iso:rule context="@gpkg:rowid">
             <iso:assert test="parent::gpkg:Metadata |
parent::atom:link[@rel='enclosure'] | parent::owc:operation">An 'gpkg:rowid'
shall be an attribute of 'gpkg:Metadata', 'atom:link with rel='enclosure' ' or
'owc:operation' .</iso:assert>
        </iso:rule>
        <iso:rule context="@gpkg:href">
             <iso:assert test="parent::gpkg:Metadata">An 'gpkg:@href' shall be
an attribute of 'gpkg:Metadata'.</iso:assert>
        </iso:rule>
    </iso:pattern>
</iso:schema>
```

This is a GeoPackage manifest example that uses both validation schemas:

```
<?xml version="1.0" encoding="UTF-8"?>
<?valbuddy_schematron ../geoPackageContext.sch?>
<?xml-model href="../atom/2005/atom.rng" type="application/relax-ng-regular-
syntax"?>
<feed xmlns="http://www.w3.org/2005/Atom"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:gmd="http://www.isotc211.org/2005/gmd"
        xmlns:georss="http://www.georss.org/georss"
        xmlns:gml="http://www.opengis.net/gml"
        xmlns:owc="http://www.opengis.net/owc/1.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:gpkg="http://www.opengis.net/gpkg/1.0"
             xsi:schemaLocation="http://www.w3.org/2005/Atom
../../../owc/1.0/atom/2005/atom.xsd
```

```
                                http://purl.org/dc/elements/1.1/
../../../csw/2.0.2/rec-dcmes.xsd
                                http://www.georss.org/georss
../../../owc/1.0/georss/1.1/georss.xsd
                                http://www.opengis.net/gml
../../../owc/1.0/georss/1.1/gmlgeorss311.xsd
                                http://www.opengis.net/owc/1.0
../../../owc/1.0/OWSContextCore.xsd
                                http://www.opengis.net/gpkg/1.0
../GeoPackageContext.xsd" xml:lang="en">
    <id>http://www.opengis.net/gpkg/uuid/sample.geopackage.manifest</id>
    <category scheme="http://www.opengis.net/spec/gpkg/specReference"
term="http://www.opengis.net/spec/gpkg/1.0/req/atom" label="GeoPackage Manifest
compliant with v 1.0 of OGC GeoPackage"/>
    <category scheme="http://www.opengis.net/spec/owc/specReference"
term="http://www.opengis.net/spec/owc/1.0/req/atom" label="This file is
compliant with version 1.0 of OGC Context"/>
    <category scheme="http://www.eionet.europa.eu/gemet" term="surface
transportation"/>
    <title>GeoPackage Manifest in OWS Context
(SampleGeoPackageManifestOWCv5.xml)</title>
    <subtitle type="html">
        Manifest containing references to GeoPackages for
&lt;b&gt;this&lt;/b&gt; and adjacent areas, and references to data tables in
this GeoPackage.
   </subtitle>
    <author>
        <name>John Doe</name>
        <email>JohnDoe@example.com</email>
        <uri>http://example.com/~johndoe</uri>
    </author>
    <dc:publisher>OGC OWS-9 Project</dc:publisher>
    <dc:date>2009-01-23T09:08:56.000Z</dc:date>
    <updated>2012-02-21T11:58:23Z</updated>
    <generator uri="http://www.terradue.com" version="1.0">CAS Context
Aggregator</generator>
    <rights>Copyright (c) 2012.  Some rights reserved.  This feed licensed under
a Creative Commons Attribution 3.0 License.</rights>
    <!-- Rights which apply to the context document  -->
    <georss:where>
        <!-- areaOfInterest  Geographic Area of interest of the users of the
content pointed or embedded in context document -->
        <!-- Although OWS:Context allows gml:Envelope here, GeoPackage requires
gml:Polygon to carry gml:id attribute to support gpkg:boundsRef attribute
reference from link rel="self" -->
        <gml:Polygon gml:id="here">
            <gml:exterior>
```

```
                <gml:LinearRing>
                    <gml:posList srsDimension="2">38.4921 44.2699 38.6058
43.4414 37.5318 43.2089 37.4215 44.0128 38.4921 44.2699</gml:posList>
                </gml:LinearRing>
            </gml:exterior>
        </gml:Polygon>
    </georss:where>
    <link rel="self" type="application/x-gpkg" href="sample.geopackage"
gpkg:area="Here" gpkg:boundsRef="#here" gpkg:sqliteVersion="3.7.9"
gpkg:spatialiteVersion="3.0.1" gpkg:geosVersion="3.3.1-CAPI-1.7.1"
gpkg:projVersion="4.7.1" gpkg:geopackageVersion="1.0.0"/>
    <!-- this gpkg -->
    <link rel="alternate" type="application/x-gpkg"
href="http://tec.army.mil/gpkg?service=GPKG&amp;request=GetGPKG&amp;id=sample.g
eopackage" gpkg:area="Here"/>
    <!-- source of this gpkg -->
    <link rel="related" type="application/x-gpkg" href="north.geopackage"
gpkg:area="North"/>
    <!-- GeoPackages for adjoining AOIs -->
    <link rel="related" type="application/x-gpkg" href="northeast.geopackage"
gpkg:area="NorthEast"/>
    <link rel="related" type="application/x-gpkg" href="east.geopackage"
gpkg:area="East"/>
    <link rel="related" type="application/x-gpkg" href="south.geopackage"
gpkg:area="SouthEast"/>
    <link rel="related" type="application/x-gpkg" href="south.geopackage"
gpkg:area="South"/>
    <link rel="related" type="application/x-gpkg" href="southwest.geopackage"
gpkg:area="SouthWest"/>
    <link rel="related" type="application/x-gpkg" href="west.geopackage"
gpkg:area="West"/>
    <link rel="related" type="application/x-gpkg" href="northwest.geopackage"
gpkg:area="NorthWest"/>
    <gpkg:Metadata gpkg:authority="http://schemas.opengis.net/iso/19139/"
gpkg:mdScope="series">
        <!-- metadata that applies to all data tables in this GeoPackage -->
        <gmd:MD_Metadata>
            <gmd:contact/>
            <gmd:dateStamp/>
            <gmd:identificationInfo/>
            <!-- incomplete example -->
        </gmd:MD_Metadata>
    </gpkg:Metadata>
    <entry>
        <!-- OWC Resource is a GeoPackage Table Reference -->
        <id>http://www.opengis.net/gpkg/uuid/sample.geopackage.roads</id>
        <category scheme="http://www.acme.com/category" term="vector features"
label="vector feature data"/>
```

```
        <title>Primary and secondary roads</title>
        <!-- abstract -->
        <dc:publisher>U.S. Census Bureau</dc:publisher>
        <updated>2012-01-31T12:00:00Z</updated>
        <dc:creator>ACME software</dc:creator>
        <rights>Copyright (c) 2012.  Some rights reserved.  This feed licensed
under a Creative Commons Attribution 3.0 License. </rights>
        <!-- Rights which apply to the resource definition -->
        <content type="xhtml">
            <div xmlns="http://www.w3.org/1999/xhtml"
xsi:schemaLocation="http://www.w3.org/1999/xhtml ../../../xhtml/xhtml1-
strict.xsd">
              Primary and secondary roads, no state or interstate highways.
             </div>
        </content>
        <georss:where>
            <!-- Although OWS:Context allows gml:Envelope here, GeoPackage
requires gml:Polygon to carry gml:id attribute to support gpkg:boundsRef
attribute reference from link rel="self" -->
            <gml:Polygon gml:id="conus">
                <gml:exterior>
                    <gml:LinearRing>
                        <gml:posList srsDimension="2">38.4921 44.2699 38.6058
43.4414 37.5318 43.2089 37.4215 44.0128 38.4921 44.2699</gml:posList>
                    </gml:LinearRing>
                </gml:exterior>
            </gml:Polygon>
        </georss:where>
        <dc:date>2009-01-23T09:08:56.000Z</dc:date>
        <link rel="self" type="application/x-gpkg" href="roads"
gpkg:boundsRef="#conus" gpkg:tableType="features"/>
            <owc:offering
code="http://www.opengis.net/spec/owc/1.0/conf/atom/wfs">
                <owc:operation code="GetCapabilities" method="GET"
type="application/xml"
href="http://geo.census.gov/arcgis_server?service=wfs&amp;version=1.1.0&amp;req
uest=GetCapabilities"/>
                <owc:operation code="GetFeature" method="GET"
type="application/gml+xml"
href="http://geo.census.gov/arcgis_server?service=wfs&amp;version=1.1.0&amp;req
uest=GetFeature&amp;typeName=roads"/>
            </owc:offering>
        <gpkg:Metadata
gpkg:authority="http://www.fgdc.gov/standards/projects/FGDC-standards-
projects/metadata/base-metadata/index_html"
gpkg:href="http://www.census.gov/geo/www/tlmetadata/tl2005femeta.txt"
gpkg:mdScope="series"/>
```

```
        <!-- online metadata example -->
        <gpkg:Metadata
gpkg:authority="http://www.fgdc.gov/standards/projects/FGDC-standards-
projects/metadata/base-metadata/index_html" gpkg:mdScope="dataset">
            <csdgm:CSDGM xmlns:csdgm="http://www.fgdc.gov/csdgm"/>
        </gpkg:Metadata>
        <gpkg:Metadata gpkg:authority="http://www.loc.gov/catdir/cpso/lcco/"
gpkg:mdScope="dataset">
            <!-- "mixed" content example for authorities that do not encode
metadata in XML -->
            Main Class: G -- GEOGRAPHY. ANTHROPOLOGY. RECREATION
            Subject:    United States--Geography--Databases.
            Authority: Census--Data processing.
        </gpkg:Metadata>
        <gpkg:Metadata gpkg:authority="http://dublincore.org"
gpkg:mdScope="featureType">
            <dc:publisher>U.S. Census Bureau</dc:publisher>
            <dc:coverage>conus</dc:coverage>
        </gpkg:Metadata>
        <gpkg:Metadata gpkg:authority="http://dublincore.org"
gpkg:mdScope="feature" gpkg:rowid="9876">
            <dc:source>Maryland DOT</dc:source>
        </gpkg:Metadata>
        <gpkg:Metadata gpkg:authority="http://schemas.opengis.net/iso/19139/"
gpkg:mdScope="attributeType" gpkg:columnName="surfaceType">
            <gmd:MD_DatatypeCode codeListValue="1234"
codeList="attributeTypes"/>
        </gpkg:Metadata>
        <gpkg:Metadata gpkg:authority="http://schemas.opengis.net/iso/19139/"
gpkg:mdScope="attribute" gpkg:columnName="geom" gpkg:rowid="9876">
            <gmd:DQ_DataQuality>
                <gmd:scope/>
                <gmd:report>
                    <gmd:DQ_AbsoluteExternalPositionalAccuracy>
                        <gmd:result>
                            <gmd:DQ_QuantitativeResult>
                                <gmd:valueUnit/>
                                <gmd:value/>
                            </gmd:DQ_QuantitativeResult>
                        </gmd:result>
                    </gmd:DQ_AbsoluteExternalPositionalAccuracy>
                </gmd:report>
            </gmd:DQ_DataQuality>
        </gpkg:Metadata>
    </entry>
</feed>
```

8 ATOM encoded OWS Context implementations in clients

Since OWS Context has been designed as an Atom extension there are already several applications that are able to parse and show some information. The geospatial part is included in the owc:offering element that is ignored by pure Atom readers. As a result of OWS-9 two integrated clients (one developed by CREAF-UAB and another by Envitia) and one GIS application (developed by CREAF) were updated to give support to the current candidate version of OWS Context. In the future it is expected that many geospatial clients and GIS viewers will adopt the format and thus new applications will emerge supporting it and even generate their own extensions.

8.1 ATOM encoded OWS Context in browsers

Some Internet browsers are able to present the document with a general title (atom:feed/atom:title) and a list of items (atom:feed/atom/entry) with title (atom:feed/atom/entry[]/atom:title), abstract (atom:feed/atom/entry[]/atom:title), time (atom:feed/atom/entry[]/atom:update), description (atom:feed/atom/entry[]/atom:content or (atom:feed/atom/entry[]/atom:summary))and download features (atom:feed/atom/entry[]/atom:link[@rel="enclosure"]/href)

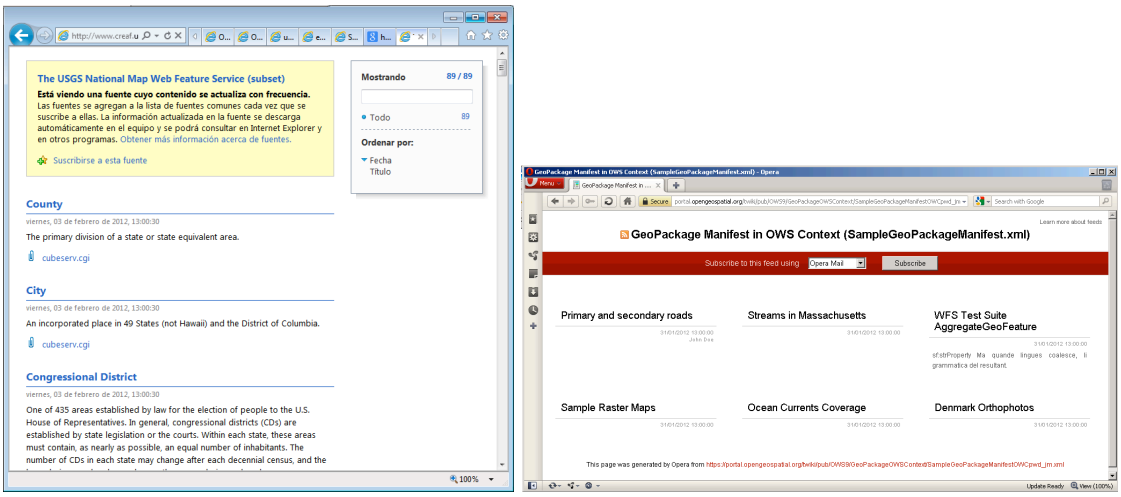


Figure 3 — OWS Context document presented in Internet Explorer and Opera

Be aware that some browsers do not include Atom readers and even when some do they do not want to interpret local files. Current situation about texted browsers is:

Table 11 — Current browsers support for Atom

Browsers	Version	Local File	Remote file	Subscription	Download
Internet Explorer	9.0.8	XML view	Atom view	yes	yes
Netscape	16..0.2	XML view	Atom view	yes	yes
Chrome	23.0	XML view	XML view	-	-

Safari	5.1.7	text viewer	Atom view	no	yes
Opera	12.12	Atom view	Atom view	yes	no

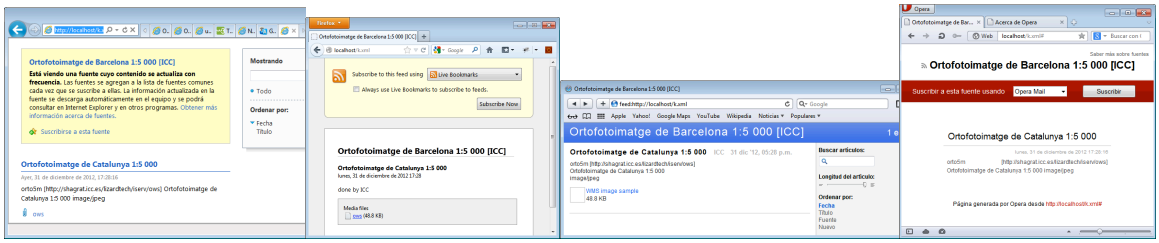


Figure 4 — OWS Context document presented in the four more common browsers with Atom support.

8.2 ATOM encoded OWS Context in Mass Market applications

The use of GeoRSS in Atom allows the same document to be interpreted by both Google maps and Bing Maps as a list of items that have the extent represented in a dynamic map.

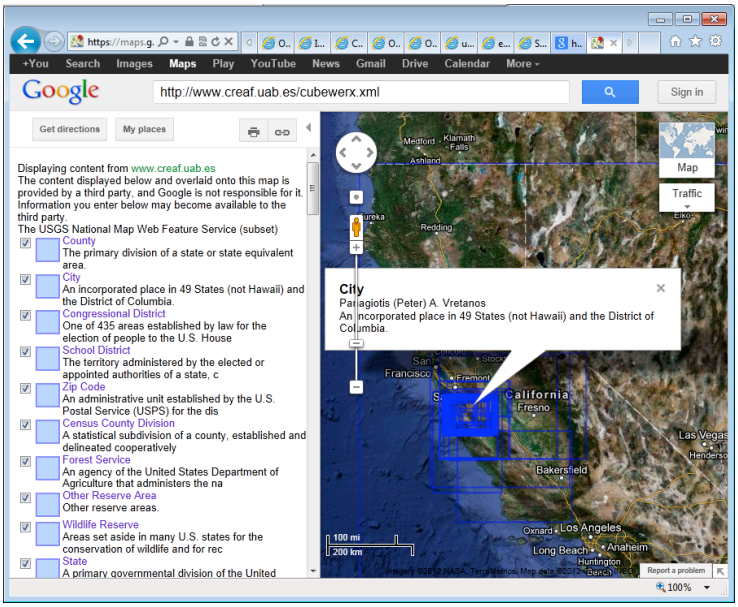


Figure 5 — OWS Context document presented in Google maps

Google maps is particularly easy to use for this purpose because you only have to put a URL to a public web service that contains the OWS Context file (local files and localhost services will not work) or in your internet browser type a URL such as: https://maps.google.com/maps?q={url_of_the_OWC}.

Bing maps does not make it that simple but it is still possible to use it if you access My places with your Microsoft live account and then press “Import” and type the OWS Context URL.

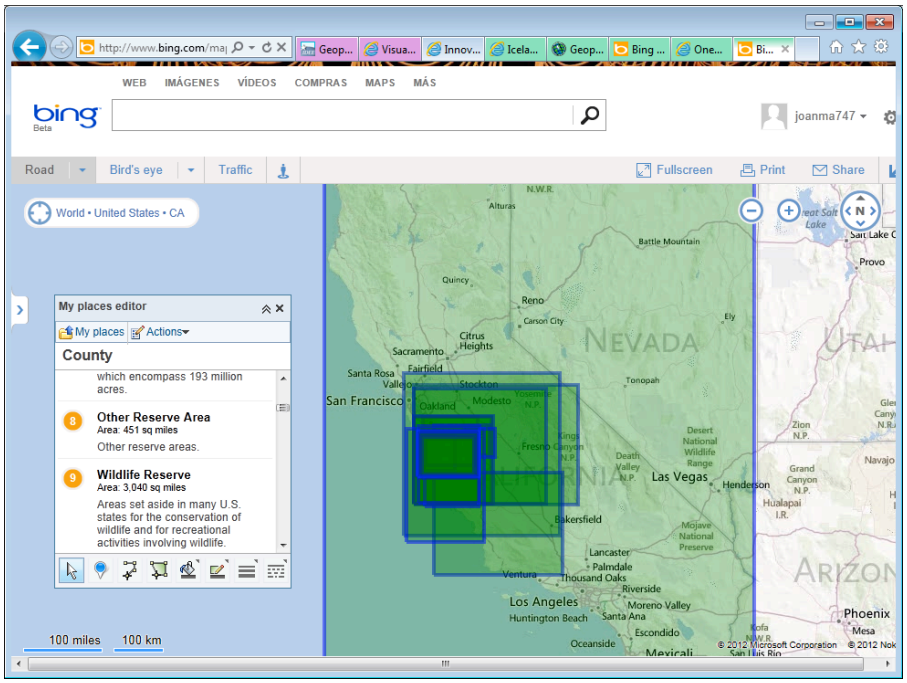


Figure 6 — OWS Context document presented in Bing maps My places.

8.3 ATOM encoded OWS Context visualization with XSLT

Due to the simplicity of the full URL to services approach, it is relatively simple to generate an XSLT transformation that lists the content of an OWS Context document, whether as a list or as an easy to read resources similar to the ones currently presented by web browsers but including the geospatial aspects. This exercise has not been done in OWS-9 but it could be a good way to explore an OWS Context document. It is also possible to generate a transformation that overlays data and presents an active legend that allows switching resources on and off.

8.3.1 Transform an OWS Context document into an overlaped view using XSLT

Since the OWS Context Atom encoding is an XML and the references to the services are included as full URLs, we have develop an XSLT transformation to transform an Atom feed into an HTML5 document that shows the exact status of the client view window, which was saved in the context document. To accomplish so and in order to calculate the scale of the map, we use the width and height of the client window as well as the extent of the view in world (geographic) coordinates. Then, we can mix elements in world coordinates (such as GML annotations in the XIMA profile included in a chapter of the

GMLJP2 standard with elements in pixel coordinates (such as WMS maps, WMTS tiles and direct SVG content). The result is an overlaid map with a legend that allows activation and deactivation of individual items. The XSL document is just 130 lines long.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:atom="http://www.w3.org/2005/Atom"
    xmlns:owc="http://www.opengis.net/owc/1.0"
    xmlns:owcht="http://www.opengis.net/owc/1.0/html"
    xmlns:svg="http://www.w3.org/2000/svg"
    xmlns:xima="http://www.opengis.net/xima"
    xmlns:georss="http://www.georss.org/georss"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:gml32="http://www.opengis.net/gml/3.2"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    exclude-result-prefixes="atom owc owcht svg georss gml gml32 xima xlink"
    >

<xsl:template match="/">

<xsl:variable name="mapwidth" select="atom:feed/owc:display/owc:pixelWidth"/>
<xsl:variable name="mapheight" select="atom:feed/owc:display/owc:pixelHeight"/>

<xsl:variable name="inv_pixelsize_x" select="$mapwidth div (number(substring-
before(atom:feed/georss:where/gml:Envelope/gml:upperCorner, ' ')) -
number(substring-before(atom:feed/georss:where/gml:Envelope/gml:lowerCorner, '
')))"/>
<xsl:variable name="inv_pixelsize_y" select="$mapheight div (number(substring-
after(atom:feed/georss:where/gml:Envelope/gml:lowerCorner, ' ')) -
number(substring-after(atom:feed/georss:where/gml:Envelope/gml:upperCorner, '
')))"/>

<xsl:variable name="offset_pixel_x" select="-$inv_pixelsize_x *
number(substring-before(atom:feed/georss:where/gml:Envelope/gml:lowerCorner, '
'))"/>
<xsl:variable name="offset_pixel_y" select="-$inv_pixelsize_y *
number(substring-after(atom:feed/georss:where/gml:Envelope/gml:upperCorner, '
'))"/>

<!-- How to do the conversion from world coordinates to pixel coordinates
<xsl:value-of select="$inv_pixelsize_x*355000+$offset_pixel_x">
<xsl:value-of select="$inv_pixelsize_y*4619000+$offset_pixel_y">
-->

<xsl:text disable-output-escaping='yes'>&lt;!DOCTYPE html></xsl:text>
<html>
<head>
```

```
<script>
<![CDATA[
function changeVisibility(layername)
{
   if (eval("document.legend."+layername+".checked"))
    document.getElementById(layername).style.visibility="visible";
   else
    document.getElementById(layername).style.visibility="hidden";
}
]]>
</script>
</head>

<body>
<h2><xsl:value-of select="atom:feed/atom:title"/></h2>
<xsl:value-of select="atom:feed/atom:subtitle"/>
<xsl:for-each select="atom:feed/atom:entry">
  <div>
    <xsl:attribute name="id">layer<xsl:number value="position()" format="1"
/></xsl:attribute>
    <xsl:attribute name="style">position:absolute; top:100px; left:5px;
visibility: visible; z-index:<xsl:number value="position()" format="1"
/>;</xsl:attribute>
    <xsl:choose>
        <xsl:when
test="owc:offering[@code='http://www.opengis.net/spec/owc/1.0/conf/atom/wms']/o
wc:operation[@code='GetMap']/@href">
             <img>
                 <xsl:attribute name="width"><xsl:value-of
select="$mapwidth"/></xsl:attribute>
                 <xsl:attribute name="height"><xsl:value-of
select="$mapheight"/></xsl:attribute>
                 <xsl:attribute name="src">
                     <xsl:value-of
select="owc:offering[@code='http://www.opengis.net/spec/owc/1.0/conf/atom/wms']
/owc:operation[@code='GetMap']/@href"/>
                 </xsl:attribute>
             </img>
        </xsl:when>
        <xsl:when
test="owc:offering[@code='http://www.opengis.net/spec/owc/1.0/conf/atom/wmts']/
owc:operation[@code='GetTile']/@href">
            <xsl:for-each
select="owc:offering[@code='http://www.opengis.net/spec/owc/1.0/conf/atom/wmts'
]/owc:operation[@code='GetTile']">
                 <img>
```

```
                    <xsl:attribute name="width"><xsl:value-of
select="@owcht:width"/></xsl:attribute>
                    <xsl:attribute name="height"><xsl:value-of
select="@owcht:height"/></xsl:attribute>
                    <xsl:attribute name="style">position:absolute;
top:<xsl:value-of select="@owcht:top"/>px; left:<xsl:value-of
select="@owcht:left"/>px; clip:rect(<xsl:call-template name="max2"><xsl:with-
param name="a" select="0"/><xsl:with-param name="b" select="-
@owcht:top"/></xsl:call-template>px,<xsl:call-template name="min2"><xsl:with-
param name="a" select="$mapwidth - @owcht:left"/><xsl:with-param name="b"
select="@owcht:width"/></xsl:call-template>px,<xsl:call-template
name="min2"><xsl:with-param name="a" select="$mapheight -
@owcht:top"/><xsl:with-param name="b" select="@owcht:height"/></xsl:call-
template>px,<xsl:call-template name="max2"><xsl:with-param name="a"
select="0"/><xsl:with-param name="b" select="-@owcht:left"/></xsl:call-
template>px); opacity:<xsl:value-of
select="@owcht:opacity"/>;filter:alpha(opacity=<xsl:value-of
select="100*@owcht:opacity"/>);</xsl:attribute>
                    <xsl:attribute name="src">
                        <xsl:value-of select="@href"/>
                    </xsl:attribute>
                </img>
            </xsl:for-each>
        </xsl:when>
        <xsl:when test="count(svg:svg)>0">
            <xsl:copy-of select="svg:svg"/>
        </xsl:when>
        <xsl:otherwise>
            <!-- I do nothing so the div exists but it is empty -->
        </xsl:otherwise>
    </xsl:choose>
  </div>
</xsl:for-each>

<div id="legend">
<xsl:attribute name="style">position:absolute; top:100px; left:<xsl:value-of
select="$mapwidth + 10"/>px; visibility: visible;</xsl:attribute>
<form name="legend">
<xsl:for-each select="atom:feed/atom:entry">
<input type="checkbox" checked="true">
    <xsl:attribute name="name">layer<xsl:number value="position()" format="1"
/></xsl:attribute>
    <xsl:attribute name="onchange">changeVisibility('layer<xsl:number
value="position()" format="1" />')</xsl:attribute>
    <xsl:value-of select="atom:title"/><br/>
</input>
</xsl:for-each>
</form>
```

```
</div>

</body>
</html>
</xsl:template>

<xsl:template name="min2">
    <xsl:param name="a" />
    <xsl:param name="b" />
    <xsl:choose>
        <xsl:when test="$a &gt;= $b">
            <xsl:value-of select="$b" />
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="$a" />
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>

<xsl:template name="max2">
    <xsl:param name="a" />
    <xsl:param name="b" />
    <xsl:choose>
        <xsl:when test="$a &gt;= $b">
            <xsl:value-of select="$a" />
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="$b" />
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>

</xsl:stylesheet>
```

Figure 7 — OWS Context document translated to HTML by just using XLST.

8.4 ATOM encoded OWS Context in a JavaScript smart clients

In the OWS-9 interoperability experiments, we have added the OWS Context support to a web map browser application called MiraMon Map Browser. The application is able to write a context document and read it again to recover the context of the previous view, it can even load a context generated by another application. When a context file is opened, the user is able to select the layer he/she wants to add to the current view.

Figure 8 — A map browser build with JavaScript is presenting a context document for content selection.

8.5 ATOM encoded OWS Context in desktop implementations.

The possibility to store direct links to content in OWS Context is particularly interesting for GIS desktop solutions. This paper also presents the development made in the MiraMon desktop GIS solution to include OWS Context. MiraMon software is able to deal either with local files, web services and database connections. As in any other GIS solution, MiraMon team designed its own file for storing and sharing the status of a GIS session that is called MiraMon map (MMM). The new OWS Context format is now adopted as a way to migrate the current MMM project file to an OWS interoperable format that can be shared with other future applications implementing OWS Context. The extensibility of the format makes it possible to map concepts in the MMM to current OWS Context elements (such as titles, data links, extent, etc) and to generate new elements that are able to include all extra content not currently covered by OWS Context.

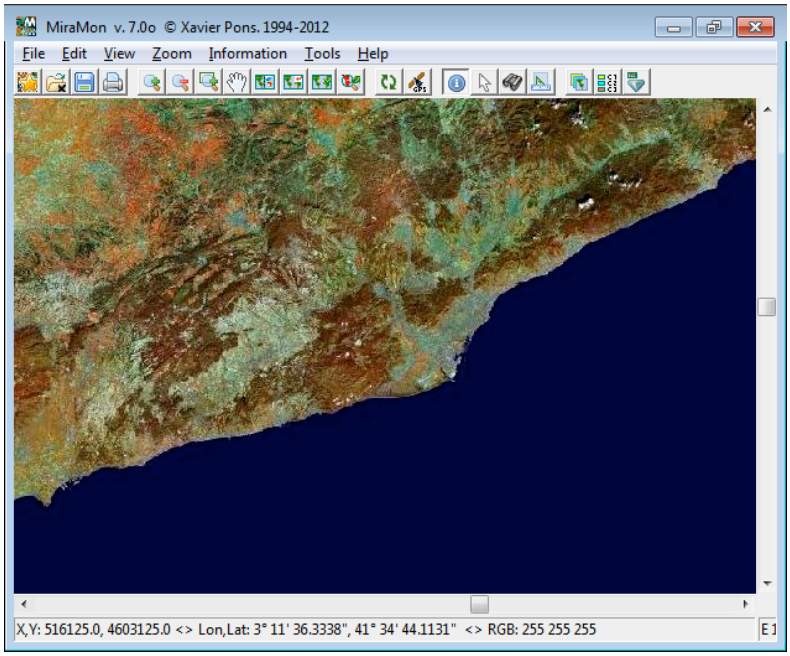


Figure 9 — A MiraMon GIS pro presenting the content of a context document.

8.6 Transform a WMC document into an OWS Context document.

In the section *6.3 Comparison between WMC and OWS Context* we have been discussing the similarities and differences of WMC and OWS Context. Both documents can be encoded in XML so, in principle it is possible to develop an XSLT conversion between them for the WMS offerings. However, this is not a straightforward exercise, thus it is going to require a character string analysis of the server URLs. The realization of this transformation was discarded during the timeframe of OWS-9.

8.7 Transform a WMS/WFS Capabilities document into an OWS Context document

A useful feature could be an XSLT transformation between a capabilities document (that is difficult to present to a user without a tool) into a context document, which can immediately be distributed in the internet as an Atom feed, it can also be seen in syndication tools as well as indexed by web crawlers.

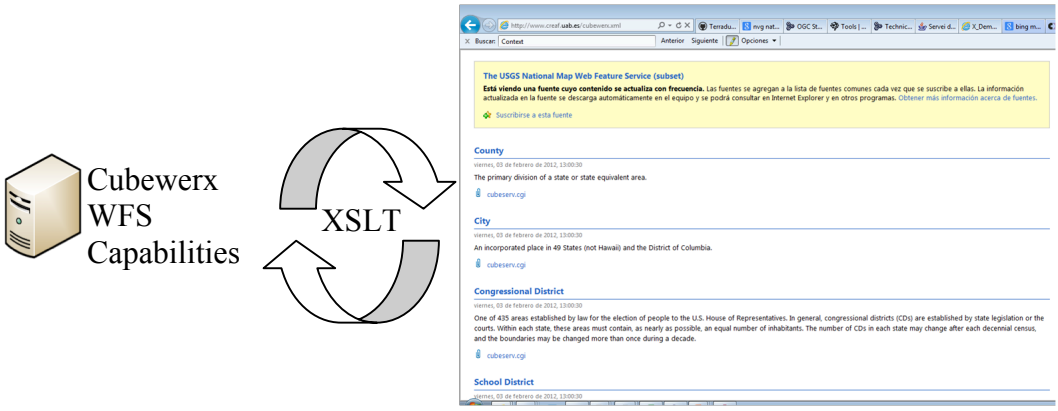


Figure 10 — Transformation between a WFS Capabilities document and an OWS Context done in OWS-9.

The ows2owc XSL Transformation file will create an OGC Context Document or Context Resource in Atom Encoding from an OGC Web Services GetCapabilities document. The idea and elaboration of the concept for WMS Capabilities was done by Terradue. Later CREAF extended it to WFS.

It currently supports:

- Web Map Specification (WMS) 1.1.1 and 1.3.0 (Elaborated by Terradue)
- Web Feature Specification (WFS) 1.1.0 (Extended by CREAF)

It accepts the following parameters:

- now : Parameter with the current or desired update date to insert on the atom:updated element (Mandatory)
- bbox : Restrict Context file to a specific BBOX in the format: minlon, minlat, maxlon, maxlat (Optional)
- entry : Restrict Context file to a given layer or feature (Optional). If not present the entire Capabilities document will be processed
- iconheight : Height of the preview image (Optional) for WMS. Default value is 100

- mapheight : Height of the map image (Optional) for WMS. Default value is 500
- mode : The processing mode (Optional)
    a. If equal to 'feed' it will produce a valid ATOM feed (default)
    b. If equal to 'fragment' it will only produce the entry with the feature or layer. It must be used with the entry parameter.

```
Example: xsltproc --stringparam entry "topp:member_map"
         --stringparam now "`date +%Y-%m-%dT%H:%M:%S`"
         --stringparam mode "fragment"
         ows2owc.xsl
         http://meet.opengeospatial.org:8080/geoserver/ows?service=WFS&request=
GetCapabilities'
```

It creates the following file:

```
<entry xmlns="http://www.w3.org/2005/Atom"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:georss="http://www.georss.org/georss"
xmlns:gml="http://www.opengis.net/gml"
xmlns:owc="http://www.opengis.net/owc/1.0/"
xmlns:ows="http://www.opengis.net/ows">
  <id>http://meet.opengeospatial.org:8080/geoserver/wfstopp:member_map/</id>
  <title>OGC members</title>
  <author><name>Raj Singh</name></author>
  <dc:publisher>Open Geospatial Consortium</dc:publisher>
  <updated>2012-12-20T17:01:57Z</updated>
  <dc:rights>Fee:NONE / Contraints:NONE</dc:rights>
  <georss:where>
    <gml:Polygon><gml:exterior><gml:LinearRing>
          <gml:posList>-41.302 -157.862 -41.302 174.805 60.671 174.805 60.671 -
157.862 -41.302 -157.862</gml:posList>
    </gml:LinearRing></gml:exterior></gml:Polygon>
  </georss:where>
  <link rel="enclosure" type="text/xml; subtype=gml/3.1.1" title="WFS output
for OGC members"
href="http://meet.opengeospatial.org:8080/geoserver/wfs?SERVICE=WFS&amp;VERSION
=1.1.0&amp;REQUEST=GetFeature&amp;BBOX=-41.302,-
157.862,60.671,174.805&amp;OUTPUTFORMAT=text/xml;
subtype=gml/3.1.1&amp;TYPENAME=topp:member_map&amp;MAXFEATURES=10"/>
  <link rel="describedby" type="text/xml" title="Description of Features"
href="http://meet.opengeospatial.org:8080/geoserver/wfs?SERVICE=WFS&amp;VERSION
=1.1.0&amp;REQUEST=describeFeatureType&amp;TYPENAME=topp:member_map"/>
```

```
  <link rel="via" type="text/xml" title="Original GetCapabilities document"
href="http://meet.opengeospatial.org:8080/geoserver/wfs?SERVICE=WFS&amp;VERSION
=1.1.0&amp;REQUEST=GetCapabilities"/>
  <content type="html">
    &lt;br/&gt;
    This resource is available from a OGC WFS Service (version 1.1.0) and it
contains the following access points:
    &lt;ul&gt;&lt;li&gt;
    &lt;a
href='http://meet.opengeospatial.org:8080/geoserver/wfs?SERVICE=WFS&amp;VERSION
=1.1.0&amp;REQUEST=GetFeature&amp;BBOX=-41.302,-
157.862,60.671,174.805&amp;OUTPUTFORMAT=text/xml;
subtype=gml/3.1.1&amp;TYPENAME=topp:member_map&amp;MAXFEATURES=10'&gt;
    GetFeature &lt;/a&gt; request in text/xml; subtype=gml/3.1.1
(atom:link[@rel="enclosure"])
    &lt;/li&gt; &lt;li&gt;
    &lt;a
href='http://meet.opengeospatial.org:8080/geoserver/wfs?SERVICE=WFS&amp;VERSION
=1.1.0&amp;REQUEST=GetCapabilities'&gt;
    GetCapabilities &lt;/a&gt; request (atom:link[@rel="via"])
    &lt;/li&gt; &lt;/li&gt; &lt;li&gt;
    &lt;a
href='http://meet.opengeospatial.org:8080/geoserver/wfs?SERVICE=WFS&amp;VERSION
=1.1.0&amp;REQUEST=describeFeatureType&amp;TYPENAME=topp:member_map'&gt;
    Describe Feature &lt;/a&gt; request for topp:member_map
(atom:link[@rel="describedby"])
    &lt;/li&gt;&lt;/ul&gt;

    &lt;p style='font-size:small'&gt;OGC Context CITE Testing XSLT (Extensible
Stylesheet Language Transformations) by Terradue Srl.&lt;/p&gt;
    </content>
  <owc:offering code="http://www.opengis.net/spec/owc/1.0/req/atom/wfs">
    <owc:operation method="GET" code="GetCapabilities"
href="http://meet.opengeospatial.org:8080/geoserver/wfs?SERVICE=WFS&amp;VERSION
=1.1.0&amp;REQUEST=GetCapabilities"/>
    <owc:operation method="GET" code="GetFeature"
href="http://meet.opengeospatial.org:8080/geoserver/wfs?SERVICE=WFS&amp;VERSION
=1.1.0&amp;REQUEST=GetFeature&amp;BBOX=-41.302,-
157.862,60.671,174.805&amp;OUTPUTFORMAT=text/xml;
subtype=gml/3.1.1&amp;TYPENAME=topp:member_map&amp;MAXFEATURES=10"/>
  </owc:offering>
</entry>
```

This transformation file works by going drought each layer or feature and creating a new Atom entry with an owc:offering element <xsl:template match="wms:Layer | Layer | wfs:FeatureType">.

```
<entry>
...
     <owc:offering>
         <xsl:attribute name="code"><xsl:value-of
select="$offering_code"/></xsl:attribute>
        <owc:operation method="GET" code="GetCapabilities">
            <xsl:attribute name="href">
                    <xsl:value-of select="$get_capabilities_request"/>
            </xsl:attribute>
        </owc:operation>
        <owc:operation method="GET">
            <xsl:attribute name="code"><xsl:value-of
select="$default_operation"/></xsl:attribute>
            <xsl:attribute name="href"><xsl:value-of
select="$get_request"/></xsl:attribute>
        </owc:operation>

        <xsl:for-each select="wms:Style | Style">
        <owc:styleSet>
            <owc:name><xsl:value-of select="wms:Name | Name"/></owc:name>
            <owc:title><xsl:value-of select="wms:Title | Title"/></owc:title>
            <owc:abstract><xsl:value-of select="wms:Abstract |
Abstract"/></owc:abstract>
            <owc:legendURL>
                <xsl:attribute name="href">
                    <xsl:value-of
select="wms:LegendURL/wms:OnlineResource/@xlink:href |
LegendURL/OnlineResource/@xlink:href"/>
                </xsl:attribute>
                <xsl:attribute name="type">
                     <xsl:value-of select="wms:LegendURL/wms:Format |
LegendURL/Format"/>
                </xsl:attribute>
            </owc:legendURL>
        </owc:styleSet>
        </xsl:for-each>
    </owc:offering>
...
</entry>
```

To define the service GetCapabilities request, firstly it is necessary to obtain the service name, version and online resource for the service :

```
<xsl:variable name="service_name"><xsl:choose>
    <xsl:when test="/wfs:WFS_Capabilities">WFS</xsl:when>
```

```
    <xsl:when test="/wms:WMS_Capabilities |
/WMT_MS_Capabilities">WMS</xsl:when>
    <xsl:otherwise>UNKNOWN</xsl:otherwise></xsl:choose>
</xsl:variable>

<xsl:variable name="version" select="/*/@version"/>

<xsl:variable name="capabilities_online_resource"

select="/wms:WMS_Capabilities/wms:Capability/wms:Request/wms:GetCapabilities/wm
s:DCPType/wms:HTTP/wms:Get/wms:OnlineResource/@xlink:href |

/WMT_MS_Capabilities/Capability/Request/GetCapabilities/DCPType/HTTP/Get/Online
Resource/@xlink:href |

/*/ows:OperationsMetadata/ows:Operation[@name='GetCapabilities']/ows:DCP/ows:HT
TP/ows:Get/@xlink:href"/>

<xsl:variable name="get_capabilities_request">
    <xsl:value-of select="$capabilities_online_resource"/>
    <xsl:if test="substring-
before($capabilities_online_resource,'?')=''">?</xsl:if>
    <xsl:value-of
select="concat('SERVICE=',$service_name,'&amp;VERSION=',$version,'&amp;REQUEST=
GetCapabilities')"/>
</xsl:variable>
```

To define the entry's bounding box, the XSLT file will look for the layer's bounding box element value or value of its ancestors bounding box element. This way it solves the issue of nested layers in WMS with spatial information being defined only at the top-most layer.

```
<xsl:variable name="maxX">
    <xsl:choose>
        <xsl:when test="$bbox!=''"><xsl:value-of
select="$coords/*[3]"/></xsl:when>
        <xsl:when test="ows:WGS84BoundingBox/ows:UpperCorner">
            <xsl:value-of select="substring-
before(ows:WGS84BoundingBox/ows:UpperCorner,' ')"/>
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="ancestor-or-
self::wms:Layer/wms:EX_GeographicBoundingBox/wms:eastBoundLongitude | ancestor-
or-self::Layer/LatLonBoundingBox/@maxx"/>
        </xsl:otherwise>
    </xsl:choose>
```

```
    </xsl:variable>
    <xsl:variable name="maxY">
    <xsl:choose>
        <xsl:when test="$bbox!=''"><xsl:value-of
select="$coords/*[4]"/></xsl:when>
        <xsl:when test="ows:WGS84BoundingBox/ows:UpperCorner">
            <xsl:value-of select="substring-
after(ows:WGS84BoundingBox/ows:UpperCorner,' ')"/>
        </xsl:when>
        <xsl:otherwise><xsl:value-of select="ancestor-or-
self::wms:Layer/wms:EX_GeographicBoundingBox/wms:northBoundLatitude | ancestor-
or-self::Layer/LatLonBoundingBox/@maxy"/></xsl:otherwise>
    </xsl:choose>
    </xsl:variable>
    <xsl:variable name="minX">
    <xsl:choose>
        <xsl:when test="$bbox!=''"><xsl:value-of
select="$coords/*[1]"/></xsl:when>
        <xsl:when test="ows:WGS84BoundingBox/ows:LowerCorner">
            <xsl:value-of select="substring-
before(ows:WGS84BoundingBox/ows:LowerCorner,' ')"/>
        </xsl:when>
        <xsl:otherwise><xsl:value-of select="ancestor-or-
self::wms:Layer/wms:EX_GeographicBoundingBox/wms:westBoundLongitude | ancestor-
or-self::Layer/LatLonBoundingBox/@minx"/></xsl:otherwise>
    </xsl:choose>
    </xsl:variable>
        <xsl:variable name="minY">
        <xsl:choose>
        <xsl:when test="$bbox!=''"><xsl:value-of
select="$coords/*[2]"/></xsl:when>
        <xsl:when test="ows:WGS84BoundingBox/ows:LowerCorner">
            <xsl:value-of select="substring-
after(ows:WGS84BoundingBox/ows:LowerCorner,' ')"/>
        </xsl:when>
        <xsl:otherwise><xsl:value-of select="ancestor-or-
self::wms:Layer/wms:EX_GeographicBoundingBox/wms:southBoundLatitude | ancestor-
or-self::Layer/LatLonBoundingBox/@miny"/></xsl:otherwise>
    </xsl:choose>
    </xsl:variable>
```

It then uses these values to define the entry's GeoRSS element:

```
<georss:where>
        <gml:Polygon>
```

```
          <gml:exterior>
            <gml:LinearRing>
              <gml:posList>
              <xsl:value-of select="concat($minY,' ',$minX,' ',$minY,'
',$maxX,' ',$maxY,' ',$maxX,' ',$maxY,' ',$minX,' ',$minY,' ',$minX)"/>
              </gml:posList>
             </gml:LinearRing>
          </gml:exterior>
        </gml:Polygon>
    </georss:where>
```

These values are also used to define the bounding box request by checking the CRS code to be used:

```
<!-- preference for Plate Carre on element -->
    <!-- if no crs available then check parent -->
    <xsl:variable name="crs">
    <xsl:choose>
        <xsl:when test="count(wms:CRS[.='EPSG:4326'] |
SRS[.='EPSG:4326'])!=0">EPSG:4326</xsl:when>
        <xsl:when test="count(wms:CRS[.='CRS:84'] |
SRS[.='CRS:84'])!=0">CRS:84</xsl:when>
        <xsl:when test="count(wms:CRS[1] | SRS[1])!=0"><xsl:value-of
select="wms:CRS[1] | SRS[1]"/></xsl:when>
        <xsl:when test="count(ancestor::wms:Layer/wms:CRS[.='EPSG:4326'] |
ancestor::Layer/SRS[.='EPSG:4326'])!=0">EPSG:4326</xsl:when>
        <xsl:when test="count(ancestor::wms:Layer/wms:CRS[.='CRS:84'] |
ancestor::Layer/SRS[.='CRS:84'])!=0">CRS:84</xsl:when>
        <xsl:otherwise><xsl:value-of select="ancestor::wms:Layer/wms:CRS[1] |
ancestor::Layer/SRS[1]"/></xsl:otherwise>
    </xsl:choose>
    </xsl:variable>

    <xsl:variable name="bbox">
    <xsl:choose>
    <xsl:when test="($service_name='WMS' and $version='1.3.0' and
$crs='EPSG:4326') or /wfs:WFS_Capabilities">
        <xsl:value-of
select="concat($minY,',',$minX,',',$maxY,',',$maxX)"/></xsl:when>
    <xsl:otherwise>
        <xsl:value-of select="concat($minX,',',$minY,',',$maxX,',',$maxY)"/>
    </xsl:otherwise>
    </xsl:choose>
    </xsl:variable>
```

To create the request it is necessary to join all those parameters in a single full request, as in:

```
    <xsl:variable name="get_request">
       <xsl:value-of select="$operation_online_resource"/><xsl:if
test="substring-
before($operation_online_resource,'?')=''">?</xsl:if><xsl:value-of
select="concat('SERVICE=',$service_name,
'&amp;VERSION=',$version,'&amp;REQUEST=',$default_operation,'&amp;BBOX=',$bbox)
"/><xsl:choose>
           <xsl:when test="$service_name='WFS'"><xsl:value-of
select="concat('&amp;OUTPUTFORMAT=',$data_format,'&amp;TYPENAME=',$name,'&amp;M
AXFEATURES=10')"/>
           </xsl:when>
           <xsl:when test="$service_name='WMS'"><xsl:value-of select="concat(
$crsName, '=',$crs,'&amp;WIDTH=',floor($map_height *
$georatio),'&amp;HEIGHT=',$map_height,'&amp;LAYERS=',$name,'&amp;FORMAT=',$data
_format,'&amp;BGCOLOR=0xffffff&amp;TRANSPARENT=TRUE&amp;EXCEPTIONS=',$exception
_format)"/>
           </xsl:when>
           <xsl:otherwise/>
       </xsl:choose>
    </xsl:variable>
```

Access the full XSLT transformation in the *trax github* site:
https://github.com/Terradue/trax/tree/master/owc/xslt

9 ATOM encoded OWS Context implementations in services

The OWS Context standard conceptually describes a data model which is able to save and distribute the context of a geospatial standard integrated client. An integrated client is a client application that is able to make requests to WMS, WMTS, WFS, WCS and WPS services and show the results in the screen, mainly as a map accompanied by some tables or textual information. For that reason, an OWS Context compliant client is supposed not just to read an OWS Context document but also to save its current status in an OWS Context document, which could then be read by the client itself or by other clients. Nevertheless other services could be able to generate a context document. In fact, we have already demonstrated that an Atom encoding for OWS Context can be used to expose the services available in a WMS or in a WFS.

Particularly, in OWS-9 we have experimented with a CSW catalogue that is able to respond as an OWS Context document. We have also experimented with JavaScript and desktop integrated clients.

9.1 OWS Context CSW server

Catalogue services are mainly intended for data discovery. A common use case is a user that requests datasets that are of his/her interest by sending a GetRecords with a filter to limit the number of results. The result is a collection of metadata records describing datasets. The metadata is usually expressed in ISO 19115 records. Since ISO metadata is quite long and to create metadata records is hard work, the producers are tempted to populate only the mandatory elements that are mainly intended for data discovery. This results in absent or insufficient information about how to access the data. In the described use case, a user can get a long list of "hits" for his/her request, but it is finally the user that has to examine the registries one by one, searching for data access information that is often absent or incomplete, all of which leaves the user with a feeling of deep frustration regarding the consulted server as well as the metadata catalogues in general.

A way to avoid data access frustration is to have a catalogue that instead of giving a response as a collection of metadata records, it responds an OWS Context document. This document will be only composed by the results that have data access information complete enough to include a reference to content or to a service that is able to provide the data.

9.1.1 CSW extension for OWS Context

Based on CSW 2.0.2 standard, we could use:

resultType=results&outputFormat=application/atom+xml&outputSchema=http://www.opengis.net/owc/1.0

The whole idea of Atom encoding is to be compatible with "normal" atom-feed readers. For that reason, we use a generic MIMEtype

The current OWSContext.SWG Atom examples use "http://www.opengis.net/owc/1.0" as an owc namespace.

In OWS-9, Compusult (http://www.compusult.net/) has developed a CSW that is able to respond an OWS Context documents as a result of a CSW query: http://ows-9.compusult.net/wes/serviceManagerCSW/csw

This is the POST payload that has to be sent to the catalogue to get an OWS Context document with the first 10 results:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<csw:GetRecords xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dct="http://purl.org/dc/terms/"
    xmlns:gmd="http://www.isotc211.org/2005/gmd/"
xmlns:gml="http://www.opengis.net/gml"
```

```
    xmlns:ogc="http://www.opengis.net/ogc"
xmlns:ows="http://www.opengis.net/ows"
    xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
    xmlns:wrs="http://www.opengis.net/cat/wrs/1.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" maxRecords="10"
    outputFormat="application/xml" outputSchema="atom"
    resultType="results" service="CSW"
    startPosition="1" version="2.0.2"
xmlns:xml="http://www.w3.org/XML/1998/namespace"
    xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2
http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd
urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0 http://docs.oasis-
open.org/regrep/v3.0/schema/rim.xsd http://www.opengis.net/cat/wrs/1.0
http://schemas.opengis.net/csw/2.0.2/profiles/ebrim/1.0/csw-ebrim.xsd">
    <csw:Query typeNames="csw:Record Service Association">
        <csw:ElementSetName typeNames="csw:Record">full</csw:ElementSetName>
        <csw:Constraint version="1.1.0">
            <ogc:Filter>
                <ogc:And>
                    <ogc:PropertyIsEqualTo>
                        <ogc:PropertyName>Association/@sourceObject
                              </ogc:PropertyName>
                        <ogc:PropertyName>Service/@id</ogc:PropertyName>
                    </ogc:PropertyIsEqualTo>
                    <ogc:PropertyIsEqualTo>
                        <ogc:PropertyName>Association/@targetObject
                              </ogc:PropertyName>
                        <ogc:PropertyName>csw:Record/@id</ogc:PropertyName>
                    </ogc:PropertyIsEqualTo>
                    <ogc:PropertyIsEqualTo>
                        <ogc:PropertyName>Association/@associationType
                              </ogc:PropertyName>
                        <ogc:Literal>
                          urn:ogc:def:ebRIM-AssociationType:OGC:OperatesOn
                        </ogc:Literal>
                    </ogc:PropertyIsEqualTo>
                    <ogc:PropertyIsEqualTo>
                        <ogc:PropertyName>Service/@id</ogc:PropertyName>
                        <ogc:Literal>4df821e7-c80d-4e25-a2c0-ac26ae953ecd
                          </ogc:Literal>
                    </ogc:PropertyIsEqualTo>
                    <ogc:Contains>
                        <ogc:PropertyName>ows:BoundingBox</ogc:PropertyName>
                        <gml:Envelope>
                            <gml:lowerCorner srsName="EPSG:4326">-90.0 -
180.0</gml:lowerCorner>
```

```
                            <gml:upperCorner srsName="EPSG:4326">90.0
180.0</gml:upperCorner>
                        </gml:Envelope>
                    </ogc:Contains>
                </ogc:And>
            </ogc:Filter>
        </csw:Constraint>
        <ogc:SortBy>
            <ogc:SortProperty>
                <ogc:PropertyName>dc:title</ogc:PropertyName>
                <ogc:SortOrder>ASC</ogc:SortOrder>
            </ogc:SortProperty>
        </ogc:SortBy>
    </csw:Query>
</csw:GetRecords>
```

The most important part of the Atom response are the <ows:offering> elements. The ideal response of the catalogue can be composed by 2 <ows:offering>. It is important to get access to the actual data, but it is also of great value to be able to download a full metadada record describing each result. For that reason the final proposed response is to use two offerings: one offering for a CSW request to get a full metadata record for each entry and another offering to get access to a WFS (this is the type of service catalogued in this case) to immediately download the data.

The first offering links to the CSW Capabilities of the catalogue and to a POST request to get only the entry metadata record:

```
        <owc:offering code="http://www.opengis.net/spec/owc/1.0/req/atom/csw">
            <owc:operation code="GetCapabilities" method="GET"
type="application/xml" href="http://ows-
9.compusult.net/wes/serviceManagerCSW/csw?SERVICE=CSW&amp;VERSION=2.0.2&amp;REQ
UEST=GetCapabilities"/>
            <owc:operation code="GetRecords" method="POST"
type="application/xml" href="http://ows-
9.compusult.net/wes/serviceManagerCSW/csw">
                <owc:payload type="application/xml">
                    <csw:GetRecords
xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                        xmlns:gmd="http://www.isotc211.org/2005/gmd/"
xmlns:gml="http://www.opengis.net/gml"
                        xmlns:ogc="http://www.opengis.net/ogc"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
maxRecords="10"
                        outputFormat="application/xml"
outputSchema="http://www.isotc211.org/2005/gmd"
                        resultType="results" service="CSW"
```

```
                        startPosition="1" version="2.0.2"
xmlns:xml="http://www.w3.org/XML/1998/namespace">
                        <csw:Query typeNames="csw:Record Service Association">
                            <csw:ElementSetName
typeNames="csw:Record">full</csw:ElementSetName>
                            <csw:Constraint version="1.1.0">
                                <ogc:Filter>
                                    <ogc:PropertyIsEqualTo>

    <ogc:PropertyName>csw:Record/@id</ogc:PropertyName>
                                        <ogc:Literal>f26a459a-b141-48cd-ab15-
ea5eeaddeea0</ogc:Literal>
                                    </ogc:PropertyIsEqualTo>
                                </ogc:Filter>
                            </csw:Constraint>
                        </csw:Query>
                    </csw:GetRecords>
                </owc:payload>
            </owc:operation>
        </owc:offering>
```

The second offering links to the WFS Capabilities of the catalogue and to a three GET requests to get the data in 3 different formats: two GML versions and one KML.

```
        <owc:offering code="http://www.opengis.net/spec/owc/1.0/req/atom/wfs">
            <owc:operation code="GetCapabilities" method="GET"
type="application/xml" href="http://services.interactive-
instruments.de/xsprojects/ows8-tds/cgi-
bin/ltds/wfs?request=GetCapabilities&amp;service=WFS"/>
            <owc:operation code="GetFeature" method="GET" type="text/xml;
subtype=gml/3.1.1" href="http://services.interactive-
instruments.de/xsprojects/ows8-tds/cgi-
bin/ltds/wfs?request=GetFeature&amp;service=WFS&amp;typename=tds:AircraftHangar
Geopoint&amp;outputFormat=text/xml; subtype=gml/3.1.1&amp;version=1.1.0"/>
            <owc:operation code="GetFeature" method="GET" type="text/xml;
subtype=gml/3.2.1" href="http://services.interactive-
instruments.de/xsprojects/ows8-tds/cgi-
bin/ltds/wfs?request=GetFeature&amp;service=WFS&amp;typename=tds:AircraftHangar
Geopoint&amp;outputFormat=text/xml; subtype=gml/3.2.1&amp;version=1.1.0"/>
            <owc:operation code="GetFeature" method="GET"
type="application/vnd.google-earth.kml+xml" href="http://services.interactive-
instruments.de/xsprojects/ows8-tds/cgi-
bin/ltds/wfs?&amp;request=GetFeature&amp;service=WFS&amp;typename=tds:AircraftH
angarGeopoint&amp;outputFormat=application/vnd.google-
earth.kml+xml&amp;version=1.1.0"/>
```

```
        </owc:offering>
```

This is how a simplified response (with only a single entry response) looks like:

```
<feed xmlns="http://www.w3.org/2005/Atom"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:georss="http://www.georss.org/georss"
xmlns:gml="http://www.opengis.net/gml"
xmlns:owc="http://www.opengis.net/owc/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2005/Atom ../atom/2005/atom.xsd
                    http://purl.org/dc/elements/1.1/ ../../../csw/2.0.2/rec-
dcmes.xsd
                    http://www.georss.org/georss ../georss/1.1/georss.xsd
                    http://www.opengis.net/gml ../georss/1.1/gmlgeorss311.xsd
                    http://www.opengis.net/owc/1.0 ../OWSContextCore.xsd"
xml:lang="en">
    <category scheme="http://www.opengis.net/spec/owc/specReference"
term="http://www.opengis.net/spec/owc/1.0/req/atom" label="This file is
compliant with version 1.0 of OGC Context"/>
    <id>http://ows-9.compusult.net/wes/serviceManagerCSW/csw/0006276a-4f6e-47c1-
94bb-f604245fac57</id>
    <title>Compusult CSW</title>
    <subtitle>An ATOM record version of Compusult's CSW using OWS-
Context</subtitle>
    <generator uri="http://www.compusult.net">Web Enterprise Suite</generator>
    <updated>2013-01-02T15:24:24.446-03:30</updated>
    <entry>
        <id>http://ows-9.compusult.net/wes/serviceManagerCSW/csw/f26a459a-b141-
48cd-ab15-ea5eeaddeea0</id>
        <title>tds:AircraftHangarGeopoint</title>
        <content/>
        <updated>2013-01-02T15:24:24.446-03:30</updated>
        <author>
            <name>interactive-instruments</name>
        </author>
        <dc:publisher>Compusult Limited</dc:publisher>
        <rights>
                        Copyright (c) 2012.  Some rights reserved.  This feed
                        licensed under a Creative Commons Attribution 3.0
License.
                    </rights>
        <georss:where>
            <gml:Envelope>
                <gml:lowerCorner>-90.0 -180.0</gml:lowerCorner>
                <gml:upperCorner>90.0 180.0</gml:upperCorner>
```

```
            </gml:Envelope>
        </georss:where>
        <owc:offering code="http://www.opengis.net/spec/owc/1.0/req/atom/csw">
            <owc:operation code="GetCapabilities" method="GET"
type="application/xml" href="http://ows-
9.compusult.net/wes/serviceManagerCSW/csw?SERVICE=CSW&amp;amp;VERSION=2.0.2&amp;amp;REQ
UEST=GetCapabilities"/>
            <owc:operation code="GetRecords" method="POST"
type="application/xml" href="http://ows-
9.compusult.net/wes/serviceManagerCSW/csw">
                <owc:payload type="application/xml">
                    <csw:GetRecords
xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                        xmlns:gmd="http://www.isotc211.org/2005/gmd/"
xmlns:gml="http://www.opengis.net/gml"
                        xmlns:ogc="http://www.opengis.net/ogc"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
maxRecords="10"
                        outputFormat="application/xml"
outputSchema="http://www.isotc211.org/2005/gmd"
                        resultType="results" service="CSW"
                        startPosition="1" version="2.0.2"
xmlns:xml="http://www.w3.org/XML/1998/namespace">
                        <csw:Query typeNames="csw:Record Service Association">
                            <csw:ElementSetName
typeNames="csw:Record">full</csw:ElementSetName>
                            <csw:Constraint version="1.1.0">
                                <ogc:Filter>
                                    <ogc:PropertyIsEqualTo>

    <ogc:PropertyName>csw:Record/@id</ogc:PropertyName>
                                        <ogc:Literal>f26a459a-b141-48cd-ab15-
ea5eeaddeea0</ogc:Literal>
                                    </ogc:PropertyIsEqualTo>
                                </ogc:Filter>
                            </csw:Constraint>
                        </csw:Query>
                    </csw:GetRecords>
                </owc:payload>
            </owc:operation>
        </owc:offering>
        <owc:offering code="http://www.opengis.net/spec/owc/1.0/req/atom/wfs">
            <owc:operation code="GetCapabilities" method="GET"
type="application/xml" href="http://services.interactive-
instruments.de/xsprojects/ows8-tds/cgi-
bin/ltds/wfs?request=GetCapabilities&amp;amp;service=WFS"/>
```

```
            <owc:operation code="GetFeature" method="GET" type="text/xml;
subtype=gml/3.1.1" href="http://services.interactive-
instruments.de/xsprojects/ows8-tds/cgi-
bin/ltds/wfs?request=GetFeature&amp;service=WFS&amp;typename=tds:AircraftHangar
Geopoint&amp;outputFormat=text/xml; subtype=gml/3.1.1&amp;version=1.1.0"/>
            <owc:operation code="GetFeature" method="GET" type="text/xml;
subtype=gml/3.2.1" href="http://services.interactive-
instruments.de/xsprojects/ows8-tds/cgi-
bin/ltds/wfs?request=GetFeature&amp;service=WFS&amp;typename=tds:AircraftHangar
Geopoint&amp;outputFormat=text/xml; subtype=gml/3.2.1&amp;version=1.1.0"/>
            <owc:operation code="GetFeature" method="GET"
type="application/vnd.google-earth.kml+xml" href="http://services.interactive-
instruments.de/xsprojects/ows8-tds/cgi-
bin/ltds/wfs?&amp;request=GetFeature&amp;service=WFS&amp;typename=tds:AircraftH
angarGeopoint&amp;outputFormat=application/vnd.google-
earth.kml+xml&amp;version=1.1.0"/>
        </owc:offering>
    </entry>
</feed>
```

9.1.2 OWS Context WPS use case

10 Future work

10.1 Annotations

In ISO19117:2005 "Protrayal" defines an annotation as an "any marking on illustrative material for the purpose of clarification". In GMLJP2 OGC 05-047r3 an annotation is an association between an annotation entity (e.g. a text label) and an image or some geometric “region” within the image. In OWS-9 discussions it has been suggested that there is an evolution path from a scribble (with style), to a named scribble, to one named scribble with another attribute or two (which is a feature instance with has an instance style), to finally a feature type / class (with type/class style). It seems that encodings and interfaces that make it easier to work at any of the stages of this path or to transition between stages, would be worth investigating, beyond the issue of how to have a scribble for features, rasters or contexts. To the authors of this ER an annotation is a marking on an illustrative feature overlapped to a map and anchored to a complete geospatial feature or part of it, or even to a spatial region for the purpose of clarification.

This are some examples of annotations expected to be possible:

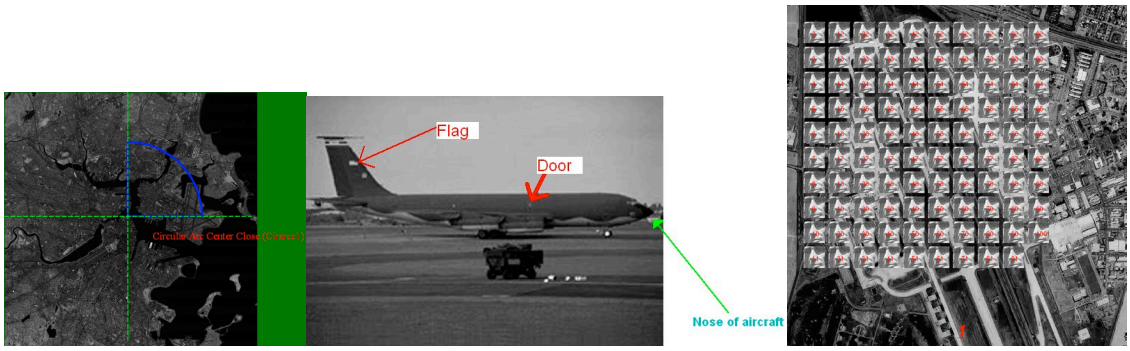


Figure 11 — Imagery Annotation Use Cases

In OWS-9 it was clear that there is a need for an OGC Annotation solution that could be used consistently in different standards such as OWS Context and GMLJP2.

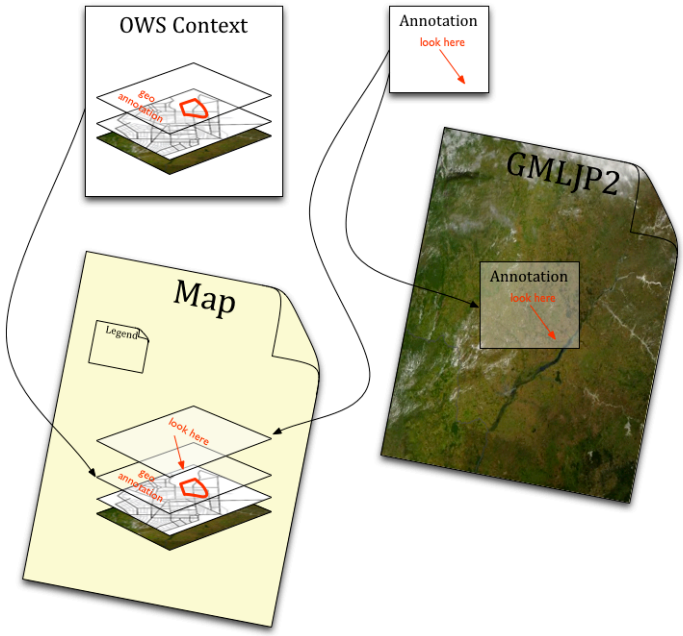


Figure 12 — A conceptual graphic showing how annotation fits with other standards (generated by Raj Singh)

Unfortunately, this topic has not been correctly addressed in the current version of OWS9 and just some initial debates have taken place. There are already some alternatives that have some good and bad capabilities:

Table 12 — Capabilities of some alternatives to annotations

Capability	XIMA	GML with SE	SVG	KML	HTML5 Canvas

Link to features	yes	as an extra geospatial property	No	No	No
Link to geospatial location	CRS	CRS	Display Coord.	WGS84 Display Coord.	
Provenance of annotations (and other metadata)	yes	yes	no	no	no
Renderable in web modern browsers	no	no	yes	no	yes
Renderable in virtual globles	no	no	no	yes	yes

One of the annotation alternatives developed by OGC in the past but unknown to many is the XML for Image and Map Annotations (XIMA), which is an old OGC discussion paper (http://portal.opengeospatial.org/files/?artifact_id=1020) well developed in GML in JPEG 2000 for Geographic Imagery Encoding Specification section 7.6 pages: 15-22 (http://portal.opengeospatial.org/files/?artifact_id=13252). GMLJP2 also includes the file annotation.xsd which is a GML application schema for annotations that follows XIMA.

In XIMA, an annotation has 3 basic elements all expressed in CRS coordinates:

- Pointer: It is a GML line (gml:Curbe, gmlLineString, etc) that links the annotation (called content in XIMA) with a region in space that is being annotated (called annotates in XIMA).

- Content: It is the annotation itself. It can be expressed in two forms. The simple one is a Label that has a text (textContent) and a GML Point with an anchor (anchorPoint). It is also possible to include pictures (Image).

- Annotates: It is the region of the space we are annotating. It can be expressed using any GML geometrical object from a gml:Point to a gml:MultiSurface.

It also includes additional elements such as styling (XIMA uses gml:defaultStyle to include symbology descriptions for the elements in svg form. Unfortunately, defaultStyle is deprecated in the newer versions of GML) and metadata (XIMA permits to include a gml:name, a gml:description and a gml:metaDataProperty to complement your annotation with metadata).

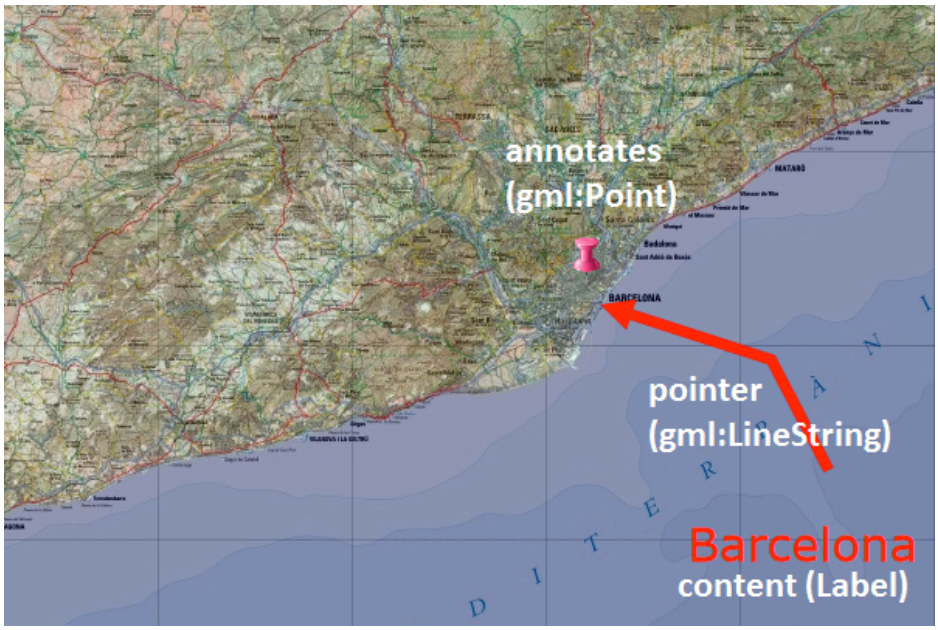


Figure 13 — Annotation elements in an example.

In fact, we experimented with XIMA in OWS Context including it as direct content in an offering:

```
        <owc:offering code="http://www.opengis.net/spec/owc/1.0/req/atom/gml">
            <owc:content type="application/gml+xml">
                <Annotation gml:id="AN_01" xmlns="http://www.opengis.net/xima"
xmlns:xima="http://www.opengis.net/xima"
xmlns:gml="http://www.opengis.net/gml/3.2"
xsi:schemaLocation="http://www.opengis.net/gml/3.2 ../../../gml/3.2.1/gml.xsd
http://www.opengis.net/xima xima.xsd">
                    <pointer>
                        <gml:LineString gml:id="LS_01">
                            <gml:pos>461300.0 4559100.0</gml:pos>
                            <gml:pos>453900.0 4573300.0</gml:pos>
                            <gml:pos>435900.0 4579100.0</gml:pos>
                        </gml:LineString>
                    </pointer>
                    <content>
                        <Label gml:id="LABEL_01">
                            <textContent>Barcelona</textContent>
                            <anchorPoint>
                            <gml:Point gml:id="PT_01">
                                <gml:pos>443100.0 4547300.0</gml:pos>
                            </gml:Point>
                            </anchorPoint>
                        </Label>
```

```
                    </content>
                    <annotates>
                        <gml:Point gml:id="PT_02">
                            <gml:pos>430100.0 4582700.0</gml:pos>
                        </gml:Point>
                    </annotates>
                </Annotation>
            </owc:content>
        </owc:offering>
```

We also developed an XSLT conversion between this kind of XIMA annotation to SVG in order to present this anotations on top of other resources in an HTML5 web browser. While doing so we are imposing some style such as colors, line witdh and arrow tips that were not encoded in the annotation schema.

```
        <xsl:when test="count(owc:offering/owc:content/xima:Annotation)>0">
            <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                <xsl:attribute name="width"><xsl:value-of
select="$mapwidth"/></xsl:attribute>
                <xsl:attribute name="height"><xsl:value-of
select="$mapheight"/></xsl:attribute>
                <defs>
                    <marker id="TriangleRed"
              viewBox="0 0 10 10" refX="0" refY="5"
              markerUnits="strokeWidth"
              markerWidth="4" markerHeight="3"
              orient="auto">
                        <path d="M 0 0 L 10 5 L 0 10 Z"
                            stroke-width="0" fill="red"/>
                    </marker>
                </defs>
                <text font-family="Verdana" font-size="30" fill="red">
                    <xsl:attribute name="x"><xsl:value-of
select="$inv_pixelsize_x*number(substring-
before(owc:offering/owc:content/xima:Annotation/xima:content/xima:Label/xima:an
chorPoint/gml32:Point/gml32:pos, ' '))+$offset_pixel_x"/></xsl:attribute>
                    <xsl:attribute name="y"><xsl:value-of
select="$inv_pixelsize_y*number(substring-
after(owc:offering/owc:content/xima:Annotation/xima:content/xima:Label/xima:anc
horPoint/gml32:Point/gml32:pos, ' '))+$offset_pixel_y"/></xsl:attribute>
                    <xsl:value-of
select="owc:offering/owc:content/xima:Annotation/xima:content/xima:Label/xima:t
extContent"/>
                </text>
                <path
                    fill="none" stroke="red" stroke-width="7"
```

```
                    marker-end="url(#TriangleRed)">
                    <xsl:attribute name="d">
                       <xsl:for-each
select="owc:offering/owc:content/xima:Annotation/xima:pointer/gml32:LineString/
gml32:pos">
                            <xsl:choose>
                                <xsl:when test="position()=1">M </xsl:when>
                                <xsl:otherwise>L </xsl:otherwise>
                            </xsl:choose>
                            <xsl:value-of
select="concat($inv_pixelsize_x*number(substring-before(., '
'))+$offset_pixel_x, ' ' , $inv_pixelsize_y*number(substring-after(., '
'))+$offset_pixel_y, ' ')"/>
                         </xsl:for-each>
                    </xsl:attribute>
                </path>
                <image width="32" height="32" xlink:href="PinPinkIcon.png">
                    <xsl:attribute name="x"><xsl:value-of
select="$inv_pixelsize_x*number(substring-
before(owc:offering/owc:content/xima:Annotation/xima:annotates/gml32:Point/gml3
2:pos, ' '))+$offset_pixel_x - 16"/></xsl:attribute>
                    <xsl:attribute name="y"><xsl:value-of
select="$inv_pixelsize_y*number(substring-
after(owc:offering/owc:content/xima:Annotation/xima:annotates/gml32:Point/gml32
:pos, ' '))+$offset_pixel_y - 32"/></xsl:attribute>
                </image>
            </svg>
        </xsl:when>
```

This small exercice deomestrates that XIMA can be used in an Atom econded OWS Context and that is can easily be transformed into an SVG that can be then rendered in modern web browsers.

10.2 ServiceMetadata documents and OWS Context convergence

In the subclause *8.7 Transform a WMS/WFS Capabilities document into an OWS Context document*, we have exposed the possibility to transform an OGC ServiceMetadata document into an OWS Context format automatically. This conversion can be extended to other services and even consider to include the possibility that an OWS service can respond an OWS Context as part of its capabilities, along with other possibilities such as an ISO 19119 document (possibly adding this as alternatives to the ServiceMetadata document in OWS Common standard).

10.3 GeoPackage manifest and OWS Context convergence

In the subsection *7.4.1 OWS Context package example* we have exposed the similarities of the GeoPackage Manifest and the OWS Context, as well as the possibility that the GeoPackage Manifest can be defined as an extension of the OWS Context.

The convergence of both standards has to be further explored. In OWS9 it has been demonstrated that an OWS Context can be used to generate a GeoPackage with the context referenced in the OWS Context. This relation has to be explored further. If a GeoPackage information has been update while the system was offline, there has to be a way to geo-synchronize the information back to the service. The original OWS Context had to include information on how transactions can be done back to the service as <owc:operation> and the GeoPackage has to inherit this information.

10.4 Packed OWSContext file

There is a risk that the use of direct context both referenced or embedded starts to grow, resulting in files with many external elements or in large OWS Context files. To develop a way to include the OWS Context document and the referenced direct content in a single file could be useful. One possibility could be to adopt the GeoPackage to do that, but another more direct approach is to use ZIP strategies such as the ones used by the ISO Open Packaging Conventions (OPC) standard (ISO/IEC 29500-2:2008 - Information technology -- Document description and processing languages -- Office Open XML File Formats -- Part 2: Open Packaging Conventions, ISO).

10.5 JSON encoding

In the subsection *7.2 JSON encoding* we described the possibility of creating a JSON encoding of the OWS Context Conceptual Model. The OWS SWG decided to postpone this development until the conceptual model (and the Atom encoding has been approved). The authors of this ER understand how easy for a Javascript developer is to get access to a JSON encoding, but concerns exists (expressed before) regarding that almost none OGC service define a JSON encoding for requests and responses, and the fact that a JSON encoding will not be completed with OGC service access. On the other hand, direct content could be expressed in GeoJSON instead of GML or similar. Another concern is about the difficulties that a non-JavaScript client can have to deal with a JSON file. You have to remember that with OWS Context, in principle, there is no service that can generate several formats of it (e.g. by changing the FORMAT parameter). On the contrary, an OWS Context is a document that you get (e.g. form a URL or by email) in a particular format and which you have to deal with your own software: it can be a web map browser, a GIS application, etc.

There is a need for a more careful study about the benefits and limitations of a JSON encoding of the OWS Context.

10.6 HTML5

In the subsection *7.3 HTML5 encoding* we have been discussing some aspects of HTML5 concerning vector visualization. Nevertheless other aspects of HTML5 can be relevant to OWS Context, some being the direct access to raster images and style manipulation on the client side, the local storage techniques that can allow clients to use OWS Context to locally store the client status for a later recovery, among other.

There is a need for reviewing the real advantages of this HTML5 benefits in the OWS context.

10.7 Relations between schemas problem

10.7.1 GML profiles and GeoRSS

The OGC 06-050r3 contains the description of the GeoRSS. The schemas for this specification are available at georss.org. Unfortunately, these schemas are defined as a profile of GML 3.1.1 with the SAME namespace than GML 3.1.1. You can see this in the following schemaLocation attribute:

```
<feed xmlns="http://www.w3.org/2005/Atom"
          xmlns:georss="http://www.georss.org/georss"
          xmlns:gml="http://www.opengis.net/gml"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.w3.org/2005/Atom
                         ../atom/2005/atom.xsd
                    http://www.georss.org/georss
                         http://georss.org/xml/1.1/georss.xsd
                    http://www.opengis.net/gml
                         http://georss.org/xml/1.1/gmlgeorss311.xsd"
          xml:lang="en">
```

The following modification is NOT a solution:

```
<feed xmlns="http://www.w3.org/2005/Atom"
          xmlns:georss="http://www.georss.org/georss"
          xmlns:gml="http://www.opengis.net/gml"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.w3.org/2005/Atom
                         ../atom/2005/atom.xsd
                    http://www.georss.org/georss
                         http://georss.org/xml/1.1/georss.xsd
                    http://www.opengis.net/gml
```

```
                        http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"
        xml:lang="en">
```

This is because the georss.xsd also includes the gml profile.

```
    <xs:import namespace="http://www.opengis.net/gml"
                    schemaLocation="./gmlgeorss311.xsd"/>
```

So it results in more problems.

Additionally, GeoRSS schemas refer to their own copy of xlinks.xsd that is different from the one that is currently used by OGC. OGC recently decided to replace their copy of xlink.xsd with the W3C version (see http://www.opengeospatial.org/blog/1597). Having two different versions of xlink.xsd generates more validation issues.

This problem affects at least to the validation of embedded GML (all versions), and payloads for WFS and CSW. Currently the only solution is to declare the owc:content elements with “lax” validation and to not validate this parts.

```
<any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
```

A solution that the authors of this ER suggest is to make a new version of georss.xsd which would include the complete gml schemas and add Schematron rules to limit the gml elements to the ones permitted by the GeoRSS. This could be done by reviewing the current White Paper document: OGC White Paper OGC 06-050r3: An Introduction to GeoRSS: A Standards Based Approach for Geo-enabling RSS feeds.

10.8 Inclusion of more services: Sensor data extension

One of the obvious missing services that have not been included in the OWS Context extensions is the Sensor Services (the SWE group) and particularly the SOS. There is a need to include them in the OWS Context standard but additional practical work (for instance in OWS10) could be needed.

Annex A

XML Schema Documents

In addition to this document, this report includes several XML Schema Documents. These XML Schema Documents are bundled in a zip file with the present document.

The OWS Context Atom extension described in this document and specified in the OGC 12-084 can be validated with XML Schema Documents (.xsd) and Schematron (.sch) also included in the zip file along with this document. Please, be aware that the intentions of the OWS Context SWG are to release schemas in RelaxNG (.rng) not included in this document. W3C Schemas and Schematron were developed in OWS-9 as a redundant way to validate OWS Context documents in Atom encoding. These XML Schema Documents combine the XML schema fragments listed in various subclauses of this document, eliminating duplications. These XML Schema Documents are named:

owc\1.0\OWSContextCore.xsd

owc\1.0\OWSContextCore.sch

These XML Schema Documents use and build on the Atom XML Schema Documents and GeoRSS specified, named:

owc\1.0\atom\2005\atom.xsd

owc\1.0\atom\2005\atom.sch

owc\1.0\atom\2005\atom.rng

owc\1.0\georss\1.1\ georss.xsd

owc\1.0\georss\1.1\ gmlgeorss311.xsd

All these XML Schema Documents contain documentation on the meaning of each element and attribute.

Some OWS Context examples are provided in the following folder:

owc\1.0\examples

Additionally, examples on how to extend the Atom encoding to describe a GeoPackage Manifest (be aware that the idea of an XML GeoPackage Manifest is still under discussion) is included in the following Schema Documents:

gpkg\1.0\GeoPackageContext.xsd

gpkg\1.0\GeoPackageContext.sch

Some GeoPackage Manifest XML are provided in the following folder:

gpkg\1.0examples

Also an extension about a more complete description on WMTS services is included in the following Schema Documents:

owc\1.0\examples\ owc2html.xsd

An XSLT transformation to generate an HTML simple client are found:

owc\1.0\examples\ owc2html.xsl

It can be tested with the following example:

gpkg\1.0examples\ annotationSimpleSVG_WMTS.xml

Bibliography

[1] Brackin R., P. Gonçalves «OGC OWS Context ATOM Encoding Specification», OGC 12-084. In preparation.

[2] Brackin R., P. Gonçalves, «OGC OWS Context Conceptual Model Specification», OGC 12-080, In preparation.

[3] Cammack R.G. 2007 Open Content Web Mapping Service: A Really Simple Syndication (RSS) Approach Location Based Services and TeleCartography, Lecture Notes in Geoinformation and Cartography2007, pp 417-431

[4] Cote P. 2007 OGC Web Services Architecture for CAD GIS and BIM, Interoperability program report. OGC 07-023r2 Version 1.0.

[5] Döllner J, B Hagedorn Integrating Urban GIS, CAD, and BIM Data By Service-Based Virtual 3D City-Models Urban and Regional Data Management – Coors, Rumor, Fendel & Zlatanova (eds) (c) 2008 Taylor & Francis Group, London ISBN 970-0-415-44059-2 pp -157-170

[6] Fee J.M., DY Venezky Creating Geospatial RSS and ATOM Feeds for Map-based Interfaces. AGU Fall Meeting Abstracts, 2007

[7] ISO 15836:2009 Information and documentation -- The Dublin Core metadata element set

[8] ISO 19115:2003 Geographic information – Metadata

[9] Kyle M., D. Burggraf, S. Forde, R. Lake (2006) GML in JPEG 2000 for Geographic Imagery (GMLJP2) Encoding Specification. Open Geospatial Consortium Inc. OGC 05-047r3. Version: 1.0.0.

[10] Manso-Callejo MA, Manrique-Sancho MT, Abad-Power P (2010) WMS Integrator: continuous access to neighboring WMS Agile 2010.

[11] Mazzetti P, Nativi S and Caron J (2009) RESTful implementation of geospatial services for Earth and Space Science applications. International Journal of Digital Earth, Vol. 2, Supplement 1, 2009, 40-61

[12] Müller M, Kadner D, Bernard L (2012) Moving Code–Sharing Geospatial Computation Logic on the Web. Multidisciplinary Research on Geographical Information in Europe and Beyond. Proceedings of the AGILE'2012 International Conference on Geographic Information Science, Avignon, April, 24-27, 2012.

ISBN: 978-90-816960-0-5 Editors: Jérôme Gensel, Didier Josselin and Danny Vandenbroucke

[13] Nottingham M, and R. Sayre , 2005, The Atom Syndication Format IRTF RFC 4287

[14] Núñez-Redó M, Díaz L, Gil J, González D, Huerta J 2011 Discovery and Integration of Web 2.0 Content into Geospatial Information Infrastructures: A Use Case in Wild Fire Monitoring. Availability, Reliability and Security for Business, Enterprise and Health Information Systems. Lecture Notes in Computer Science. Vol. 6908, pp 50-68

[15] Pons, X., 2004. Manual of MiraMon. Geographic information system and remote sensing software. Bellaterra: Centre de Recerca Ecològica i Aplicacions Forestals (CREAF) ISBN: 84-931323-5-7.

[16] Sai Ma, Minruo Li, Weichang Du, Service Composition for GIS. 2008 IEEE Congress on Services 2008 - Part I

[17] Smolders, S., Alegre, C., Gianfranceschi, S., Gilles, M., Resch, B. & Everding, T. (2011), GENESIS Employing Web Processing Services and Sensor Web Technology for Environmental Management. 14th AGILE International Conference on Graphic Information Science. Utrecht, (NL).

[18] Sonnet J. (2005) Web Map Context Documents, Open Geospatial Consortium Inc. OGC 05-005. Version: 1.1.0.

[19] Wick M, Becker T, 2007 Enhancing RSS Feeds with Extracted Geospatial Information for Further Processing and Visualization. The Geospatial Web. Advanced Information and Knowledge Processing 2007, pp 105-115

