OGC® Testbed-10 Rules for JSON and GeoJSON Adoption:  
Focus on OWS-Context

Copyright © 2014 Open Geospatial Consortium.
To obtain additional rights of use, visit http://www.opengeospatial.org/legal/.

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.
Preface

This work reported in this document is funded through the OGC Testbed-10 Interoperability Testbed public-private partnership and covers activities performed on the Open Mobility Thread. This thread explores the geospatial standards requirements needed to support the growing emerging mobile environment where client applications are mobile, information services are mobile, and increasingly distributed across cloud infrastructures. The work performed in this activity address these requirements while leveraging on the work achieved in the OWS-9 Testbed in the areas of new OWS Context encodings.
License Agreement

Permission is hereby granted by the Open Geospatial Consortium, (“Licensor”), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER’S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR’s sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>1.1</td>
<td>Scope</td>
</tr>
<tr>
<td>1.2</td>
<td>Document contributor contact points</td>
</tr>
<tr>
<td>1.3</td>
<td>Revision history</td>
</tr>
<tr>
<td>1.4</td>
<td>Future work</td>
</tr>
<tr>
<td>1.5</td>
<td>Forward</td>
</tr>
<tr>
<td>2</td>
<td>References</td>
</tr>
<tr>
<td>3</td>
<td>Terms and definitions</td>
</tr>
<tr>
<td>4</td>
<td>Conventions</td>
</tr>
<tr>
<td>4.1</td>
<td>Abbreviated terms</td>
</tr>
<tr>
<td>5</td>
<td>Overview</td>
</tr>
<tr>
<td>6</td>
<td>JSON from XML Encoding Rules</td>
</tr>
<tr>
<td>6.1</td>
<td>XML Elements to JSON Objects</td>
</tr>
<tr>
<td>6.1.1</td>
<td>XML Attributes</td>
</tr>
<tr>
<td>6.1.2</td>
<td>XML Nested Elements</td>
</tr>
<tr>
<td>6.1.3</td>
<td>XML Mixed Elements</td>
</tr>
<tr>
<td>6.1.4</td>
<td>XML Repeated Elements</td>
</tr>
<tr>
<td>6.1.5</td>
<td>Atom Encoding</td>
</tr>
<tr>
<td>6.2</td>
<td>XML Values</td>
</tr>
<tr>
<td>6.2.1</td>
<td>String</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Number</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Boolean</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Empty</td>
</tr>
<tr>
<td>6.2.5</td>
<td>Date</td>
</tr>
<tr>
<td>6.3</td>
<td>Inline XML</td>
</tr>
<tr>
<td>7</td>
<td>XML to GeoJSON Encoding Rules</td>
</tr>
<tr>
<td>7.1</td>
<td>Document root</td>
</tr>
<tr>
<td>7.1.1</td>
<td>Atom Encoding</td>
</tr>
<tr>
<td>7.2</td>
<td>Document Features</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Atom Encoding</td>
</tr>
<tr>
<td>8</td>
<td>Atom OWS Context to GeoJSON Encoding Rules</td>
</tr>
<tr>
<td>Annex A</td>
<td>EXTensible Stylesheet Transformation Files</td>
</tr>
<tr>
<td>A.1</td>
<td>atom2json.xsl</td>
</tr>
<tr>
<td>A.2</td>
<td>owc2geojson.xsl</td>
</tr>
<tr>
<td>Annex B</td>
<td>Examples</td>
</tr>
<tr>
<td>B.1</td>
<td>GeoTIFF Example</td>
</tr>
<tr>
<td>B.2</td>
<td>WPS Example</td>
</tr>
</tbody>
</table>
OGC® Testbed-10 Rules for JSON and GeoJSON Adoption: Focus on OWS-Context

1 Introduction

1.1 Scope

This document identifies the generic rules for obtaining JSON documents directly from existing XML documents and schemas elements. It is primordially targeting the OWS Context JSON Encoding design, but is presented in a generic approach. Such generic approach can offer the guidelines for other OGC services, when defining and using JSON encodings.

1.2 Document contributor contact points

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pedro Gonçalves</td>
<td>Terradue Srl</td>
</tr>
</tbody>
</table>

1.3 Revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Editor</th>
<th>Primary clauses modified</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-01-30</td>
<td>0</td>
<td>Pedro Gonçalves</td>
<td>Definition of transformation rules and GeoJSON encoding</td>
<td>Typos corrections and added future work on JSON-LD</td>
</tr>
<tr>
<td>2014-03-15</td>
<td>1</td>
<td>Pedro Gonçalves</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1.4 Future work

Improvements in this document are desirable to consider the application of JSON-LD (JavaScript Object Notation for Linked Data). JSON-LD is designed around the concept of a "context" to provide additional mappings from JSON to an ontology model. The context links object properties in a JSON document to concept. JSON-LD allows values to be coerced to a specified type or to be tagged with a language. A context can be
embedded directly in the document or put into a separate file and referenced from
different documents (e.g. via a HTTP Link header).

1.5 Forward

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

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

2 References

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

ISO-8601, *Data elements and interchange formats — Information interchange —
Representation of dates and times*, Third edition 2004-12-01, ISO.

OGC 06-121r3, *OGC® Web Services Common Standard*

OGC 12-080, *OWS Context Conceptual Model*

OGC 12-084r2, *OWS Context Atom Encoding*

RFC-4287, *The Atom Syndication Format*

NOTE This OWS Common Standard contains a list of normative references that are also applicable to
this Implementation Standard.

In addition to this document, this report includes several XML and JSON examples files
as specified in Annex E.

3 Terms and definitions

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

4 Conventions

4.1 Abbreviated terms

API Application Program Interface
Overview

This document identifies the generic rules for obtaining JSON documents directly from existing XML documents and schemas elements. The primary requirement for this work activity is the OWS Context JSON Encoding design, but is presented as a generic approach. Such a generic approach can offer the guidelines for defining and using JSON encodings in other OGC services.

Once the XML documents are represented as a JSON objects, their manipulation within Javascript is much facilitated. However it is not possible to make a direct transformation because not everything in XML can be represented in JSON. XML element can contain child XML elements, text and attributes. A JSON objects can contain child JSON objects, numbers, text (double-quoted Unicode escaped strings), Boolean (true or false), ordered arrays and null values.

The mapping of the JSON Boolean, strings and numbers to text is straightforward with a simple mapping to the XSD simple types, and the nesting property of XML is also present in JSON. However there is no way to differentiate the XML child elements from the XML attributes when transforming it to JSON. In XML, attributes are generally considered as a way to convey information that is not a part of the data itself but that provide auxiliary information (e.g. encoding, format). However this is not strictly enforced in XML and furthermore attributes can have the same name of child elements. As such attributes must be considered as integrated part to the element information model.

Another important point to consider is the capability of XML to have child elements with the same name. As this is illegal in JSON it will be necessary to establish rules to the creation of JSON arrays for some elements.

6 JSON from XML Encoding Rules

This section identifies the basic rules that will guide the JSON encoding design. It documents the rules and guidelines to obtain JSON objects from specific data types or directly from XML elements.

6.1 XML Elements to JSON Objects

The transformation from an XML element to a JSON object will be guided by the following rules:

1. The XML element local name is the JSON object name.
2. The XML element single text node is the JSON object value.

3. The XML element attributes nodes are transformed in JSON nested objects (see 6.1.1)

4. The XML nested elements are transformed in JSON nested objects (see 6.1.2)

5. A XML element text node is transformed in a JSON nested object when other types of nodes are present (see 6.1.3)

6. The XML element text value can be casted to a JSON object value type (see 6.2)

7. XML fragments can be transformed in text members (see 6.3)

EXAMPLE XML document transformation in JSON

```
<tree>
  <child other="thing"/>
  <another>my string</another>
</tree>
```

"tree" : {
   "child" : {
       "other" : "thing"
   },
   "another" : "my string",
}

6.1.1 XML Attributes

All XML attributes are added to JSON object in a new name-value pair taking in consideration that:

1. Attribute name is a JSON name.
2. Attribute value is the JSON value (see 6.2)

EXAMPLE Generic XML attributes transformation in JSON

```
<tree att="some">
  <child other="thing"/>
  <branch olive="true"/>
</tree>
```

"tree" : {
   "att" : "some",
   "child" : {
       "other" : "thing"
   },
   "branch" : { 
       "olive" : "true",
   
```
NOTE This rule implies that attributes names are not equal to any of children elements

6.1.2 XML Nested Elements

The nested XML elements are to be transformed directly on nested JSON objects, taking in consideration that:

1. The element name will be the JSON object name.
2. The element attributes are transformed in JSON nested objects (see 6.1.1)
3. The XML nested elements are transformed in JSON nested objects (drill down recurrence)

EXAMPLE Generic XML document transformation in JSON

```xml
<tree att="some">
  <child other="thing"/>
  <another>my string</another>
  <branch olive="true">
    <leaf>green</leaf>
  </branch>
</tree>
```

```
"tree" : {
  "att" : "some",
  "child" : {
    "other" : "thing"
  },
  "another" : "my string",
  "branch" : {
    "olive" : "true",
    "leaf" : "green"
  }
}
```

6.1.3 XML Mixed Elements

In its most basic notation a XML element contains a text node or it contains an ensemble of attribute nodes and other child elements. Another pattern is to define mixed content XML elements where text nodes are together with other child elements.

For this XML mixed element the text node will be transformed into a JSON object where the name is the parent XML element name and the value is the text node contents.

EXAMPLE Mixed content XML document transformation in JSON

```xml
<tree att="some">
  mystring
</tree>
```
6.1.4 XML Repeated Elements

An XML element can have attributes and element nodes with the same name. In this situation all such element should be aggregated in a JSON array. The JSON array name can be changed to the plural of the XML element name when convenient.

EXAMPLE Repeated XML elements transformation in a JSON array

```
<author>
  <name>NSIDC User Services</name>
  <email>nsidc@nsidc.org</email>
  <uri>http://nsidc.org/cgi-bin/atlas_north?</uri>
</author>

<author>
  <name>John Doe</name>
  <email>JohnDoe@example.com</email>
  <uri>http://example.com/~johndoe</uri>
</author>

"authors": [
  {
    "name": "NSIDC User Services",
    "email": "nsidc@nsidc.org",
```
6.1.5 Atom Encoding

When the XML source is encoded as an Atom feed document the following rules shall be observed:

1. All the `atom:authors` elements are added to a “authors” array
2. All the `atom:contributor` elements are added to a “contributors” array
3. All the `atom:category` elements are added to a “categories” array
4. All the `atom:link` elements are added to a “links” array

6.2 XML Values

A JSON value MUST be an object, array, number, or string, or one of the following three literal names: “false”, “null” and “true”.

An object is an unordered collection of zero or more name/value pairs, where a name is a string and a value is a string, number, Boolean, null, object, or array.

There isn’t a direct and unambiguous mapping between XML, JavaScript and JSON data types. Mapping XML data types involves the definition of several rules and best practices. Most data types must be represented by their string or number representation and an agreement has to be established on how to convert between the types.

6.2.1 String

The XML attributes and text nodes can have characters that are unsafe in JSON and Text nodes can also contain multiple lines. In JSON, string need to be in between quotation marks and all special characters need to be escaped.

NOTE The RFC 4627 defines the representation of strings as numbers as:

All Unicode characters may be placed within the quotation marks except for the characters that must be escaped: quotation mark, reverse solidus, and the control characters (U+0000 through U+001F).

EXAMPLE Transformation of XML text nodes in JSON strings

```xml
<text> mystring "true" </text>
```
and then another
</tree>
"text" : " mystring "true"\n and then another \n"

6.2.2 Number

Numbers must be explicitly represented as integer and an optional fraction and exponent. The fraction is a decimal point followed by one or more digits. Leading zeros, octal and hex numbers are not allowed and must be represented as a string. Infinity and NaN must be represented as a string.

NOTE The RFC 4627 defines the representation of numbers as:

The representation of numbers is similar to that used in most programming languages. A number contains an integer component that may be prefixed with an optional minus sign, which may be followed by a fraction part and/or an exponent part. Octal and hex forms are not allowed. Leading zeros are not allowed. A fraction part is a decimal point followed by one or more digits. An exponent part begins with the letter E in upper or lowercase, which may be followed by a plus or minus sign. The E and optional sign are followed by one or more digits. Numeric values that cannot be represented as sequences of digits (such as Infinity and NaN) are not permitted.

6.2.3 Boolean

XML Boolean type defines the values “true” and “1” as true, and as false the values “false” and “0”. If the type of the value is specifically unknown at transformation time then only the “true” and “false” values are mapped directly to the respective JSON value.

EXAMPLE Transformation of XML Boolean values in JSON

<tree value="false">
  <child>true</child>
  <numberorboolean>0</numberofboolean>
</tree>

"tree" : {
  "value" : false,
  "child" : true,
  "numberorboolean" : 0
}

6.2.4 Empty

The XML empty elements must be explicitly transformed to the null JSON object.

EXAMPLE Transformation of XML empty element in JSON

<tree value="false">
  <child/>
</tree>
"tree" : {
    "child" : null
}

6.2.5 Date

The XML and JavaScript date type is not available directly in JSON. Strings, number or any other JSON object can be used to represent a date value. Since the JavaScript version 1.8.5, the most common accepted format is to represent it as a string encoded according to the ISO-8601 standard (YYYY-MM-DDTHH:mm:ss.sssZ). From that version, JavaScript engines and all major browsers support a `Date.toJSON()` method on the JavaScript Date Object.

6.3 Inline XML

In some special cases it is necessary to convey a XML fragment directly in the JSON object without any transformation. The XML should be transformed into an escaped string (see 6.2.1). The contents of the string shall be the same as the XML fragment (escaped but unaltered).

EXAMPLE Transformation of XML fragment in JSON string

```xml
<owc:operation code="GetRecords"
    method="POST"
    href="http://some.net/wes/serviceManagerCSW/csw">
    <owc:request 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">
            <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>9496276a-4f6e-47c1-94bb-f604245fac57</ogc:Literal>
                        </ogc:PropertyIsEqualTo>
                    </ogc:Filter>
                </csw:Constraint>
            </csw:Query>
        </csw:GetRecords>
    </owc:request>
</owc:operation>
```
7 XML to GeoJSON Encoding Rules

GeoJSON is a format for encoding collections of simple geographical features along with their non-spatial attributes using JSON. GeoJSON objects may represent a geometry, a feature, or a collection of features. It supports the following geometry types: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, and
GeometryCollection. Features in GeoJSON contain a geometry object and additional properties, and a feature collection represents a list of features.

This section will provide guidance and rules for writing GeoJSON documents. As in the previous section it was designed with OWS Context in mind but the rules are generic and useful for other information models.

All the rules defined in section 6 apply except when explicitly overridden.

7.1 Document root

The root of the document is a GeoJSON Feature Collection Object. This is a JSON object with a member named “type” with the value equal to “FeatureCollection”. It must have a member called “features” with an array (see section 7.2). If not empty, each member of the array must have a GeoJSON feature. Other possible members are:

1. “crs” - CRS object with a coordinate reference system
2. “properties” – any JSON object or a JSON null value
3. “id” with the universally unique identifier of the feature
4. “bbox” – an array of the box coordinate range of the feature collection
5. “geometry” – GeoJSON geometry object with the spatial coverage of the feature collection

EXAMPLE OWS document root as GeoJSON Feature Collection

```json
{
    "type": "FeatureCollection",
    "id" : "http://ows.acme.eu/385d7d71-b8c7-739e2c0b5e76/",
    "bbox": [-45.0, -24.0, -41.0, -20.0],
    "properties" :
    {
        "lang" : "en",
        "title" : "List of ACME features"
    },
    "features": [
        ...
    ]
}
```

7.1.1 Atom Encoding

When the XML source is encoded as an Atom feed document the following rules shall be observed:

5. The atom:feed/atom:id is mapped to the GeoJSON “id” member
6. The atom:feed/georss:where is mapped to GeoJSON “geometry” member
7. The atom:feed/georss:box is mapped to GeoJSON “bbox” member
8. The \texttt{atom:feed/atom:entry} elements are mapped to GeoJSON “features” member (see section 7.2)
9. All others \texttt{atom:feed} child elements are to be mapped into GeoJSON “properties” members

7.2 Document Features

Each member of the features array must be a GeoJSON Feature and it shall have the following members:

1. “type” with the value of “Feature”
2. “geometry” with any type of GeoJSON Geometry objects
3. “properties” with any JSON object (including nested objects)
4. “id” with the universally unique identifier of the feature

EXAMPLE OWS document root as GeoJSON Feature Collection

```json
{
  "type": "FeatureCollection",
  "id": "http://ows.acme.eu/385d7d71-b8c7-739e2c0b5e76/",
  "bbox": [-45.0, -24.0, -41.0, -20.0],
  "features": [
    {
      "type": "Feature",
      "id": "http://www.acme.net/feature/1",
      "geometry": {
        "type": "Point",
        "coordinates": [-43.184472,-22.936861]
      },
      "properties": {
        "title": "Some Useful Point",
        "updated": "2012-05-10T14:25:01.200Z"
      }
    },
    {
      "type": "Feature",
      "id": "http://www.acme.net/feature/2",
      "geometry": {
        "type": "Point",
        "coordinates": [-44,-21]
      },
      "properties": {
        "title": "Another Some Useful Point",
        "updated": "2012-05-10T14:35:00.400Z"
      }
    }
  ]
}
```
7.2.1 Atom Encoding

When the XML source is encoded as an Atom feed document the following rules shall be observed:

1. The atom:entry/atom:id is mapped to the GeoJSON “id” member
2. The atom:entry/georss:where is mapped to GeoJSON “geometry” member
3. The atom:entry/georss:box is mapped to GeoJSON “bbox” member
4. All others atom:entry child elements are to be mapped into GeoJSON “properties” members

8 Atom OWS Context to GeoJSON Encoding Rules

All the rules defined in section 6 and 7 provide almost all the necessary elements for a correct transformation of an Atom-encoded OWS Context Document. The additional rules that need to be observed are the following:

1. All the owc:offering elements are added to a “offerings” array member of the “properties” object
2. All the owc:operation elements are added to a “operations” array member of the respective “offerings” member.
3. All the owc:content elements are added to a “contents” array member of the respective “offerings” member.

EXAMPLE Fragment of ows:offering elements transformation to GeoJSON

```xml
<owc:offering code="http://www.opengis.net/spec/owc-atom/1.0/reg/geotiff">
  <owc:content type="image/tiff"
</owc:offering>

<owc:offering code="http://www.opengis.net/spec/owc-atom/1.0/reg/wms">
  <owc:operation code="GetCapabilities" method="GET"
    type="application/xml"
    href="http://acme.org/wms?VERSION=1.3.0&REQUEST=GetCapabilities"/>
  <owc:operation code="GetMap" method="GET" type="image/png"
    href="http://acme.org/wms?VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG:4326&BBOX=33.5,-117.3,34.8,-117.8&WIDTH=500&HEIGHT=500&LAYERS=gdal_eg&display=image/png&BGCOLOR=0xffffff&TRANSPARENT=TRUE&EXCEPTIONS=application/vnd.ogc.se_xml"/>
</owc:offering>

"offerings" : ["}
"code" : "http://www.opengis.net/spec/owc-atom/1.0/req/geotiff",
"operations" : [],
"contents" : [{
   "type" : "image/tiff",
   "content" : ""
}]
},

"code" : "http://www.opengis.net/spec/owc-atom/1.0/req/wms",
"operations" : [{
   "code" : "GetCapabilities",
   "method" : "GET",
   "type" : "application/xml",
   "href" : "http://acme.org/wms?VERSION=1.3.0&REQUEST=GetCapabilities",
   "request":{},
   "result":{}
 },{
   "code" : "GetMap",
   "method" : "GET",
   "type" : "image/png",
   "href" : "http://acme.org/wms?VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG:4326&BBOX=33.5,-117.3,34,-117.8&WIDTH=500&HEIGHT=500&LAYERS=gdal_eg&FORMAT=image/png&BGCOLOR=0xffffff&TRANSPARENT=TRUE&EXCEPTIONS=application/vnd.ogc.se_xml",
   "request":{},
   "result":{}
}],

"contents" : []
]
Annex A

EXtensible Stylesheet Transformation Files

A.1 atom2json.xsl

```xml
<?xml version="1.0" encoding="UTF-8" ?>

<xsl:stylesheet version="1.0"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:atom="http://www.w3.org/2005/Atom"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:georss="http://www.georss.org/georss"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:owc="http://www.opengis.net/owc/1.0"
    >
<xsl:output method="text" indent="yes"/>
<xsl:variable name="new-line" select="'
'"/>
<xsl:variable name="tab" select="'    '"/>
<xsl:variable name="quotation" select="''"/>
<xsl:variable name="apostrophe" select="'"/>
<xsl:strip-space elements="*"/>

<xsl:template match="/">
    {<xsl:apply-templates select="atom:feed"/>}</xsl:template>

<xsl:template match="atom:feed">
    <xsl:value-of select="concat($tab,$tab)"/>
    <xsl:apply-templates select="." mode="atomCommonAttributes"/>
    <xsl:for-each select="atom:*[name()!='id' and name()!='author' and name()!='category' and name()!='contributor' and name()!='link' and name()!='entry']">
        <xsl:apply-templates select="."/>
        <xsl:if test="position() &lt; last()">
            ,
        </xsl:if>
    </xsl:for-each>
    <xsl:value-of select="concat($new-line,$tab,$tab)"/>
    "authors" : [<xsl:for-each select="atom:author">
        <xsl:apply-templates select="."/>
        <xsl:if test="position() &lt; last()",
        <xsl:for-each select="atom:*">
            <xsl:value-of select="concat($tab,$tab)"/>
        </xsl:for-each>
    </xsl:for-each>],
    "categories" : [<xsl:for-each select="atom:category">
        <xsl:apply-templates select="."/>
        <xsl:if test="position() &lt; last()",
        <xsl:for-each select="atom:*">
            <xsl:value-of select="concat($tab,$tab)"/>
        </xsl:for-each>
    </xsl:for-each>],
    "links" : [<xsl:for-each select="atom:link">
        <xsl:apply-templates select="."/>
        <xsl:if test="position() &lt; last()",
        <xsl:for-each select="atom:*">
            <xsl:value-of select="concat($tab,$tab)"/>
        </xsl:for-each>
    </xsl:for-each>],
    "entries" : [<xsl:for-each select="atom:entry">
        <xsl:apply-templates select="."/>
        <xsl:if test="position() &lt; last()",
        <xsl:for-each select="atom:*">
            <xsl:value-of select="concat($tab,$tab)"/>
        </xsl:for-each>
    </xsl:for-each>]
```
select="concat($tab,$tab)"/>
</xsl:if>
</xsl:for-each>
</xsl:template>

<xsl:template match="atom:*" mode="atomCommonAttributes">
<xsl:apply-templates select="@xml:base | @xml:lang"/>
</xsl:template>

<xsl:template match="@* | atom:*"><xsl:value-of select="name(.)"/>
</xsl:template>

<xsl:template match="@xml:*"><xsl:value-of select="local-name(.)"/></xsl:template>

<xsl:for-each select="@* | atom:*"><xsl:apply-templates select="."/>
<xsl:if test="position() &lt; last()&gt;&lt;xsl:value-of select="concat($new-line,$tab,$tab,$tab)"/&gt;&lt;xsl:if test="name(../..)='entry'&gt;&lt;xsl:value-of select="$tab"/&gt;&lt;/xsl:if&gt;&lt;/xsl:if&gt;&lt;/xsl:for-each></xsl:template>

<xsl:template match="atom:entry">
<xsl:for-each select="atom:*[name()!='id' and name()!='author' and name()!='category' and name()!='contributor' and name()!='link']">
<xsl:value-of select="concat($new-line,$tab,$tab,$tab)"/>
<xsl:apply-templates select="."/>
<xsl:if test="position() &lt; last()&gt;&lt;xsl:value-of select="concat($new-line,$tab,$tab,$tab)"/&gt;&lt;xsl:if test="name(../..)='entry'&gt;&lt;xsl:value-of select="$tab"/&gt;&lt;/xsl:if&gt;&lt;/xsl:if&gt;&lt;/xsl:for-each>
</xsl:template>

<xsl:template name="safestring">
<xsl:param name="val" select="''"/>
<xsl:call-template name="remove-quote">
<xsl:with-param name="val">
<xsl:call-template name="string-replace-all">
<xsl:with-param name="text" select="$val"/>
<xsl:with-param name="replace" select="''"/>
<xsl:with-param name="by" select="''"/>
</xsl:call-template></xsl:with-param>
</xsl:call-template>
</xsl:template>
A.2  owc2geojson.xsl

<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:atom="http://www.w3.org/2005/Atom"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:georss="http://www.georss.org/georss"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:owc="http://www.opengis.net/owc/1.0"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:exsl="http://exslt.org/common"/>

<xsl:template name="remove-quote">
    <xsl:param name="val" select="''"/>
    <xsl:call-template name="return-value">
        <xsl:with-param name="val">
            <xsl:call-template name="string-replace-all">
                <xsl:with-param name="text" select="$val"/>
                <xsl:with-param name="replace" select="'"/
                <xsl:with-param name="by" select="'\'"/>
            </xsl:call-template>
        </xsl:with-param>
    </xsl:call-template>
</xsl:template>

<xsl:template name="return-value">
    <xsl:param name="val" select="''"/>
    <xsl:value-of select="normalize-space($val)"/>
</xsl:template>

<xsl:template name="string-replace-all">
    <xsl:param name="text"/>
    <xsl:param name="replace"/>
    <xsl:param name="by"/>
    <xsl:choose>
        <xsl:when test="contains($text,$replace)"/>
            <xsl:value-of select="substring-before($text,$replace)"/>
            <xsl:value-of select="$by"/>
        <xsl:call-template name="string-replace-all">
            <xsl:with-param name="text" select="substring-after($text,$replace)"/>
            <xsl:with-param name="replace" select="$replace"/>
            <xsl:with-param name="by" select="$by"/>
        </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
        <xsl:value-of select="$text"/>
    </xsl:otherwise>
</xsl:choose>
</xsl:template>

</xsl:stylesheet>
<xsl:import href="atom2json.xsl"/>
<xsl:include href="xml-to-string.xsl"/>
<xsl:output method="text" indent="yes"/>

<!-- GeoRSS to GeoJson transformation block -->
<xsl:template name="gml2coord">
  <xsl:param name="string" select="."/>
  <xsl:variable name="other" select="substring-after(substring-after($string,' '),', ')"/>
  <xsl:variable name="point" select="(" />
  <xsl:variable name="other" select="substring-after(substring-after($string,' '),', ')"/>
  <xsl:variable name="point" select="(" />
  <xsl:if test="$other!=''">
    <xsl:call-template name="gml2coord"><xsl:with-param name="string" select="$other"/></xsl:call-template>
  </xsl:if>
</xsl:template>

<xsl:template match="gml:Point | georss:point">
  "type" : "Point",
  "coordinates" : 
  <xsl:call-template name="gml2coord"><xsl:with-param name="string" select="."/></xsl:call-template>
</xsl:template>

<xsl:template match="gml:LineString | georss:Line">
  "type" : "LineString",
  "coordinates" : 
  <xsl:call-template name="gml2coord"><xsl:with-param name="string" select="."/></xsl:call-template>
</xsl:template>

<xsl:template match="gml:Polygon | georss:polygon">
  "type" : "Polygon",
  "coordinates" : 
  <xsl:call-template name="gml2coord"><xsl:with-param name="string" select="."/></xsl:call-template>
</xsl:template>

<xsl:template match="gml:Envelope">
  <xsl:variable name="x1" select="substring-after(gml:lowerCorner,',' )"/>
  <xsl:variable name="x2" select="substring-after(gml:upperCorner,',' )"/>
  <xsl:variable name="y1" select="substring-before(gml:lowerCorner,'')"/>
  <xsl:variable name="y2" select="substring-before(gml:upperCorner,'')"/>
  "type" : "Polygon",
  "coordinates" : 
  <xsl:value-of select="concat('[', $x1, '],', $y1, ',']"/>
  <xsl:value-of select="concat('[', $x1, '],', $y2, ',']"/>
  <xsl:value-of select="concat('[', $x2, '],', $y2, ',']"/>
  <xsl:value-of select="concat('[', $x2, '],', $y1, ',']"/>
</xsl:template>

<xsl:template match="georss:box">
  <xsl:variable name="x1" select="substring-before(substring-after(.,',''),'')"/>
  <xsl:variable name="x2" select="substring-after(substring-after(.,','),'')"/>
</xsl:template>
"type": "Polygon",
"coordinates": [[
<xsl:value-of select="concat('[', $x1, ',', $y1, ']' )"
>,
<xsl:value-of select="concat('[', $x1, ',', $y2, ']' )"
>,
<xsl:value-of select="concat('[', $x2, ',', $y2, ']' )"
>,
<xsl:value-of select="concat('[', $x2, ',', $y1, ']' )"
>,
<xsl:value-of select="concat('[', $x1, ',', $y1, ']' )"
>]]
</xsl:template>

<xsl:template match="georss:where">
<xsl:apply-templates select="*"/>
</xsl:template>

<!-- GeoRSS to GeoJson transformation block -->

<!-- override of atom:feed as features are outside properties -->

<xsl:template match="atom:feed">
  "type": "FeatureCollection",
  "id": "<xsl:value-of select="atom:id"/>",
  "geometry": {
    "coordinates": [
      <xsl:apply-templates select="georss:*[1]"/>
    ]
  },
  "properties": {
    "authors": [
      <xsl:for-each select="atom:author">
        <xsl:apply-templates select="*"/>
      </xsl:for-each>
    ],
    "contributors": [
      <xsl:for-each select="atom:contributor">
        <xsl:apply-templates select="*"/>
      </xsl:for-each>
    ],
    "categories": [
      <xsl:for-each select="atom:category">
        <xsl:apply-templates select="*"/>
      </xsl:for-each>
    ],
    "links": [
      <xsl:for-each select="atom:link">
        <xsl:apply-templates select="*"/>
      </xsl:for-each>
    ],
    "features": [
      <xsl:for-each select="atom:entry">
        <xsl:apply-templates select="*"/>
      </xsl:for-each>
    ]
  }
</xsl:template>
<!-- override extension atom:entry for -->
<xsl:template match="atom:entry">
  "type": "Feature",
  "id": "<xsl:value-of select="atom:id"/>",
  "geometry": {<xsl:apply-templates
    select="(georss:*|../georss:*)[last()]"/>
  },
  "properties": {<xsl:apply-imports/>
    "offerings" : [<xsl:for-each select="owc:offering">
      <xsl:apply-templates select="."/>
    </xsl:for-each>
  ]
}</xsl:template>

<xsl:template match="atom:id"/>

<xsl:template match="owc:offering">
  <xsl:for-each select="@*"/>
  <xsl:value-of select="concat($tab,""")"/>
  "operations" : [<xsl:for-each select="owc:operation">
    <xsl:value-of select="concat($tab,$tab,$tab,$tab)"/>
    <xsl:if test="position() &lt; last()">
      ,
    </xsl:if></xsl:for-each>],
  "contents" : [<xsl:for-each select="owc:content">
    <xsl:apply-templates select="."/>
    <xsl:value-of select="concat($new-line,$tab,$tab,$tab,$tab)"/>
  </xsl:for-each>]</xsl:template>

<xsl:template match="owc:operation"><xsl:for-each select="@*"/>
  <xsl:value-of select="concat($new-line,$tab,$tab,$tab,$tab)"/>
  <xsl:if test="position() &lt; last()">
    ,
  </xsl:if></xsl:for-each>
  "request": {<xsl:apply-templates select="owc:request"/>},
  "result": {<xsl:apply-templates
    select="owc:result"/>
</xsl:template>

<xsl:template match="owc:request / owc:result ">
  <xsl:for-each select="@*"/>
  <xsl:value-of select="concat($new-line,$tab,$tab,$tab,$tab,$tab)"/>
  <xsl:for-each>
    <xsl:variable name="myNode" select="*[1]"/>
    <xsl:variable name="nodeAsStr" select="safestring($myNode)"/>
    <xsl:call-template name="safestring">
      <xsl:with-param name="val" select="$nodeAsStr"/>
    </xsl:call-template>
  </xsl:for-each>
  " contents": "<xsl:value-of select="local-name()"/>
  "<xsl:value-of select="$nodeAsStr"/>
</xsl:template>

<xsl:template match="owc:content">
Annex B

Examples

B.1 GeoTIFF Example

```json
{
  "type": "FeatureCollection",
  "id": "http://www.opengis.net/owc/1.0/examples/geotiff",
  "geometry": {},
  "properties": {
    "lang": "en",
    "title": "GeoTIFF Example",
    "subtitle": "GeoTIFF Example",
    "updated": "2012-11-04T17:26:23Z",
    "authors": [{"name": "Joan Maso"},
      "contributors": [{"name": "Pedro Goncalves"}],
    "categories": [],
    "links": [{
      "rel": "profile",
      "href": "http://www.opengis.net/spec/owc-
        atom/1.0/req/core",
      "title": "This file is compliant with version 1.0 of
        OGC Context"
    }],
  },
  "features": [{
    "type": "Feature",
    "id": "ftp://ftp.remotesensing.org/pub/geotiff/samples/gdal_eg/cea.txt",
    "geometry": {
      "type": "Polygon",
      "coordinates": [[-117.30874,33.66497],[-117.30874,33.94383],[-117.60838,33.94383],[-117.60838,33.66497],[-117.30874,33.66497]],
    "properties": {
      "title": "GeoTIFF Example",
      "updated": "2011-11-01T00:00:00Z",
      "content": "GeoTIFF Example coming from ftp://ftp.remotesensing.org/pub/geotiff/samples/gdal_eg",
      "authors": [],
      "contributors": [],
      "categories": [],
      "links": [{"rel": "enclosure",
        "type": "image/tiff",
```
B.2 WPS Example

{  "type": "FeatureCollection",

  "href": "ftp://ftp.remotesensing.org/pub/geotiff/samples/gdal_eg/ce
da.tif",
  "title": "GeoTiff Example from gdal_eg",
  "length": "270993"},
  "offerings": [{
    "code": "http://www.opengis.net/spec/owc-
    atom/1.0/req/geotiff",
    "operations": [],
    "contents": [{
      "type": "image/tiff",
      "href": "ftp://ftp.remotesensing.org/pub/geotiff/samples/gdal_eg/ce
da.tif",
      "content": ""
    }]
  },{
    "code": "http://www.opengis.net/spec/owc-
    atom/1.0/req/wms",
    "operations": [{
      "code": "GetCapabilities",
      "method": "GET",
      "type": "application/xml",
      "href": "http://acme.org/wms?VERSION=1.3.0&REQUEST=GetCapabilities",
      "request":{},
      "result":{}
    },{
      "code": "GetMap",
      "method": "GET",
      "type": "image/png",
      "href": "http://acme.org/wms?VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG:
      4326&BBOX=33.5,-117.3,34,-117.8&WIDTH=500&HEIGHT=500&LAYERS=gdal_eg&FORMAT=image/png&BGCOLOR=0xffffff&TRANSPARENT=TRUE&EXCEPTIONS=application/vnd.ogc.se_xml",
      "request":{},
      "result":{}
    }],
    "contents": []
  }]
}
"id": "http://www.opengis.net/owc/1.0/examples/wps_52north",
"geometry": {},
"properties": {
  "lang": "en",
  "title": "WPS 52North example",
  "subtitle": "WPS 52North example",
  "updated": "2012-11-04T17:26:23Z",
  "authors": [{"name": "Joan Mas\v{z}"}],
  "contributors": [],
  "categories": [],
  "links": [{
    "rel": "profile",
    "href": "http://www.opengis.net/spec/owc-atom/1.0/req/core",
    "title": "This file is compliant with version 1.0 of OGC Context"
  }]
},
"features": [{
  "type": "Feature",
  "id": "http://geoprocessing.demo.52north.org:8080/wps/WebProcessingService",
  "geometry": {},
  "properties": {
    "title": "WPS 52 north",
    "updated": "2013-05-19T00:00:00Z",
    "content": "WPS 52North",
    "authors": [],
    "contributors": [],
    "categories": [],
    "links": [{"rel": "via",
      "type": "application/xml",
      "href": "http://www.opengis.uab.cat/wms/satcat/metadades/EPSG_23031/Cat_20110301.htm",
      "title": "HMTL metadata in Catalan (nothing to do with this WPS. Sorry!)"}],
    "offerings": [{
      "code": "http://www.opengis.net/spec/owc-atom/1.0/req/wps",
      "operations": [{
        "code": "GetCapabilities",
        "method": "GET",
        "type": "application/xml",
        "href": "http://geoprocessing.demo.52north.org:8080/wps/WebProcessingService?SERVICE=WPS&VERSION=1.0.0&REQUEST=GetCapabilities",
        "request":{}
      }]
    }]
  }
}]}
"result":{},
  },{
    "code" : "DescribeProcess",
    "method" : "GET",
    "type" : "application/xml",
    "request":{},
    "result":{}
  },{
    "code" : "Execute",
    "method" : "POST",
    "href" : "http://geoprocessing.demo.52north.org:8080/wps/WebProcessingService?",
    "request":{
      "type" : "text/xml",
      "request" : "<wps:Execute service=\"WPS\" version=\"1.0.0\"
        xsi:schemaLocation="http://www.opengis.net/wps/1.0.0 http://schemas.opengis.net/wps/1.0.0/wpsExecute_request.xsd"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:ows="http://www.opengis.net/ows/1.1"
        xmlns:wps="http://www.opengis.net/wps/1.0.0">
        <ows:Identifier xmlns:ows="http://www.opengis.net/ows/1.1">org.n52.wps.server.algorithm.SimpleBufferAlgorithm</ows:Identifier>
        <wps:DataInputs xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:ows="http://www.opengis.net/ows/1.1"
        xmlns:wps="http://www.opengis.net/wps/1.0.0">
          <wps:Input xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:ows="http://www.opengis.net/ows/1.1"
        xmlns:wps="http://www.opengis.net/wps/1.0.0">
            <ows:Identifier xmlns:ows="http://www.opengis.net/ows/1.1">data</ows:Identifier>
            <wps:Reference schema="http://schemas.opengis.net/gml/3.1.1/base/feature.xsd"
        xlink:href="http://geoprocessing.demo.52north.org:8080/geoserver/wfs?SERVICE=WFS&amp;VERSION=1.0.0&amp;REQUEST=GetFeature&amp;TYPENAME=topp:tasmania_roads&amp;SRS=EPSG:4326&amp;OUTPUTFORMAT=GML3"
        method="GET"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:ows="http://www.opengis.net/ows/1.1"
        xmlns:wps="http://www.opengis.net/wps/1.0.0">
        <ows:Identifier xmlns:ows="http://www.opengis.net/ows/1.1">data</ows:Identifier>
        </wps:Reference>
        <wps:Input xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:ows="http://www.opengis.net/ows/1.1"
        xmlns:wps="http://www.opengis.net/wps/1.0.0">
            <ows:Identifier xmlns:ows="http://www.opengis.net/ows/1.1">width</ows:Identifier>
            <wps:Data xmlns:wps="http://www.opengis.net/wps/1.0.0">
                <wps:LiteralData dataType="xs:double">
                    0.05</wps:LiteralData>
            </wps:Data>
        </wps:Input>
        </wps:DataInputs>
    </wps:Execute>
  }
}
Example 2
Bibliography

[1] TBD