OGC® DOCUMENT: 16-080

External identifier of this OGC® document: http://www.opengis.net/doc/PER/t12-A063

OGC®
Making location count.

TESTBED-12 OWS CONTEXT USER GUIDE

USER GUIDE

PUBLISHED

Submission Date: 2017-06-29
Approval Date: 2017-06-29
Publication Date: 2017-01-23
Editor: Roger Brackin

OGC®
Making location count.

License Agreement

Copyright notice

Note

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.

CONTENTS

LIST OF FIGURES

I SECURITY CONSIDERATIONS

No security considerations have been made for this document.

II INTRODUCTION

While over the last 10 years use of open standards compliant geospatial web services such as WMS, WMTS, WFS, WCS and CSW has grown massively, there has not been an effective open standards mechanism to assemble a collection of information such that it can be exchanged easily among a community. While some GIS vendors support such a capability the solution is typically not portable between vendors. The OGC did develop the Web Map Context standard (05-005) but this only supported WMS and so was relatively limited. The OWS Context Standard (OWC) was developed to allow advanced sharing of geospatial 'Context' or 'view' between users across a wide range of application types (browser, desktop applications, apps) and technologies (desktop, mobile, embedded etc). The standard is modular and easily implemented, and supports a wide range of OGC service standards as well as in-line or local content.

OWS Context offers a solution to a wide range of sharing requirements. Chapter 1 describes a list of use case examples. Chapter 2 will provide a detailed description of each of these examples.

II.A. Usage Example — Shared Situational Awareness

Users in a range of environments have access to the central services, but typically wish to collaborate using a range of information. In a collaboration built on for example an emergency response there is typically a wide range of stakeholders and a need to provide them with relevant common information.

II.B. Usage Example — Sharing Catalogue Searches

A users wishes to save or exchange the query and results from various catalogue searches to avoid duplication of effort. In this case a user would execute one or more searches and retain each search request and possibly results set and be able to exchange the search and results set with others so that they could review the results or modify and re-execute the search.

II.C. Usage Example — Processing Services Management

A user wishes to save and/or exchange the configuration and/or results of an analysis or processing activity. The process to be executed will be stored in the context document as well as inline or referenced results.

II.D. Usage Example — Index or Catalogue of Data Collections

A user wishes to create a collection of datasets, for example OGC GeoPackages which are split geospatially (tiled) and/or based on themes. The OWS Context document can provide a catalogue or inventory of these.

II.E. Benefits of OWS Context Documents

Some of the key benefits of OWS Context are:

- OWC is open standards compliant allowing sharing between different system technologies.
- OWC documents are easily created, exchanged, catalogued and read.
- OWC documents can be encoded in XML/ATOM or GeoJSON. The two formats are interchangable.
- OWC supports not only WMS but a wide range of other OGC Services (CSW, WFS, WCS, WPS, WMTS etc)
- OWC Supports inline content in well known encodings such as GML and KML.
- OWS Context Documents can reference local files in for example GeoTIFF.
- OWC is easily extensible to include new service types, inline content and file = formats (OGC and non-OGC) without the need for a change in the standard.

An OWS Context Document primarily contains references to information and so is itself relatively small in size. It therefore is easy to exchange and access.

1

NORMATIVE REFERENCES

1 NORMATIVE REFERENCES

The following normative references are used in this document.

Roger Brackin, Pedro Gonçalves: OGC 12-080r2, *OGC OWS Context Conceptual Model*. Open Geospatial Consortium (2014). http://www.opengis.net/doc/IS/owc-conceptual/1.0.

Roger Brackin, Pedro Gonçalves: OGC 12-084r2, *OGC OWS Context Atom Encoding Standard*. Open Geospatial Consortium (2014). http://www.opengis.net/doc/IS/ows-context-atom/1.0.

Pedro Gonçalves, Roger Brackin: OGC 14-055r2, *OGC OWS Context GeoJSON Encoding Standard*. Open Geospatial Consortium (2017). http://www.opengis.net/doc/IS/ows-context-geojson/1.0.

OGC CDB Core Standard. This document has now been approved as an OGC Standard but has yet to be published to the OGC Standards registry.

GeoRSS Geographically Encoded Objects for RSS Feeds. (http://www.georss.org/)

2

COMMON USE CASES

2 COMMON USE CASES

The Context Document goal is to support the exchange of a set of information elements between human users or application programs for a range of purposes such that the Area of Interest, time range, resources and their configuration is unambiguously exchanged between applications.

2.1. Use Cases

The OWS Context document is aimed at meeting a range of user context exchange requirements around sharing information. This section provides more detail on the primary use cases described in section 1 including a definition ofcommon sub use-cases.

2.1.1. Shared Situational Awareness/COP Exchange

Users in a range of environments have access to the central services, but typically wish to collaborate using a shared set of information of information. In a collaboration built on for example an emergency response there is typically a wide range of stakeholders and a need to provide them with relevant common information. Often one person or group (typically geo-support) has the responsibility to provide a set of information to other users (in this case commanders and on-scene responders) in time sensitive situations to allow them to deal with the evolving situation; given that they do not have time to assemble information themselves. The OWS Context Document standard allows the geo-support personnel to assemble a set of relevant information, accessible via web services, and pass it to other users (by email message, file transfer or by storing it within an OGC catalogue).

2.1.1.1. On-Line COP Use Case

The exchange of a common operating picture or view of information is recognized as one of the most important uses of the OWS Context Document. Often a COP defines a geographic Area of Interest and a set of information layers and queries to specific points of interest. The content of a COP may be made up of both web services and local content, and the OWS Context document allows this to be supported. It also allows the inclusion of alternatives for a given layers.

A critical element of a shared situational awareness environment, whether used in a military, civil incident management or any other situational awareness context is the ability to exchange COPs between systems based on different technology. There are many equivilent approaches to OWS Context, but none which offer a vendor agnostic approach. This hetrogeneous model means COPs can be shared in a number of environments, for example between command centers and

deployed users in vehicles or on foot, and using everything from laptops in mobile vehicles through to tablets, mobile devices or wearable technology (see Figure 1).

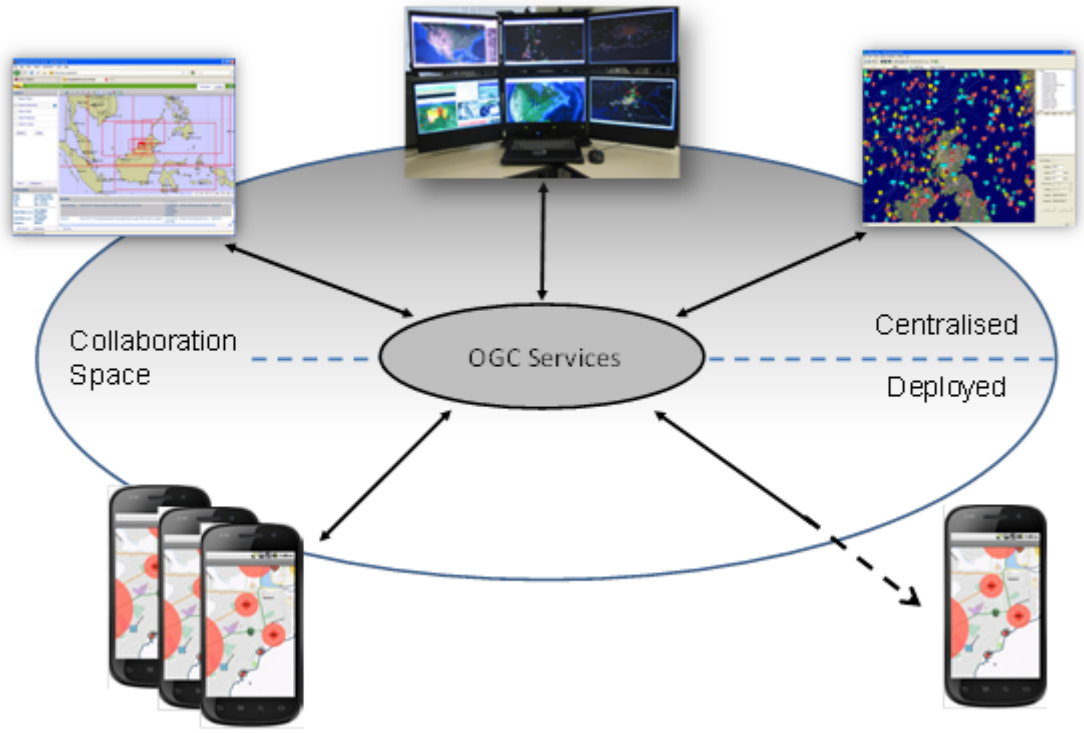

Figure 1 — OWS Context SSA Use Case

A COP may take many forms and have many uses, it may represent a 2D map display, a 3D map display, an augmented reality view, or a textual representation of geospatial information.

The COP exchanged is not simply a graphic but a 'live' view. When loaded it should show the latest state of the information referenced. This is a capability of OWS Context.

In this basic COP use case, OWS Contexts are simply exchanged between systems using a file system or via a messaging system such as email. More advanced exchanges are possible via a catalogue where users can identify groups of context documents relevant to them directly or incidentially. In addition it is possible that users could be notified on new contexts using a subscription scheme (potentially based on the OGC PubSub Standard).

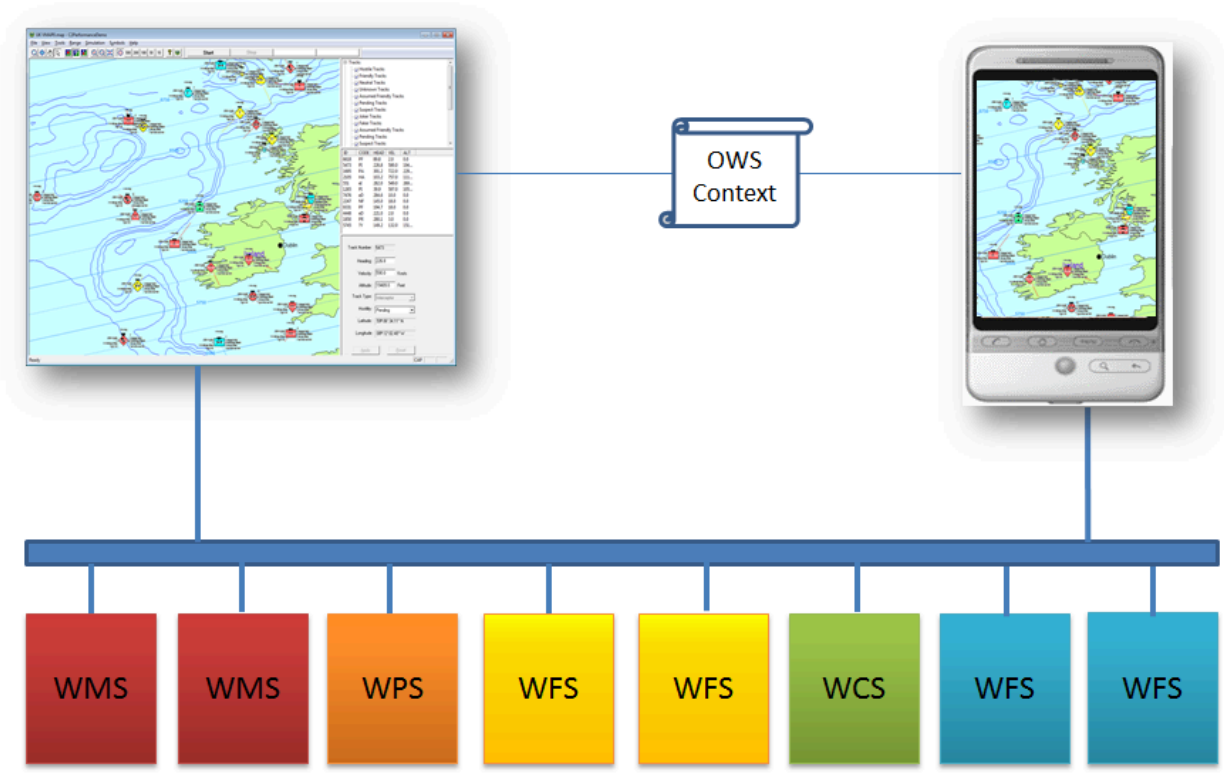

Figure 2 — OWS Context SSA Use Case

2.1.1.2. On-Line COP Exchange Via Catalogue

OGC Compliant catalogue services provide a useful way to manage OWS Context Documents.

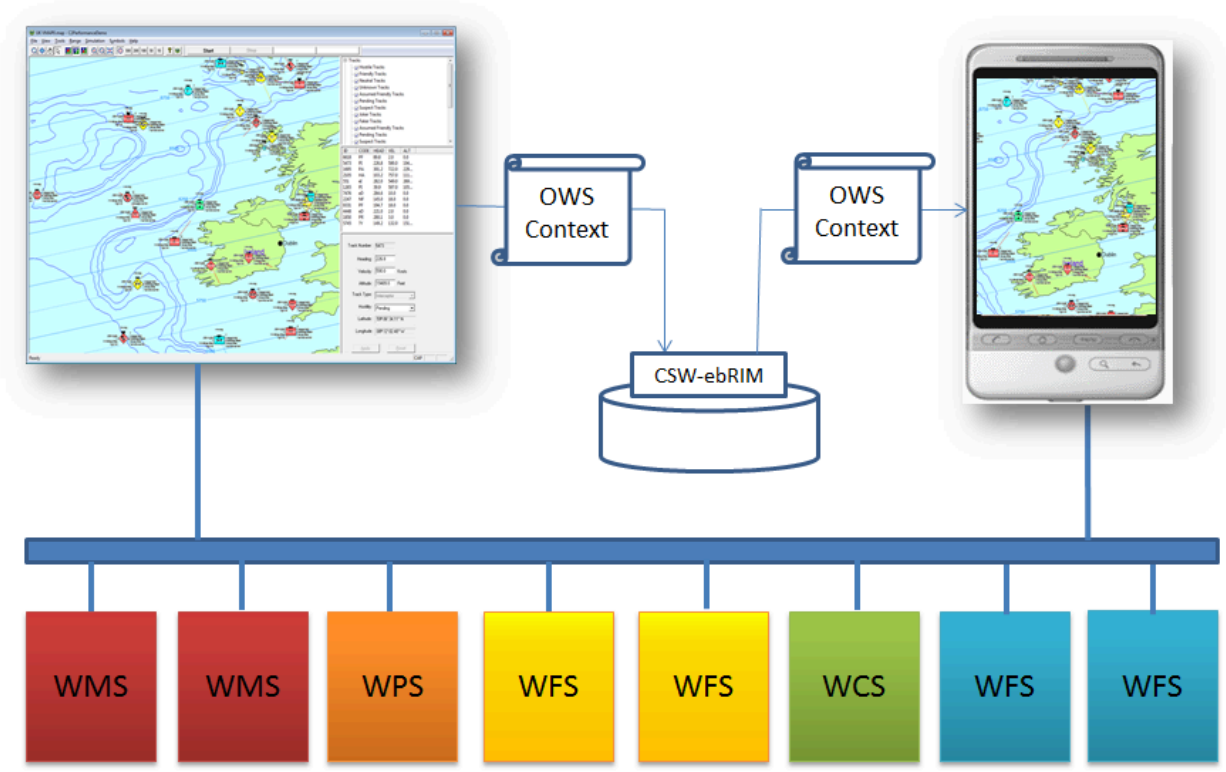

Figure 3 — OWS Context Catalogue Storage Use Case

2.1.1.3. Off-Line COP Exchange

As part of this there is the recognition that in some cases the services referenced may not be available, and therefore some information (for example overlays or thumbnails) may need to be carried in the context document itself or alongside it. The context document layers may need to support both an on-line and an off-line alternative. One delivery model for a range of geospatial information is the GeoPackage standard. One specific model is to use the OWS Context document to provide a defintion of the services to be harvested into a GeoPackage so that the content can then be used off-line. Either a WPS service (GeoPackager) or the client itself can use the context document to load the GeoPackage with data and potentially update a GeoPackage which was already created. The context document allows the layers (WMS or WMTS) and feature types (WFS) required to be defined but also the geo-extent of data to be harvested.

Of course GeoPackage is not the only carrier encoding that could be used in this way, and evolving standards such as CDB [Ref 4] could potentially be created from an OWS Context defintion. Alternatively completely proprietary caches could be created on a client, once again using OWS Context to define the content of a cache. The attractive element of this overall workflow is that a client can see what will be cached using the normal on-line access, before executing the process to cache the data.

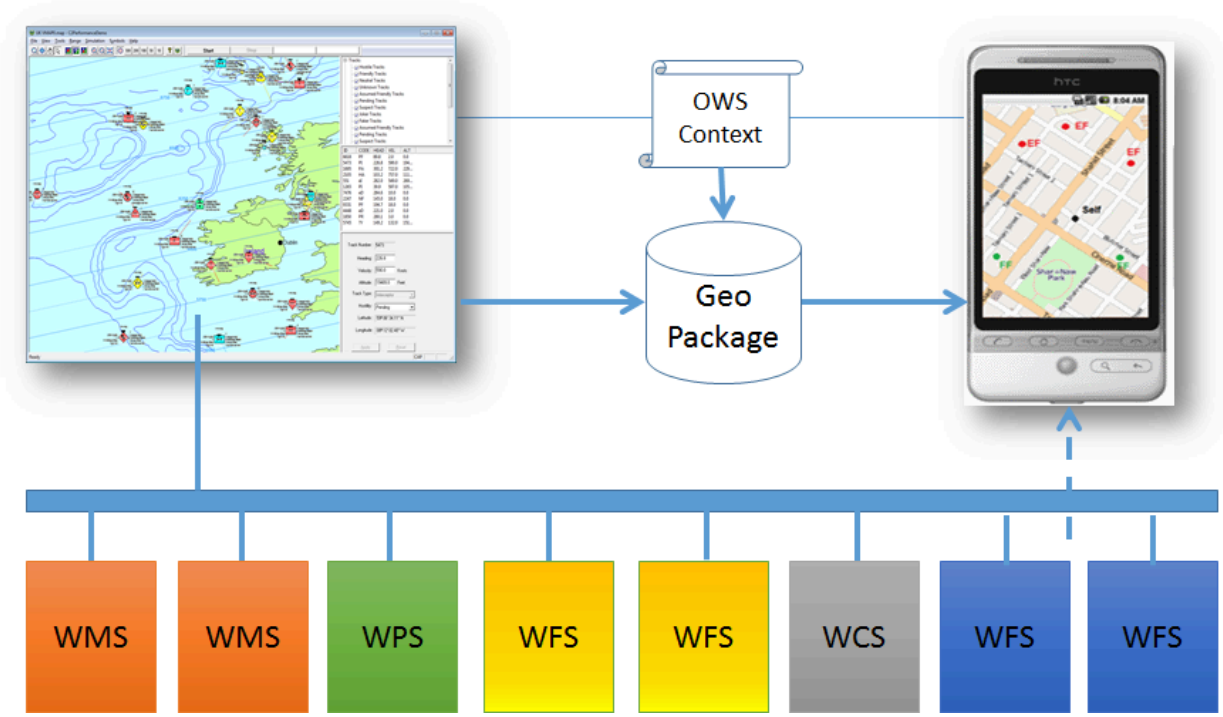

Figure 4 — OWS Context Off-line Use Case

2.1.2. Exchange of a Catalogue Query and Results

This use case relates to the exchange of discovery results from various catalogue searches, to avoid duplication of effort. In this case a user would execute one or more searches and retain each search request and possibly results set and be able to exchange the search and results set with others so that they could review the results or modify and re-execute the search. To do this primarily CSW requests and optionally results will be stored in the context document. This allows the retrieved results to be reviewed but also allows the query used to obtain the results to be re-run or modified and rerun.

2.1.3. Exchange of the Configuration and Results of a Process

A user wishes to save and/or exchange the configuration and/or results of an analysis or processing activity. The process to be executed will be stored in the context document as well as inline or referenced results. This type of context document might use WPS to define the processing, with results returned by the WPS, inline in the Context document in GML, in a referenced image, or available via another OGC service (WMS, WMTS etc).

2.2. Requirements

The above use cases lead to the following general requirements for an OWS Context document.

- The Context Document shall provide the capability to identify the temporal extent of the COP (one or more time envelopes). (See Chapter 5 — Time Interval of Interest Metadata)
- The Context Document shall provide the capability to define a series of configured resources together which provide relevant information to the COP User. (See Chapter 5 — Resources)
- The Context Document shall define the order of precedence of the resources included (this could be interpreted as, for example, the order of display by visualization clients) (See Chapter 5 — Ordering of Resources)
- The Context Document shall allow any service type to be specified and any rules to be specified. (See Chapter 5 — Offering)
- The Context Document shall provide information to allow clients to test if a service matches a supported profile in order to understand if they can interpret it. (See Chapter 5 — Telling if an XML/GeoJSON Document is a Context Document)
- The Context Document shall allow information targeted at different representations to be included (i.e. not just targeted at geographic visualization or just visualization). (See Chapter 6 — Terradue — Using OWS Context to store Searches)
- The Context Document shall allow information to be marked as enabled or disabled, i.e. it is to be presented to the user when the context is opened or isn't. Source: WMC Specification: which has and on/off option (layer not displayed or not when loaded). (See Chapter 5 — Visibility Attribute)
- It shall be possible to associate information with embedded graphics information. (See Chapter 5 — Content Offerings)
- The Context Document should allow the in-line inclusion of a resource (literal value of a resource) in the context document. (See Chapter 5 — Content Offerings)
- The Context Document should allow the parameters which define the resource or processing service steps to be captured. (See Chapter 5 — Envitia_TB12_OWC.xml and Envitia_TB12_OWC.json)

3

THE OWS CONTEXT DOCUMENT STRUCTURE

3 THE OWS CONTEXT DOCUMENT STRUCTURE

An OWS Context (OWC) document consists of a number of key elements. These are shown in Figure 5.

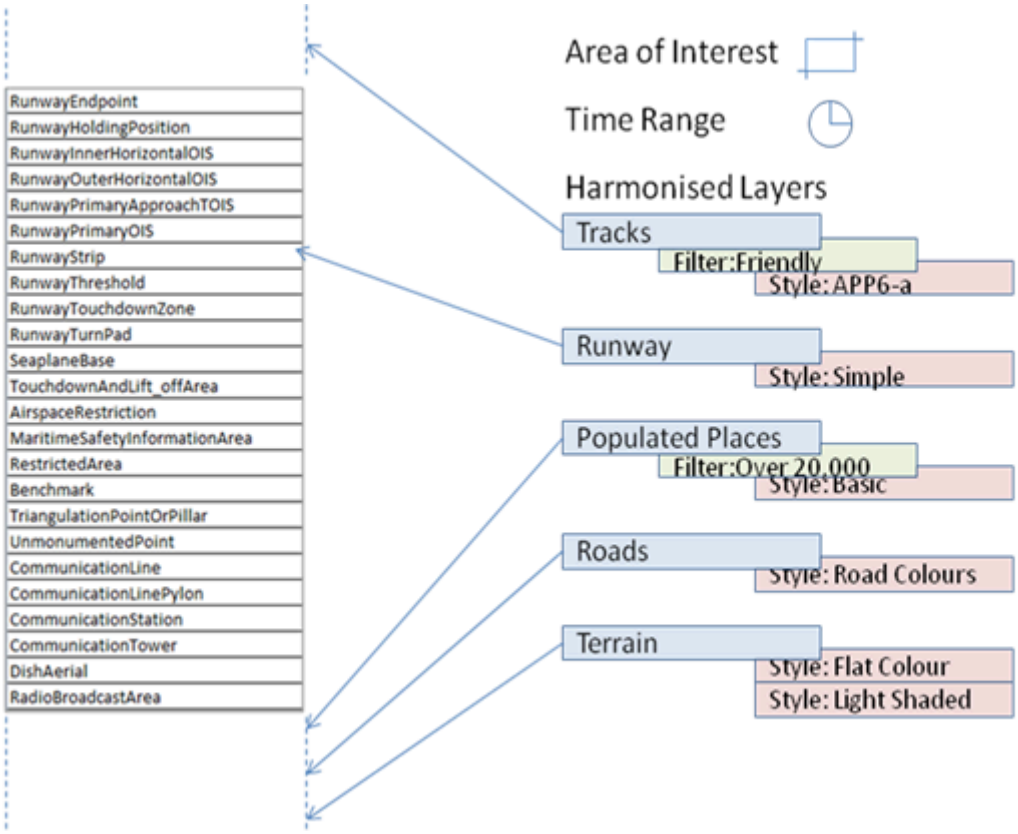

Figure 5 — Structure of OWS Context Document

The overall information includes a name, abstract and creation date as well as a range of high level metadata. It also includes an Area of Interest and a time range of interest (all optional). When using OWC for visualization the Area of Interest is typically the area displayed on screen when the context document is loaded. Similarly the time-range of interest is the time range that any time slider in the application will be set to. The core of the OWC document is an ordered list of resources. Again for visualization purposes the application should load the resources in the list such that the first resource is at the top (i.e. reverse draw order).

Each layer in a context document is known as a 'Resource' which in the Atom encoding is mapped to a 'Entry' XML element. In reality a resource can be realised in a number of different ways, and so an OWC document allows various options to be specified. These are known as offerings. The intention is that these are, as far as is possible by the format used, equivalent and no priority is assigned to their order in the standard. They are intended to be alternatives that the client can use to allow it to visualise or use the resource. This structure is shown in Figure 6.

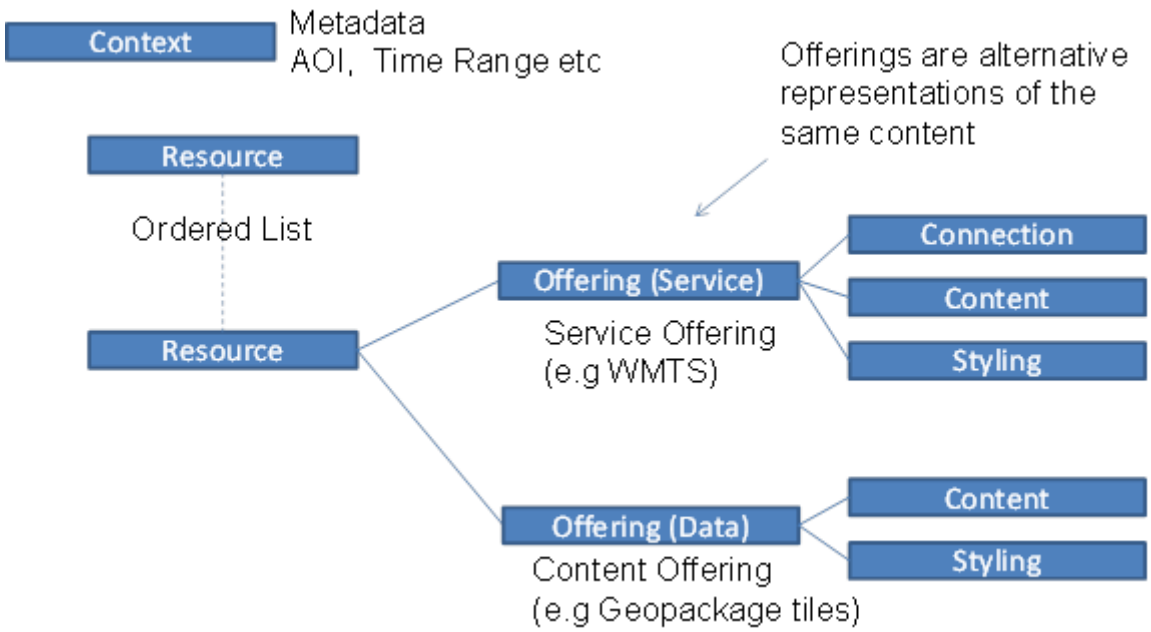

Figure 6 — OWS Context Document Offering

An example of two equivalent offerings is a WMS for a resource and then a WMTS request retrieving equivalent data. The client can read either and be reasonably certain that the result presented to the user is the same. Another example is a resource which might offer a WFS and also in-line GML which is the equivalent of the request.

There are two types of offering, service offerings and data offerings. A service offering has a service request (in the form of a capabilities request and a data request) and optional content and styling elements. A data offering has a content element and optional styling elements.

4

OWS CONTEXT FOR USERS

4 OWS CONTEXT FOR USERS

The goal of this section is to give end users of OWS Context documents and idea or how they can be used. To do this a two existing applications are exploited to show the typical workflow. As can be seen in Clause 6 (Applications and Tools) there are a number of other applications which support OWS context with a range of use cases, so the following example is purely one method of exploiting it.

The example in this section use the Envitia Horizon Web Client which was used during testbed 12. The functionality described is for guidance only. OWS Context does not mandate client behavior, and other implentors may design the functionality in a different way.

4.1. Building the View to be Shared

Firstly the user uses a browser based client to create an OWS Context Document. This process follows a fairly normal GIS Workflow.

4.1.1. Catalogue Search

The creation of most geospatial views begins with a discovery process. Typically geospatial information users define an area of interest, and categories of information they are interested and perform some form of search. The figure below shows an example of just such a search. In this case the user was looking for data in the area of ??? and after executing the search can see a range of data sources.

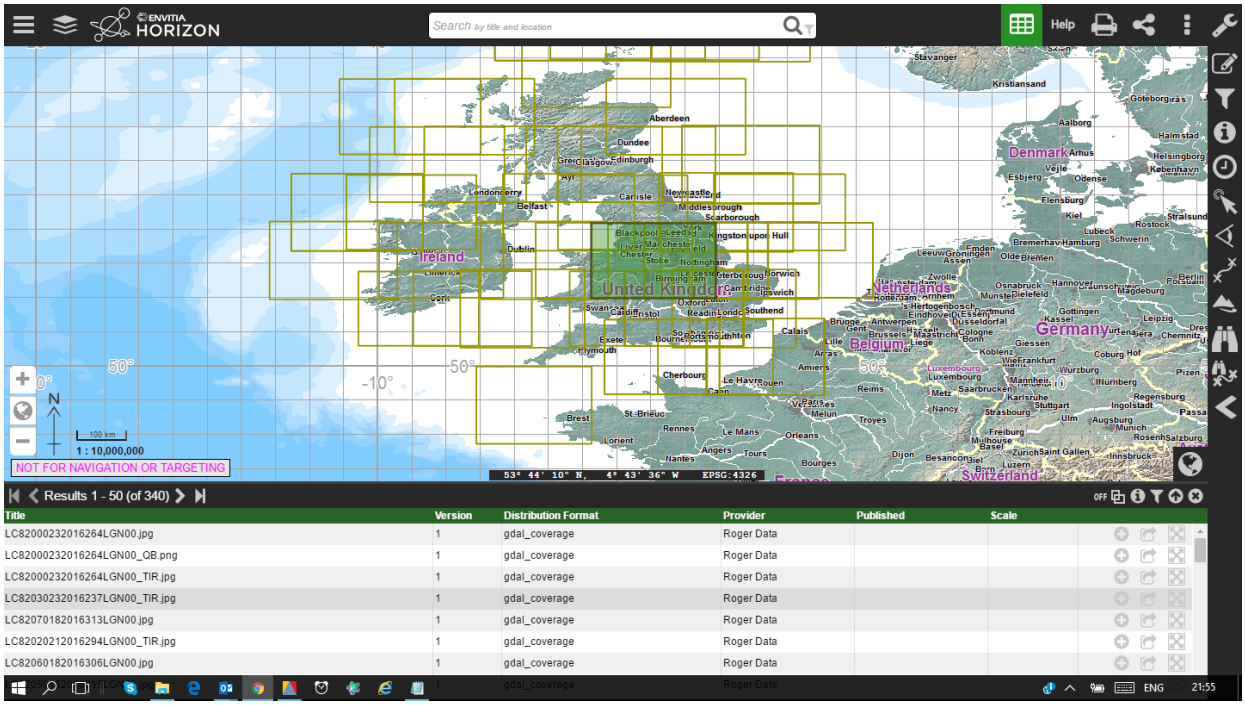

Figure 7 — Web Client Searching for Data

4.1.2. Adding information from WMS Services.

Each returned dataset may be available as different service types. For example a rivers layer may be available via a WMS service or a WFS service. The user can see these various options in the Metadata and select to add one.As a shortcut, in this case the client gives the user a shortcut button to add the WMS if its present. The result is a layer added on top of the default background map in the client. This process is completed for a number of layers until the user is happy with the content. An example of this is shown in Figure 8.

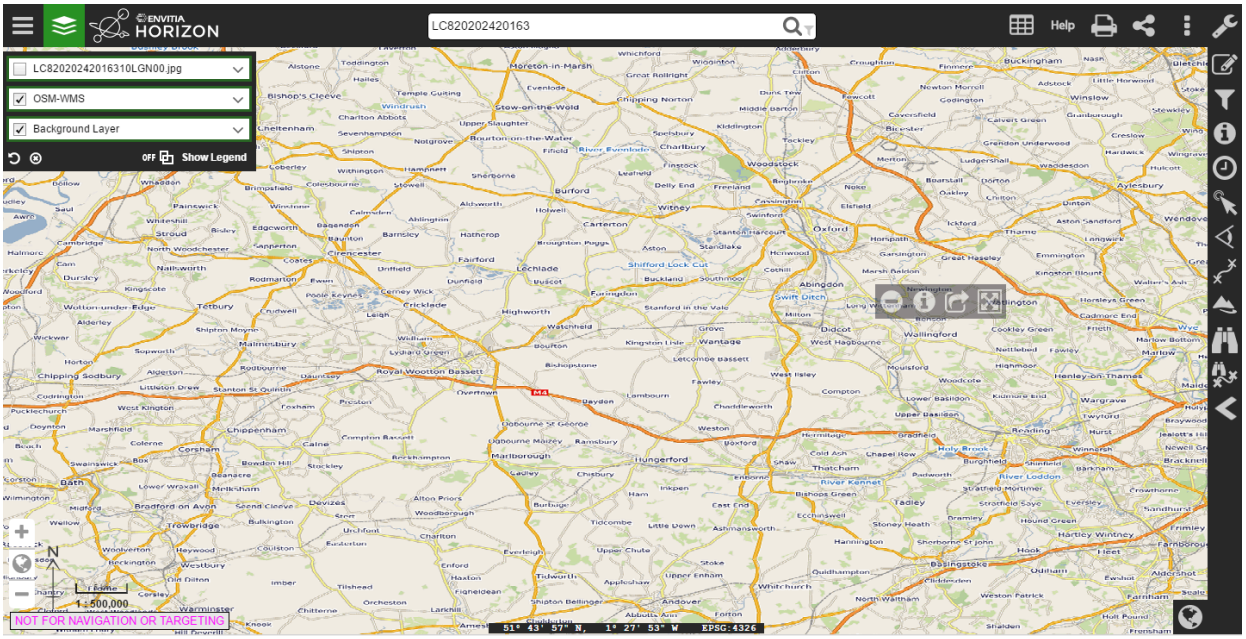

Figure 8 — Client Showing Two added Layers

4.1.3. Setting Layer Visibility

It may be a large number of layers are included in a Context Document as the creator wishes to pass on to the client the best range of data available. But there is a desire to keep the view simple for the user recieving the document. A good way of providing a rich set of infomation but limiting the clutter, is to include a number of additional layers but set their visibility off. In the example in Figure 8 the imagery layer (second from the top in the list to the left) is set to off and therefore the imagery is not visible. The context document will be exported with this setting.

4.1.4. Adding Annotation

A common requirement is also to provide the receiving user with some annotation related to the view. In Horizon users have the ability to create as many annotation layers as they wish, but in this case we create a single one and add some annotation.

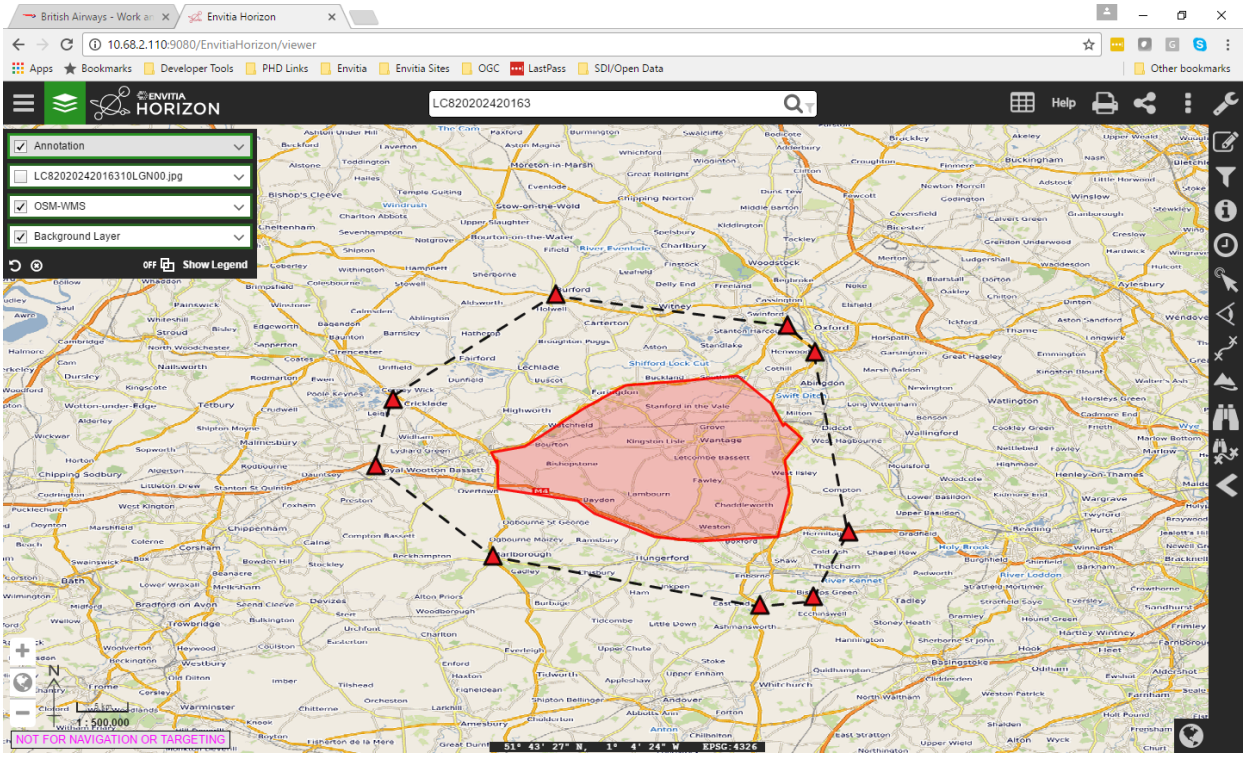

Figure 9 — Adding Annotation to the View

4.1.5. Saving The Context Document

The next step is to Save the Document, Horizon includes the capability to save context documents. The user can envoke the 'Save Context' option and then the dialog is displayed.

In the Horizon Client the context document can be saved to an OGC CSW-ebRIM compliant Catalogue service (also known as a Registry) or to the local file system on the client.

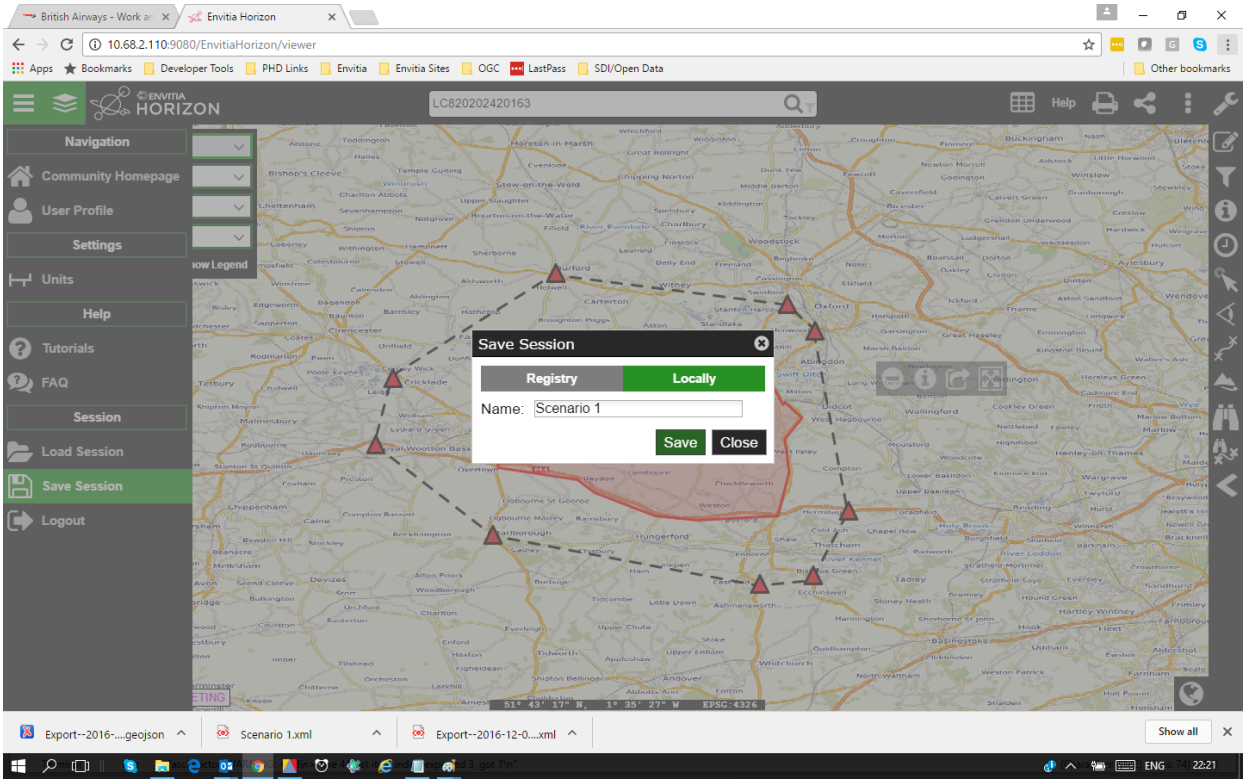

Figure 10 — Selecting Location to Save Context

The dialog allows the layers loaded to be output in the Context Document. The layers in the client are listed and the user can pick which layers are included.

In addition the user can select whether an area of interest is set (this can be toggled or or off). If set the context will be output with a bounding box in the context document. The other key option in the dialog is the ability to select which encoding of OWS Context is used. The options are ATOM/XML and GeoJSON. The choice really depends on the level of support in the client the document will be sent to, i.e whether it supports the ATOM/XML or GeoJSON encoding.

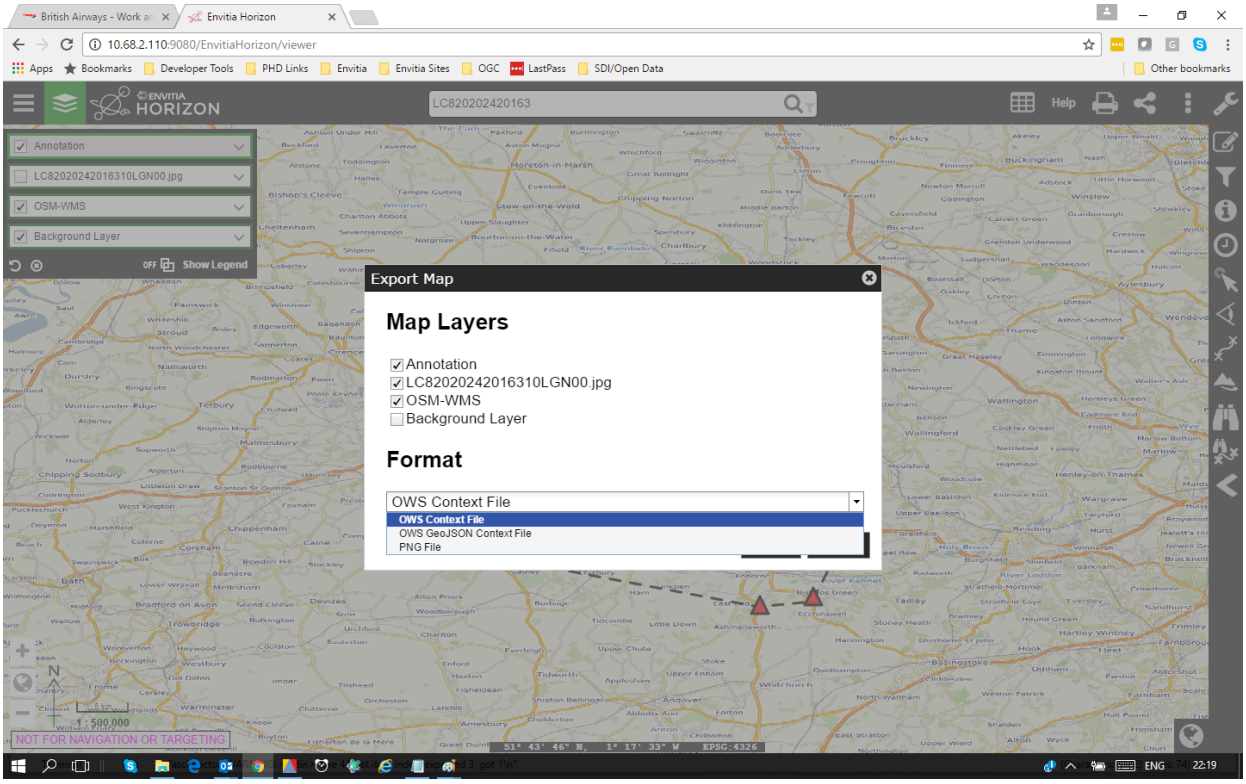

Figure 11 — Selecting Location to Save Context

4.1.6. Exchanging the Context Document

OWS Context documents are simple text endcodings in either XML or JSON. As a result they can be sent as email attachments without zipping or converting them.

The recieving user can simply save the document to a local file system ready to be loaded.

4.1.7. Loading the Context Document

Envitia Horizon can then be used to read the OWS Context document. The user uses the load function in the Web Client. Firstly the client is opened.

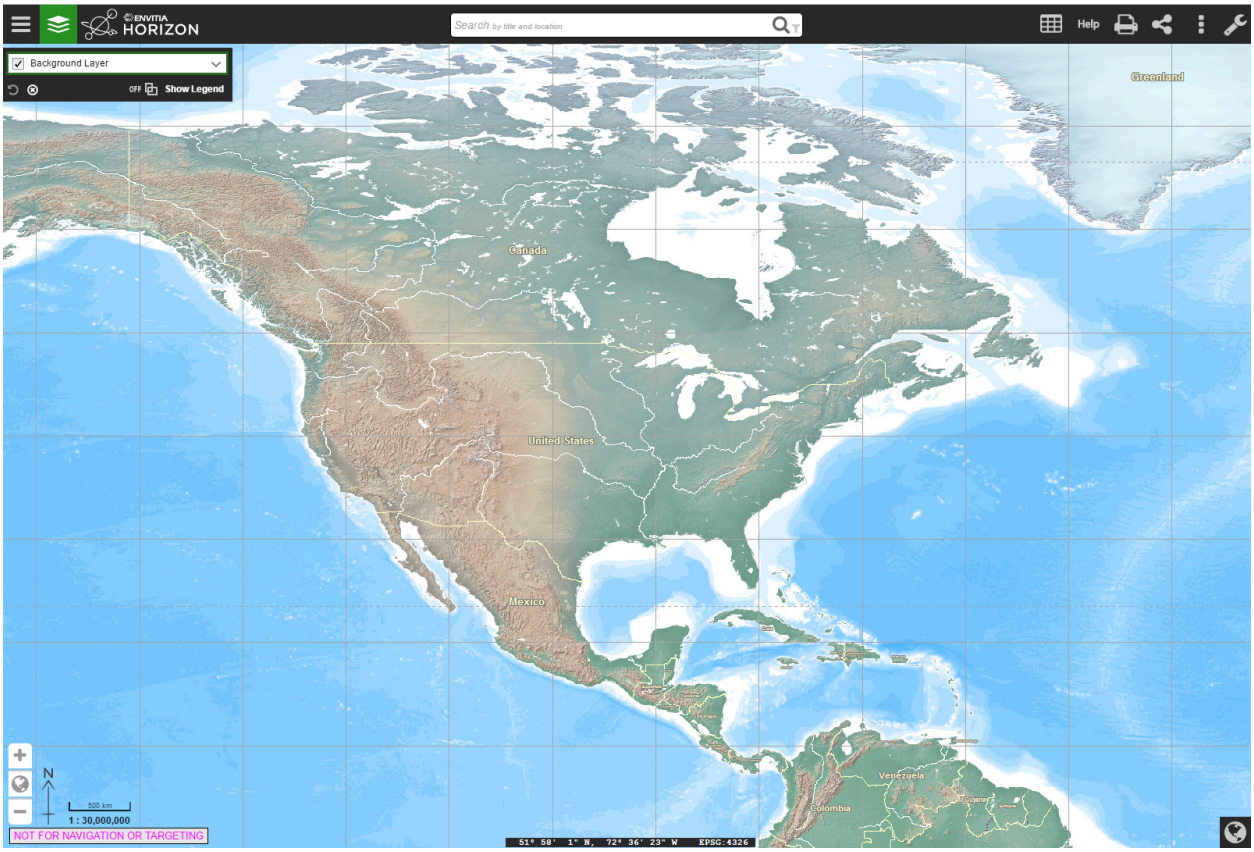

Figure 12 — Open Envitia Horizon

Th import function allows a document to be picked from the file system.

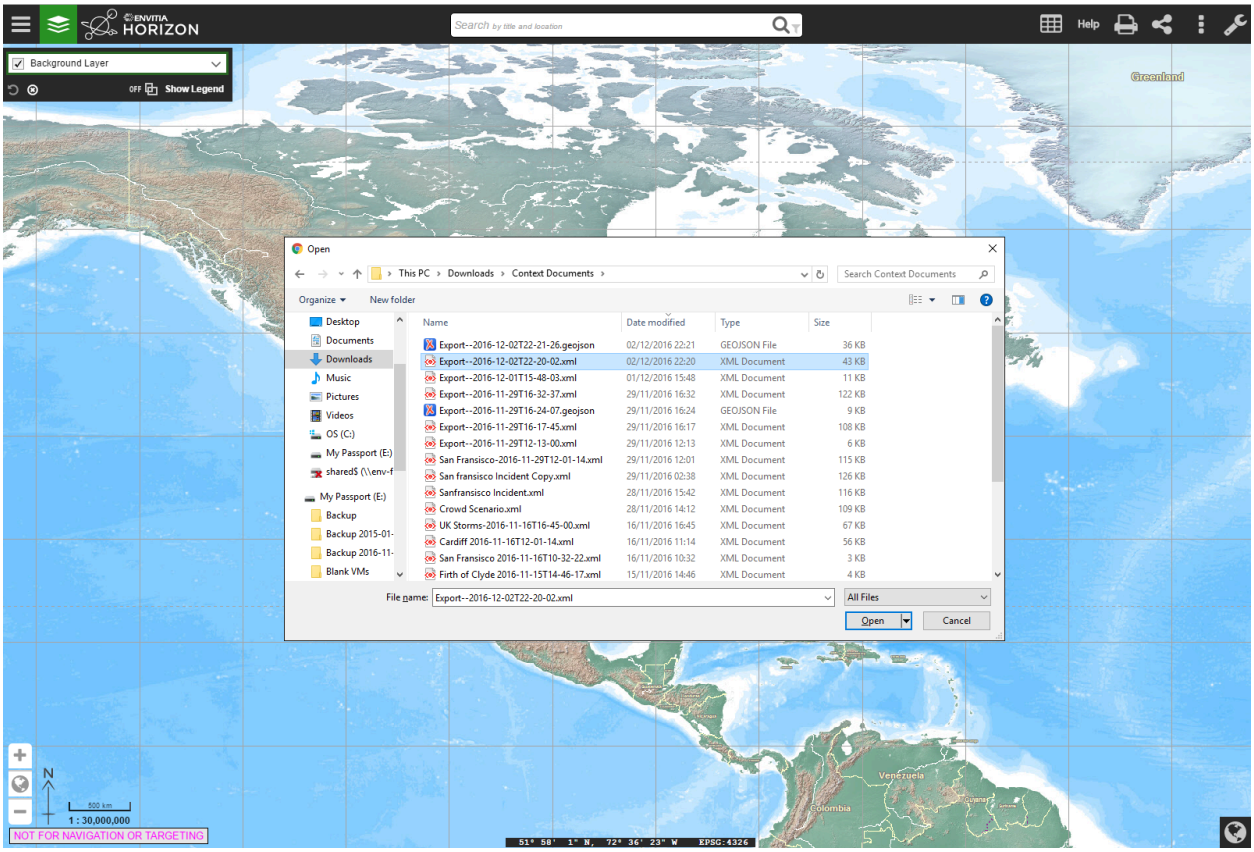

Figure 13 — Open Envitia Horizon

When a file is selected and loaded the geographic display zooms to the relevant area and loads all layers selected as on.

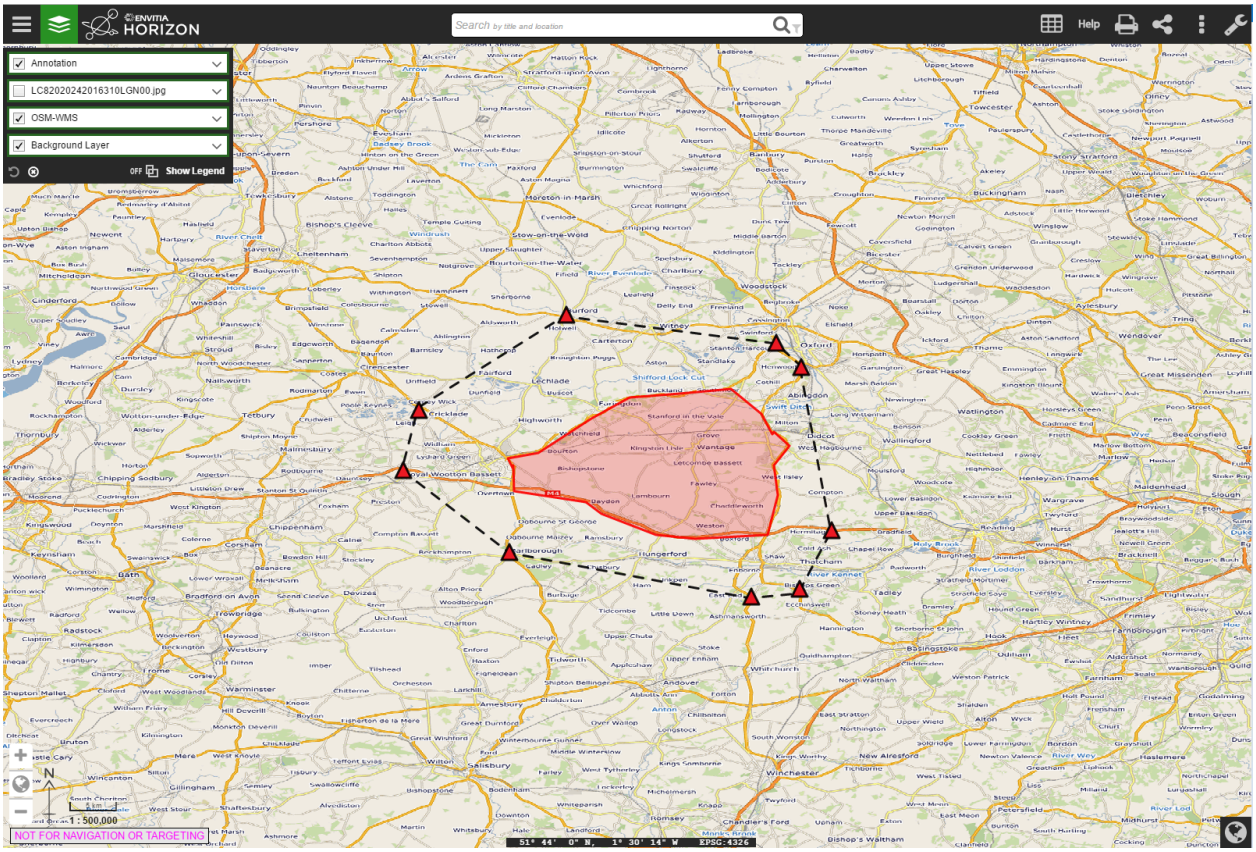

Figure 14 — Context document loaded

4.1.8. Creating a Revised Document

In this example we have been using the same client for both creator and reader. It is therefore possible to then modify the context document by using the catalogue to find and add more data or add additional annotations. A new context document can be created which includes the additional content.

5

OWS CONTEXT FOR DEVELOPERS

5 OWS CONTEXT FOR DEVELOPERS

The goal of this section is to provide guidence to developers on implementing OWS Context in applications. It addresses both reading and writing OWS Context documents. The section outlines a typical workflow for applications showing common operating pictures (i.e. to exchange a view of information between multiple users), but other use cases are considered at the end of this chapter.

5.1. Reading OWS Context Documents

There are two encodings of OWS Context, an ATOM/XML encoding and a GeoJSON encoding (both are approved OGC standards, although the GeoJSON encoding is in the final stages of publication). This section addresses both encodings, highlighting differences where they exist.

5.1.1. General Syntax Rules

Before proceeding too far it is worth noting a couple of things about the Context encodings.

5.1.1.1. ATOM/XML Encoding

- The ATOM/XML encoding can be read by normal XML Parsers such as Xerces.
- The encoding uses ATOM (a dialect of XML). This cannot be validated using an XML schema. The approach used for validation of ATOM is Relax NG. Do not try and perform schema validation. (Note a Relax NG Validator for OWS Context ATOM/XML is available on the OGC Team Engine, see Clause 6 for details)
- The ATOM model allows for foreign content to be present in the document. The implication of this is that a reader should not terminate when an element is encountered that it does not understand or expect. The element should simply be ignored.

5.1.1.2. GeoJSON Encoding

The GeoJSON encoding provides a JSON based alternative which is easily consumed by browser apps using JavaScript. It complies with the GeoJSON encoding model. In some cases minor limitations exist in this encoding and these are described where they occur.

5.1.2. Telling if an XML Document is a Context Document

OWS Context documents use the standard extensions of .xml (for an ATOM/XML encoding) and .json or .geojson (for a GeoJSON Encoding). The method of determining if the document is an OWS Context document is for an application to query the following elements inside the file.

5.1.2.1. ATOM/XML Encoding

For an ATOM/XML document XML tag at the top level (i.e directly under the root element which will be 'feed') contains an atom link property:

```
<atom:link
  rel="profile"
  href="http://www.opengis.net/spec/owc-atom/1.0/req/core"
  title="This file is compliant with
  version 1.0 of OWS Context"/>
```

In particular the validation should verify the existence of the link tag, and its attributes rel and href and their values. The title is for information only.

If the document is in a catalogue, it is also likely that the application can tell if it's a context document from its mime type which is:

5.1.2.2. GeoJSON Encoding

As with ATOM/XML, the document needs to be parsed and the element identifying a GeoJSON document as an OWS Context document is the 'profile' property on the overall FeatureCollection within the document, as shown below.

```
{
  "type": "FeatureCollection",
  "id": "http://www.opengis.net/owc/1.0/examples/geojson/1",
  "properties" : {
    "links" : {
    "profiles" : ["http://www.opengis.net/spec/owc-geojson/1.0/req/core"],
    ...
  },
  "features": [
    ...
    }]
  }
```

5.1.3. Mandatory Metadata

Aside from the above, the only other mandatory items in a context document are:

- <atom:language>
- <atom:id>

- <atom:title>
- <atom:updated>

So an application can depend on these tags being present when scanning context documents and this is the minimum information that could be catalogued, listed or presented to the user.

5.1.3.1. ATOM/XML Encoding

Wherever possible in the ATOM/XML encoding of OWS Context the atom element is used so that the document could be interpreted by an atom reader for wider interoperability.

```
<feed xmlns="http://www.w3.org/2005/Atom" xml:lang="en" >
<link 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"/>
  <id>https://portal.opengeospatial.org/twiki/bin/view/OWSContextswg/
SpecAtomEncoding#1</id>
  <title>Context Example :: Algal Pigment</title>
  <updated>2012-02-21T11:58:23Z</updated>
<feed>
```

This is the minimum needed in an OWS Context document. It has no area of interest and no time range and it loads nothing, thus it is of little practical use but it is nevertheless a valid (and harmless) context document. There are valid reasons for such a 'null' document which relate to security and access control, thus it is allowed.

5.1.3.2. GeoJSON Encoding

The equivilent GeoJSON encoding of this content is as follows:

```
"properties" : {
    "lang"    : "en",
    "title"   : "OWS Context GeoJSON Example",
    "updated" : "2012-11-04T17:26:23Z",
    ....
},
```

Note in the Atom encoding the language is captured in the 'feed' tag whereas in the GeoJSON Encoding it is within the 'properties' element.

5.1.4. Optional Metadata

The document has a number of other 'metadata' tags at the top level (abstract, author etc), but in particular there is a reference to ISO compliant metadata which is the most extensive information source.

5.1.4.1. ATOM/XML Encoding

```
<link rel="via" type="application/xml"
```

```
href="http://www.acme.com/collections/algal.xml" title="Algal XML metadata"/>
```

As well as this optional metadata, OWC provides a series of top level elements which are key metadata. As a quick note to writers of OWS Context, the information in these elements and in any referenced ISO metadata, if present should be consistent. The user of the context document should be able to depend on this and use either. The top level elements include:

- atom:subtitle (an abstract for the document)
- atom:author/atom:name (author of the document)
- dc:publisher (publisher, Dublin core Extension)
- atom:generator (application used to create the context)
- atom:rights (access rights. This field is not well defined).
- atom:category/@term (Keywords, multiple)

Each of these has a direct GeoJSON equivilent. See the specification for details.

5.1.4.2. GeoJSON Encoding

The GeoJSON Encoding of these elements is as properties under the 'FeatureCollection' element in GeoJSON.

```
    "subtitle"         (an abstract for the document)
    "authors"          (author(s) of the document)
    "publisher"        (publisher of the document)
    "creator"          (application used to create the context)
    "rights"           (access rights. This field is not well defined).
    "categories.term"  (Keywords, multiple)
```

As well as these attributes though there are two critical optional attributes which an application should interpret, these are area of interest and time interval. These are described below.

5.1.5. Area of Interest Metadata

A context document optionally contains an area of interest metadata item which is defined by an envelope encoded in georss form.

5.1.5.1. ATOM/XML Encoding

In ATOM/XML this typically looks as follows:

```
<georss:where>
    <gml:Polygon>
<gml:exterior>
      <gml:LinearRing>
<gml:posList srsDimension="2">-90 -180 90 -180 90 180 -90 180 -90 -180</gml:
posList>
    </gml:LinearRing>
```

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

Firstly, as this is georss [REF 5], this is specified in WGS-84 (implicit). Note that although the above draws a box the envelope can be any shape of non-intersecting polygon.

5.1.5.2. GeoJSON Encoding

> ***IMPORTANT***
>
> *In GeoJSON there is an issue, as a geometric primitive cannot be a FeatureCollection, it needs to be on a feature. This means a GeoJSON encodiing can only capture a bounding box AOI, not a polygonal one.*

An example of the equivilent GeoJSON Encoding for the above is shown below.

```
{
"type": "FeatureCollection",
  "id": "http://www.opengis.net/owc/1.0/examples/geojson/1",
  "properties" : {
    ....
    },
  "bbox": [-180,-90,180,90],
  "features": [{
    ...
```

In general converting an OWS Context document from ATOM/XML to GeoJSON or from GeoJSON to ATOM/XML is lossless, but in this one respect, complex bounding envelopes will be converted in a conversion from ATOM/XML to GeoJSON to rectangular bounding envelopes.

5.1.5.3. Display Client Use the AoI of an OWS Context Document

The OWS Context document standard does not recommend any specific client action in relation to properties such as the Area of interest, but it is desirable that all clients represent the contents in a similar way as possible. The following is suggested behaviour which maximises the chances of commonality in interpretation of the context documents. Ultimately it may form best practice but should be considered at this stage simply suggested behaviour.

Suggestion 1: If no bounding box is specified, do not change the current view when the context document is loaded. The merit of this is that a user can create a 'master' Context Document with the AOI and then a series of other themed OWS Context Documents which when loaded will supplement the content but wont change the AOI.

Suggestion 2: If a bounding box is specified, but it is a different shape to the screen area of the client in which the OWC is to be displayed, perform a best fit, which ensures that the entire area of the AOI envelope is displayed.

Suggestion 3: An alternative, if possible, would be to force the application's geo-view to fit the OWS Context document AOI Aspect ratio but this is often problematic. Because it is unlikely that the client window AOI is a different shape from the AOI in the context document.

Suggestion 4: Some clients (particularly web clients) operate on fixed zoom values and so may not be able to exactly match the zoom level of an OWS Context document's AOI. The recommendation is, if this is the case, adopt the lower zoom (smaller scale) so that the entire AOI is displayed.

Suggestion 5: The client could display the AOI in some way (say as a dotted box which can be turned on or off). This means the user can see what the intended AOI of the context document was.

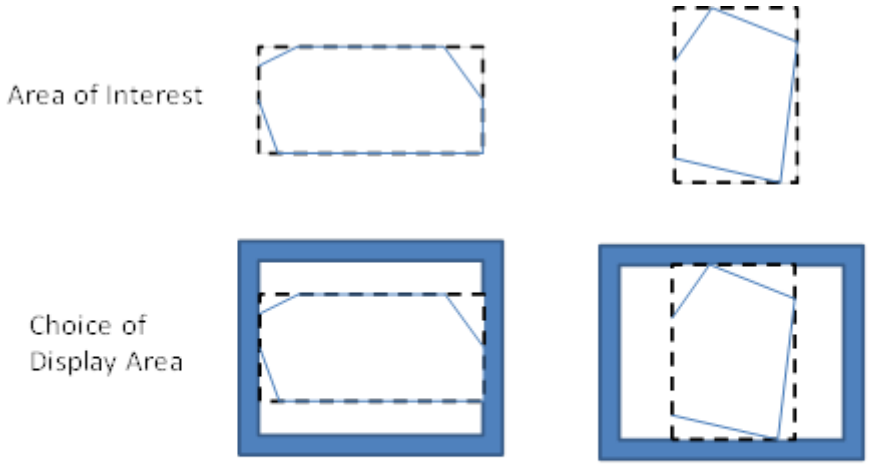

Figure 15 — Matching AOI to Display.

There are of course options in terms of the behaviour. The application could ask the user if they want to zoom to the extent of the context document. Clients could also optionally display the extent of the context document as well as zooming to it. But the overall behavior should be that if the context document has an AOI the user should be able to centre on and zoom to it easily.

There is also a potential issue with some browser based clients as these often have fixed zoom levels. So matching the zoom level in an OWS Context document to the zoom level options available in a browser client is always a little difficult.

IMPORTANT

So, from the above, it should be noted that it is not possible to exactly re-create the view in terms of AOI present on the screen that created an OWS Context document. A close approximation is possible but re-creating the exact area is challenging on hetrogeneous devices.

The AOI is also not intended to provide metadata on the extent of the data present in the context document which may well extend outside of the AOI. It is providing a clear indication of what geographic area of interest is. The AOI is of course also valuable when searching for

Context documents for a given operation or mission. As a result writers of context documents are encouraged to populate it.

5.1.6. Time Interval of Interest Metadata

This element also has an important role in a context document for some applications. Some of the data included in a context document may have a time range. This might include weather data or event based recording data. The Time Interval of Interest, like the area of interest, is not the time extent of the data, it is the time range which is expected to be of interest to the user. So if a particular event (e.g. movement of a hurricane) has a range of times where something critical happens, the time interval would be set to that range. The interval specified should be compliant with an ISO DateTime interval.

5.1.6.1. Display Client use of Times and Time Intervals

Again, not part of the OWS Context standard but the suggested behavior of the client for time interval is to set the end stops of a time slider to the time range of interest. This is though quite advanced behavior and many clients will not support time range. This is only suggested behavior and the time range may be ignored.

5.1.7. Resources

Resources are the key element of an OWC document. They each reference a set of geospatial information to be treated as a logical element. In display clients resources will be realized as 'layers' in typical display clients. The resources are ordered such that the first item in the document is to be displayed at the front. Resources contain a large number of elements, these fall into several categories.

5.1.7.1. ATOM/XML Encoding

Within the ATOM/XML encoding resources are mapped to ATOM 'Entry' elements:

```
<feed xmlns="http://www.w3.org/2005/Atom" xml:lang="en">
...
<entry>
  <id>http://www.acme.eu/geoserver/wms/#world_countries</id>
...
</entry>
...
</feed>
GeoJSON encoding
Within the GeoJSON Encoding resources are mapped to GeoJSON Feature Elements.

{
  "type": "FeatureCollection",
  "id": "http://www.opengis.net/owc/1.0/examples/geojson/1",
  "properties": {
    ...
  },
```

```
  "features": [{
    "id" : "http://www.acme.eu/geoserver/wms/#world_countries”,
      ...
  }]
}
```

5.1.7.2. Ordering of Resources

Resources in an OWS Context document are ordered. Particularly for visualisation this defines the order in which layers are drawn. The reason for this is that it is often important for visualisation. If an OWS Context document references many layers, background map route data, a series of critical overlays etc, it is important that these are drawn in the right order. For both ATOM/XML and GeoJSON the draw order is the back to the front.

> ***IMPORTANT***
>
> *A potential confusion in the standard is the draw order. It is easy to assume resources are drawn in the order they are read, but this is not true. The draw order for OWS Context Document Resources is from last to first. This is true for both ATOM/XML and GeoJSON.*

The statement about order is present in a footnote at the bottom of the OWS Context Class table in each encoding document.

5.1.7.3. ATOM XML Encoding

The specific statement in the standard, is:

“This specification assigns no significance to the order of appearance of the child elements of atom:feed with the exception of atom:entry The order of atom:entry elements on the atom:feed MAY be used to identifiy the drawing order of the entries. In that case, the first atom:entry represents the top most layer”.

5.1.8. GeoJSON Encoding

The specific statement in the standard is:

“with the exception of member of the features array (the actual Resources). The order of the member of the features MAY be used to identify the drawing order of the resources. In that case, the first item of the array represents the top most layer.

So both documents draw from last resource to first resource with the first resource in the file being drawn last.

5.1.9. Mandatory Elements which related to all content.

These elements are:

- id id of the entry. Unique within the document.
- title title of the entry (used for display of the layer in lists)
- content abstract of the entry.
- updated update date of the layer (as opposed to the whole document).

5.1.9.1. ATOM/XML Encoding

The ATOM/XML items which map to the above are:

- <atom:id> id of the entry. Unique within the document.
- <atom:title> title of the entry (used for display of the layer in lists)
- <atom:content> abstract of the entry.
- <atom:updated> update date of the layer (as opposed to the whole document).

The content element is optional within context but is mandatory in ATOM. It is meant to be populated with displayable content encoded in HTML.

```
<html>
  <body>
  This is an abstract for an OWS Context Document
  </body>
</html>
```

5.1.9.2. GeoJSON encoding

And the equivilent GeoJSON elements are:

- “id”
- “title”
- “abstract”
- “updated”

5.1.10. Optional Metadata Elements

There are a number of elements that fall into this category:

- atom:author (author of the resource)
- dc:publisher (publisher of the resource)
- atom:rights (access rights information for the resource),
- <atom:link@rel=alternate..> Content Description reference (hyperlink)
- <atom:category/@term..> One or more keywords for the layer
- Geospatial extent
- Temporal extent.

GeoJSON equivilents are documented in the specification.

5.1.11. Visibility Attribute

This attribute is an option property on a resource. By default a resource in an OWS Context document is 'Active'. For a client visualising an OWS Context document and treating a resource as a layer for display (the most common usage) active is typically interpreted as visible on load.

This property is intended to allow resources to be provided in a context document, not displayed initially but available to users if they want to use the data. In reality the attribute only needs to be specified where the visibility of the layer is to be off (false) as the resource is by default 'on'.

In the ATOM/XML Encoding this property is encoded as follows:

```
<category scheme="http://www.opengis.net/spec/owc/active"term="false"/>
```

In GeoJSON it is encoded as follows:

```
"properties" : { "active": false }
```

For each of the properties in an OWS Context document you can find a mapping from the conceptual element (in the OWS Context Document Conceptual Model) to the specific encoding.

5.1.12. Display Scale Attributes

A resource has two attributes which, if present are intended to set the display scale range of the resource. These are:

- minScaleDemominator

- maxScaleDenominator

See the specification to determine how to set these, but they related to the scale value that a layer first appears as the user zooms in (maxScaleDenominator) and the scale that it disappears again.

5.1.13. Folder Attribute

The folder attribute is intended to support the concept present in many clients or organising layers into folders. If resources 'Roads' and Crossings are present in a context document and the folder attribute for each is set to 'Transportation/Land' the Roads and Crossings resources would be placed within a tree under Transportation/Land: E.g.

For clients which don't support such a hierarchy the folder attribute can be ignored. Also note, it is up to the writer of the context document to ensure that the folder organisation is consistent with the order of the layers. If it isn't the context document is invalid and the result is undefined.

5.1.14. Offering

As discussed in section 2 an OWC document is trying to address two sorts of client. It is trying to satisfy the need of simple clients to be able to visualise the contents but also to provide enough information for more advanced clients to use the contents as well as visualise the initial view. This means they will for example allow the user to zoom in or out on the initial view represented by the context document.

The primary element which supports the more advanced client is the 'Offering'. A resource (which in GIS terms is a layer as described above) can have a number of offerings, and each offering is focussed on a particular representation of information. These can be one of a number of OGC Web Services, specifically WMS, WMTS, WFS, WCS, WPS and CSW, one of a number of inline or referenced formats, specifically GML, KML, GeoTIFF, GMLJP2, GMLCOV, or a custom offering type defined in a profile or by an organisation.

5.1.15. Multiple Offerings and Priority

Firstly a resource can have multiple offerings, and the goal is for them to be 'more or less semantically equivalent'. In theory a client should be able to choose to read any of the offerings and get the same result.

So for example a OWS Context document has a resource represented by four offerings, a WMS, a WFS with portrayal as SLD, and an inline GML Offering again with portrayal as SLD. Different clients could use these offerings as appropriate:

- a simple browser based client could use the WMS offering provided, using the standard portrayal

- a more sophisticated client, currently on-line is able to apply SLD base portrayal and intends to use the underlying geometry could use the WFS offering and the associated SLD Document.

  Example: A second example is where an OWS Context document specifies both a WFS offering and an in-line GML Offering:

- if the client is operating on-line (and can access the WMS service specified) then it could use the WMS offering.

- if the client is operating off-line then it could default to using in-line offerings (in this case the GML offering) and is thus able to display data.

No priority is given to offerings within a Resource. But it is quite valid for a profile of an OWS Context to force a priority order on the offerings.

Similarly for services such as WCS, a typical client behaviour might be to display this when on-line but to use an alternative offering eg. the GMLJP2 image while off-line.

5.1.16. Offering Identitifcation and Compliance

An offering begins with a simple tag, and then the URI of the offering type, see below:

```
<owc:offering code="http://www.opengis.net/spec/owc-atom/1.0/req/wms">
…
</owc:offering>
```

The equivilent GeoJSON offieting code would be:

```
"properties" : {
  "offerings" : [{
  "code" : "http://www.opengis.net/spec/owc-geojson/1.0/req/wms"
  }]
}
```

The offering type for those types supported in the standard begin with:

`http://www.opengis.net/spec/owc-atom/1.0/req/`

Of for GeoJSON

`http://www.opengis.net/spec/owc-geojson/1.0/req/`

The offering type is actually a direct reference to the OWC Standard Requirement Class. This allows the offering to be easily reviewed as the specification section relating to it is easily identifiable.

It is also important to note that a server or client can implement as many or as few offering types as it wishes. In fact no offering type is mandatory. Thus in declaring compliance it is necessary to declare which conformance classes over and above the core are supported. The reason for not mandating any specific offerings (for example WMS) is that the current approach allows users to simply implement the conformance classes and offerings they need. So for example if you wish to use an OWC document to pass around Web Processing Service requests, you only need to support the Core and the WPS Offering Conformance class.

5.1.17. Conforming to a Specific Offering

IN the OWS Context 1.0 speification, each offering is defined in a requirement class. These classes are specified in the OWS Context Conceptual Model specification (appendix A). The specific operations, content elements and style elements allowed in each specific offering type are specified in this appendix.

5.2. Extension Offerings

The model used for specifying the offering also lends itself to clear delineation of the source of offerings where OWC is extended, for example an offering defined for the ACME company (remember road runner!) would be:

`http://www.acme.com/spec/owc-atom/spec/owc-atom/1.0/req/xxx`

`http://www.acme.com/spec/owc-atom/spec/owc-geojson/1.0/req/xxx`

where xxx was the offering type that the company wished to define. In fact the form of the URI, after the company specification prefix (http://www.acme.com) is really a matter for the profile or offering specification author. OWC Standard Working Group recommends though that the URL be a resolvable URL to allow others to identify how to support the extension offering if they wish.

The OWC Specification uses a relatively simple format to describe extensions. These can be found in the OWS Context Document Conceptual Model (REF???). All of the extensions are described in Appendix A so this provides examples of each type of offering element. Define the requirement class, typically with a minimum of one requirement which is a table summarising:

- Any operations present and their multiplicity
- Any content elements and their multiplicity
- Any style elements and their multiplicity.

Lastly it is recommended that the relevant standards are referenced too.

5.2.1. Structure of an Offering

An offering is designed to allow specialist clients (either OGC or others reading extensions) to be able to exploit the offerings easily. Firstly there are really two main types of offering, the web service offering (referencing data via a web service end point) and the content offering (either referencing information in-line or via a file link).

5.2.2. Web Service Offerings

For a web service offering the offering includes two web service URIs (defined in 'Operation' tags. The first is an OGC Get Capabilities URI and the second is a GetData URI. Here's an example of the Get Capabilities operation:

```
<owc:offering code="http://www.opengis.net/spec/owc-atom/1.0/req/wms">
  <owc:operation method="GET" code="GetCapabilities"
    href="http://www.someserver.com/wrs.cgi?REQUEST=GetCapabilities&#x26;
SERVICE=WMS
    &#x26;VERSION=1.1.1"/>
```

Note this is a fully expanded URI rather than a set of parameters. This is in some senses more complex for clients, but it was agreed, after much discussion, to be the most general approach. Anything can be encoded that can be put in a URI, clients can easily generate context documents without having to break this up, and reading clients at least have a good exemplar of a request that should work (and can test it) before breaking it up.

> ***IMPORTANT***
>
> *One specific note to client implementors, both encoders and readers, the web service calls in an OWS Context Document are not directly executable from xml as specific XML encoding rules apply to URIs. Clients creating a context need to convert any special characters to a valid XML encoding (for example & to &) and clients reading the document need to do the reverse in any URLs before executing them via http.*

Most service offerings have two operations, a 'GetCapabilities' operation and a data operation such as 'GetMap' for WMS or a 'GetFeature' for WFS. Typically the GetCapabilities is an http GET operation, whereas the get data may be either a GET or a POST. An operation has several key parameters. They are:

- Code: This specifies the type of operation, for example GetCapabilities.
- Method: This defines the access method, for example GET or POST.
- Href: This is the URI containing the definition of the request.

Here's an example of a complete WMS Offering (http GET is used on both operations):

```
<owc:offering code="http://www.opengis.net/spec/owc-atom/1.0/req/wms">
  <owc:operation code="GetCapabilities" method="GET"
    type="application/xml"
    href="http://www.opengis.uab.cat/cgi-bin/SatCat/MiraMon.cgi?SERVICE=
WMS&#x26;VER
SION=1.1.1&#x26;REQUEST=GetCapabilities"/>
    <owc:operation code="GetMap" method="GET" type="image/jpeg"
      href="http://www.opengis.uab.cat/cgi-bin/SatCat/MiraMon.cgi?SERVICE=
WMS&#x26;VER
SION=1.1.1&#x26;REQUEST=GetMap&#x26;SRS=EPSG:23031&#x26;BBOX=355000,4539000,475
000
```

```
,4619000&#x26;WIDTH=600&#x26;HEIGHT=400&#x26;LAYERS=TotCatalunyaED50&#x26;
FORMAT=im
age/jpeg&#x26;STYLES=opti_fals&#x26;TIME=2011-03"/>
</owc:offering>
```

When a POST method is used the body of the request is delivered in a 'request' tag, which specifies a type, for example 'application/xml'. There is one other key element of an operation, and that is the ability to capture the result of the request in the operation. A typical example is a catalogue request offering, where the result as well as the request can be included. Here is a POST request fragment, with the post body (in the owc:request tag) and the result captured (in the owc:result tag).

```
<owc:operation method="POST" code="GetRecords"
href="http://www.someserver.com/wrs.cgi?">
<owc:request type="application/xml">
<GetRecords
service="CSW"
version="2.0.2"
maxRecords="5"
startPosition="1"
resultType="results"
outputFormat="application/xml"
outputSchema="http://www.opengis.net/cat/csw/2.0.2"
xmlns="http://www.opengis.net/cat/csw/2.0.2"
xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:ows="http://www.opengis.net/ows"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dct="http://purl.org/dc/terms/"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2
http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd">
<Query typeNames="csw:Record">
<ElementSetName typeNames="csw:Record">full</ElementSetName>
<Constraint version="1.1.0">
<ogc:Filter>
<ogc:And> <ogc:PropertyIsLike escapeChar="\" singleChar="?"
wildCard="*">
<ogc:PropertyName>dc:title</ogc:PropertyName>
<ogc:Literal>*Elevation*</ogc:Literal>
</ogc:PropertyIsLike>
<ogc:Intersects>
<ogc:PropertyName>ows:BoundingBox</ogc:PropertyName>
<gml:Envelope>
<gml:lowerCorner>14.05 46.46</gml:lowerCorner>
<gml:upperCorner>17.24 48.42</gml:upperCorner>
</gml:Envelope>
</ogc:Intersects>
</ogc:And>
</ogc:Filter>
</Constraint>
</Query>
</GetRecords>
</owc:request>
<owc:result type="application/xml">
<csw:Record
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:ows="http://www.opengis.net/ows"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2
http://schemas.opengis.net/csw/2.0.2/record.xsd">
<dc:creator>U.S. Geological Survey</dc:creator>
<dc:contributor>State of Texas</dc:contributor>
<dc:publisher>U.S. Geological Survey</dc:publisher>
<dc:subject>Elevation, Hypsography, and Contours</dc:subject>
<dc:subject>elevation</dc:subject>
<dct:abstract>Elevation data collected for the National
Elevation Dataset (NED) ba based on 30m horizontal and 15m vertical
accuracy.</dct:abstract>
<dc:identifier>ac522ef2-89a6-11db-91b1-
7eea55d89593</dc:identifier>
<dc:relation>OfferedBy</dc:relation>
<dc:source>dd1b2ce7-0722-4642-8cd4-6f885f132777</dc:source>
<dc:rights>Copyright © 2004, State of Texas</dc:rights>
<dc:type>Service</dc:type>
<dc:title>National Elevation Mapping Service for
Texas</dc:title>
<dct:modified>2004-03-01</dct:modified>
<dc:language>en</dc:language>
<ows:BoundingBox>
<ows:LowerCorner>-108.44 28.229</ows:LowerCorner>
<ows:UpperCorner>-96.223 34.353</ows:UpperCorner>
</ows:BoundingBox>
</csw:Record>
</owc:result>
</owc:operation>
</owc:offering>
</entry>
...
</feed>
```

The use of this is when using an OWS Context document to deliver for example a set of catalogue queries. It may be that a geospatial support person has been asked to find potential datasets, and they have prepared queries to find the relevant elevation datasets. The above structure would deliver the resulting datasets discovered and also provide the recipient with the queries used to obtain them.

5.2.3. Content Offerings

Content Offerings allow content to be embedded in an OWS Context document. These may be for example annotations or other information which is relatively small but is needed to qualify the bulk of the information in the context document. It might for example give an indication of where an oil spill is or where a polygon is needed to show the oil coverage and a point to show where the vessel is resting. An offering with content in it will use the 'owc:content' tag in the offering. In the example below the owc:content element is defining content inline.

```
<owc:offering
code="http://www.opengis.net/spec/owc-atom/1.0/req/gml">
<owc:content type="application/gml+xml">
<gml:FeatureCollection gml:id="1234567890"
xmlns:clk="http://www.envitia.com/clk" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.envitia.com/clk clk.xsd">
<gml:boundedBy>
<gml:Envelope srsName="urn:opengis:def:crs:EPSG::28992">
<gml:lowerCorner>5.000000 -76.318245</gml:lowerCorner>
```

```
<gml:upperCorner>28.485352 -37.000000</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
<gml:featureMembers>
<clk:al212010_position>
<clk:geometry>
<gml:Point srsName="urn:opengis:def:crs:EPSG::4326">
<gml:Pos srsDimension="2">
5.000000 -37.000000
</gml:Pos>
</gml:Point>
</clk:geometry>
</clk:al212010_position>
</gml:featureMembers>
</gml:FeatureCollection>
</owc:content>
</owc:offering>
```

A content element can also reference content via a URL or a relative or absolute file path.

```
<owc:offering
code="http://www.opengis.net/spec/owc-atom/1.0/req/geotiff">
<content type="image/tiff"
href="file:\\home\gdal_eg/cea.tif"/>
</owc:offering>
</entry>
```

5.2.4. Style Elements in Offerings

Offerings can have style documents associated with them. This is possible for both web services and content offerings. In fact an offering can have multiple style documents associated with it. There is a ‘default’ tag which indicates which style should be displayed when the OWC document is loaded. Styles are not necessarily needed for all offerings, for example they may be provided with a WFS or WCS offering to define the appearance for a layer which is to be visualised, but are not essential for WMS. The suggested behaviour for styles is if there is more than one, the user should be able to select the alternative styles.

5.3. Extending The OWS Context Document Standard

The OWS Context Document has a number of methods of extension.

5.3.1. General Extension Mechanisms

Firstly, the OWS Context conceptual model contains ‘extension’ on almost all complex groups, so anything can be extended. In the ATOM/XML encoding of OWS Context this is implicit at ATOM allows extension at any point. Unlike XSD/XML Schema validation, Relax NG used for ATOM/XML allows extra content at any point, and ATOM parsers are mandated to ignore content they dont understand. GeoJSON also allows implementors to add content. There is

currently no validation mechanism for GeoJSON but any validator would ignore additional unknown content as long as it is syntactically valid GeoJSON.

5.3.2. Identifying if an OWS Context Document has been Extended

Alas at present there is no way, without reading all the content, to identify if an OWS Context document is a standard or extended document, or to characterise the extension.

In Atom it is technically possible to add further 'rel=profile' elements to indicate this but it is not mandated by the standard. For example the following is valid:

```
<atom:link
  rel="profile"
  href="http://www.opengis.net/spec/owc-atom/1.0/req/core"
  title="This file is compliant with
  version 1.0 of OWS Context"/>

<atom:link
  rel="profile"
  href="http://www.MyWebsite.com/spec/owc-textExtensions/1.0/req/csv"
  title="This file is compliant with the MyWebsite Text Extensions to
   OGC Context"/>
```

5.4. Extension Offerings

The most common type of extension to an OWS Context Document is to add offerings. Offerings relate to a specific web service, file or API interface (for example WMS, GeoTIFF or SQL Database). It is common to want to add additional types, either generally or specifically required to support an organisational requirement.

The model used for specifying the offering also lends itself to clear delineation of the source of offerings where OWC is extended, for example an offering defined for the ACME company (remember road runner!) would be:

`http://www.acme.com/spec/owc-atom/1.0/req/xxx`

and in GeoJSON

`http://www.acme.com/spec/owc-geojson/1.0/req/xxx`

where xxx was the offering type that the company wished to define. In fact the form of the URI, after the company specification prefix (http://www.acme.com) is really a matter for the profile or offering specification author.

The SWG suggests the inclusion of the encoding type (e.g owc-geojson) and a version number (which relates to the offering itself) as a way of future-proofing the offering definition.

The OWC SWG recommends though that the offering URL be a resolvable URL to allow others to identify how to support the extension offering if they wish.

The OWC Specification uses a relatively simple format to describe extensions. These can be found in the OWS Context Document Conceptual Model 12-080r2). All of the extensions are described in Appendix A so this provides examples of each type of offering element. Define the requirement class, typically with a minimum of one requirement which is a table summarising:

- Any operations present and their multiplicity
- Any content elements and their multiplicity
- Any style elements and their multiplicity.

The convention within the standard is to use the 'Requirement Class' id to identify the offering. This is convenient as it points to the specification section that relates to the offering.

For an external organisation wishing to create an offering type, we recommend the following:

- Define an offering code which is a valid URI in a namespace owned by the organisation.
- If possible make the offering code a URL and make it resolvable.
- Define the offering purpose and implementation in the HTML or link supplied at the end of the URL.
- Register the Offering and URL with www.owscontext.org

Considering the contents of an offering, within the constraints of the encoding environment (ATOM/XML, GeoJSON etc) an implementor is free to use any syntax they wish. It is helpful though to use the standard patterns within offerings so that implementation issues are eased on clients.

The OWS Context Offerings defined in the standard are characterised by two types, ServiceOfferings (which reference a web service) and ContentOfferings (which reference a file or encode content in-line).

Service offerings typically

5.5. Examples of OWS Context Documents

The following are examples of OWS Context Documents created during Testbed 12. Note they are not validated and so it should not be assumed that the implementation is correct. If in doubt please consult the standards themselves (See [ref 2] and [ref 3]):

http://www.opengeospatial.org/standards/owc

5.5.1. Standard Examples of OWS Context Documents

The following examples are provided alongside the OWS Context Standard in the OGC Standards Repository.

http://schemas.opengis.net/owc/1.0/examples/

5.5.2. Envitia TB12 OWS Context documents in ATOM/XML and GeoJSON

The following documents were generated from the Envitia Horizon Client. They show the following offering types as listed in the table below.

LAYER NAME	OFFERING TYPE	ACCESS METHOD
Intervisibility	WPS	POST
us__countiescountiesType	WFS	GET
us_counties	WMS	GET
BlueMarbleCov	WCS	GET

The Atom/XML example is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xml:lang="en" xmlns="http://www.w3.org/2005/Atom" 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://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"
    xmlns:env="http://www.envitia.com/env">
    <link rel="profile" href="http://www.opengis.net/spec/owc-atom/1.0/req/
core"
        title="This file is compliant with version 1.0 of OWS Context"/>
    <id>1475248700290</id>
    <title>Export--2016-09-30T16:18:20</title>
    <updated>2016-09-30T16:18:20Z</updated>
    <author>
        <name>Envitia</name>
        <email>support@envitia.com</email>
        <uri>http://www.envitia.com</uri>
    </author>
    <georss:where>
        <gml:Polygon>
```

```
            <gml:exterior>
                <gml:LinearRing>
                    <gml:posList srsDimension="2" srsName="EPSG:4326">-
23.293116707657
                        -154.30193347887 94.382556027313 -154.30193347887
94.382556027313
                        57.363088142915 -23.293116707657 57.363088142915 -
23.293116707657
                        -154.30193347887</gml:posList>
                </gml:LinearRing>
            </gml:exterior>
        </gml:Polygon>
    </georss:where>
    <entry>
        <id>OpenLayers_Layer_Image_135270</id>
        <title>Intervisibility</title>
        <updated>2016-09-30T16:18:20Z</updated>
        <georss:where>
            <gml:Polygon>
                <gml:exterior>
                    <gml:LinearRing>
                        <gml:posList srsDimension="2" srsName="EPSG:4326">
51.532798732668745
                            -3.11458080574287 51.644713993337504 -
3.11458080574287
                            51.644713993337504 -2.934917748918966
51.532798732668745
                            -2.934917748918966 51.532798732668745 -
3.11458080574287</gml:posList>
                    </gml:LinearRing>
                </gml:exterior>
            </gml:Polygon>
        </georss:where>
        <content type="html">Intervisibility</content>
        <category term="true" scheme="http://www.opengis.net/owc/active"/>
        <category term="1" scheme="http://www.envitia.com/horizon/layer/
opacity"/>
        <owc:offering code="http://www.opengis.net/spec/owc-atom/1.0/req/wps">
            <owc:operation code="GetCapabilities" method="GET"
                href="http://10.68.2.68:11080/MapLinkOGCServices/OGC?REQUEST=
GetCapabilities&#x26;SERVICE=WPS&#x26;VERSION=1.0.0"/>
            <owc:operation code="Execute" method="POST"
                href="http://10.68.2.68:11080/MapLinkOGCServices/OGC?">
                <owc:request type="text/xml">
                    <wps:Execute service="WPS" version="1.0.0"
                        xmlns:wps="http://www.opengis.net/wps/1.0.0"
                        xmlns:ows="http://www.opengis.net/ows/1.1"
                        xmlns:xlink="http://www.w3.org/1999/xlink"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        xsi:schemaLocation="http://www.opengis.net/wps/1.0.0 ..
\schemas\wps\1.0.0\wps\Execute_request.xsd">
                        <ows:Identifier>MultiViewShed</ows:Identifier>
                        <wps:DataInputs>
                            <wps:Input>
                                <ows:Identifier>source</ows:Identifier>
                                <ows:Title>source</ows:Title>
                                <wps:Data>
                                    <wps:LiteralData>britsouthlatlon</wps:
LiteralData>
                                </wps:Data>
                            </wps:Input>
                            <wps:Input>
                                <ows:Identifier>view_htype</ows:Identifier>
```

```
                                <ows:Title>view_htype</ows:Title>
                                <wps:Data>
                                    <wps:LiteralData>groundHeight</wps:
LiteralData>
                                </wps:Data>
                            </wps:Input>
                            <wps:Input>
                                <ows:Identifier>view_maxRadius</ows:Identifier>
                                <ows:Title>view_maxRadius</ows:Title>
                                <wps:Data>
                                    <wps:LiteralData>10000</wps:LiteralData>
                                </wps:Data>
                            </wps:Input>
                            <wps:Input>
                                <ows:Identifier>target_height</ows:Identifier>
                                <ows:Title>target_height</ows:Title>
                                <wps:Data>
                                    <wps:LiteralData>0</wps:LiteralData>
                                </wps:Data>
                            </wps:Input>
                            <wps:Input>
                                <ows:Identifier>target_htype</ows:Identifier>
                                <ows:Title>target_htype</ows:Title>
                                <wps:Data>
                                    <wps:LiteralData>groundHeight</wps:
LiteralData>
                                </wps:Data>
                            </wps:Input>
                            <wps:Input>
                                <ows:Identifier>requiredDisplayWidth</ows:
Identifier>
                                <ows:Title>requiredDisplayWidth</ows:Title>
                                <wps:Data>
                                    <wps:LiteralData>143</wps:LiteralData>
                                </wps:Data>
                            </wps:Input>
                            <wps:Input>
                                <ows:Identifier>requiredDisplayHeight</ows:
Identifier>
                                <ows:Title>requiredDisplayHeight</ows:Title>
                                <wps:Data>
                                    <wps:LiteralData>89</wps:LiteralData>
                                </wps:Data>
                            </wps:Input>
                            <wps:Input>
                                <ows:Identifier>displayStyle</ows:Identifier>
                                <ows:Title>displayStyle</ows:Title>
                                <wps:Data>
                                    <wps:LiteralData>redGreen</wps:LiteralData>
                                </wps:Data>
                            </wps:Input>
                            <wps:Input>
                                <ows:Identifier>viewPoints</ows:Identifier>
                                <ows:Title>viewPoints</ows:Title>
                                <wps:Data>
                                    <wps:LiteralData>&#x3c;gml:LineString
                                        srsName=&#x22;EPSG:4326&#x22;&#x3e;
&#x3c;gml:posList
                                        srsDimension=&#x22;3&#x22;&#x3e;
51.588791004021
                                        -3.0247492773309 0
                                        &#x3c;/gml:posList&#x3e;&#x3c;/gml:
LineString&#x3e;</wps:LiteralData>
```

```
                                </wps:Data>
                            </wps:Input>
                            <wps:Input>
                                <ows:Identifier>requiredDisplayExtent</ows:
Identifier>
                                <ows:Title>requiredDisplayExtent</ows:Title>
                                <wps:Data>
                                    <wps:BoundingBoxData crs="EPSG:4326">
                                        <ows:LowerCorner>-3.11458080574287
                                            51.532798732668745</ows:
LowerCorner>
                                        <ows:UpperCorner>-2.934917748918966
                                            51.644713993337504</ows:
UpperCorner>
                                    </wps:BoundingBoxData>
                                </wps:Data>
                            </wps:Input>
                        </wps:DataInputs>
                        <wps:ResponseForm>
                            <wps:ResponseDocument>
                                <wps:Output mimeType="image/png" asReference=
"true">
                                    <ows:Identifier>image</ows:Identifier>
                                </wps:Output>
                            </wps:ResponseDocument>
                        </wps:ResponseForm>
                    </wps:Execute>
                </owc:request>
            </owc:operation>
        </owc:offering>
    </entry>
    <entry>
        <id>OpenLayers_Layer_Vector_134151</id>
        <title>us__countiescountiesType</title>
        <updated>2016-09-30T16:18:20Z</updated>
        <georss:where>
            <gml:Polygon>
                <gml:exterior>
                    <gml:LinearRing>
                        <gml:posList srsDimension="2" srsName="EPSG:4326">-90 -
180 90 -180 90 180
                            -90 180 -90 -180</gml:posList>
                    </gml:LinearRing>
                </gml:exterior>
            </gml:Polygon>
        </georss:where>
        <content type="html">us__countiescountiesType</content>
        <category term="true" scheme="http://www.opengis.net/owc/active"/>
        <category term="1" scheme="http://www.envitia.com/horizon/layer/
opacity"/>
        <owc:offering code="http://www.opengis.net/spec/owc-atom/1.0/req/wfs">
            <owc:operation code="GetCapabilities" method="GET"
                href="http://demo.luciad.com:8080/LuciadFusion/wfs?REQUEST=
GetCapabilities&#x26;SERVICE=WFS"/>
            <owc:operation code="GetFeature" method="GET"
                href="http://demo.luciad.com:8080/LuciadFusion/wfs?REQUEST=
GetFeature&#x26;SERVICE=WFS&#x26;VERSION=1.0.0&#x26;BBOX=-154.30193347887,-2
3.293116707657,57.363088142915,94.382556027313&#x26;NAMESPACES=xmlns(feature,
null)&#x26;TYPENAME=feature:us__countiescountiesType"
                />
        </owc:offering>
    </entry>
    <entry>
```

```
        <id>OpenLayers_Layer_WMS_133883</id>
        <title>us_counties</title>
        <updated>2016-09-30T16:18:20Z</updated>
        <georss:where>
            <gml:Polygon>
                <gml:exterior>
                    <gml:LinearRing>
                        <gml:posList srsDimension="2" srsName="EPSG:4326">
18.924781799316
                            -178.21502685547 71.406646728516 -178.21502685547
71.406646728516
                            -66.969848632813 18.924781799316 -66.969848632813
18.924781799316
                            -178.21502685547</gml:posList>
                    </gml:LinearRing>
                </gml:exterior>
            </gml:Polygon>
        </georss:where>
        <content type="html">us_counties</content>
        <category term="true" scheme="http://www.opengis.net/owc/active"/>
        <category term="1" scheme="http://www.envitia.com/horizon/layer/
opacity"/>
        <owc:offering code="http://www.opengis.net/spec/owc-atom/1.0/req/wms">
            <owc:operation code="GetCapabilities" method="GET"
                href="http://demo.luciad.com:8080/LuciadFusion/wms?REQUEST=
GetCapabilities&#x26;SERVICE=WMS"/>
            <owc:operation code="GetMap" method="GET"
                href="http://demo.luciad.com:8080/LuciadFusion/wms?REQUEST=
GetMap&#x26;SERVICE=WMS&#x26;TRANSPARENT=true&#x26;LAYERS=us_counties&#x26;
FORMAT=image/png&#x26;VERSION=1.1.1&#x26;STYLES=&#x26;SRS=EPSG:4326&#x26;WIDTH=
1680&#x26;HEIGHT=934&#x26;BBOX=-154.30193347887,-23.293116707657,57.36308814291
5,94.382556027313"
            />
        </owc:offering>
    </entry>
    <entry>
        <id>OpenLayers_Layer_Image_134365</id>
        <title>BlueMarbleCov</title>
        <updated>2016-09-30T16:18:20Z</updated>
        <georss:where>
            <gml:Polygon>
                <gml:exterior>
                    <gml:LinearRing>
                        <gml:posList srsDimension="2" srsName="EPSG:4326">-90 -
180 90 -180 90 180
                            -90 180 -90 -180</gml:posList>
                    </gml:LinearRing>
                </gml:exterior>
            </gml:Polygon>
        </georss:where>
        <content type="html">BlueMarbleCov</content>
        <category term="true" scheme="http://www.opengis.net/owc/active"/>
        <category term="1" scheme="http://www.envitia.com/horizon/layer/
opacity"/>
        <owc:offering code="http://www.opengis.net/spec/owc-atom/1.0/req/wcs">
            <owc:operation code="GetCapabilities" method="GET"
                href="http://ows.rasdaman.org/rasdaman/ows?REQUEST=
GetCapabilities&#x26;SERVICE=WCS&#x26;VERSION=2.0.0"/>
            <owc:operation code="GetCoverage" method="GET"
                href="http://ows.rasdaman.org/rasdaman/ows?SCALEFACTOR=
10.1742970582237l8&#x26;format=image/png&#x26;CoverageId=BlueMarbleCov&#x26;
request=GetCoverage&#x26;version=2.0.0&#x26;service=WCS"
            />
```

```
        </owc:offering>
    </entry>
</feed>
```

Envitia TB12 Atom/XML Example

And the equivilent GeoJSON example is as follows:

```
{
    "type": "FeatureCollection",
    "properties": {
        "lang": "en",
        "title": "Export--2016-09-30T16:18:30",
        "updated": "2016-09-30T16:18:30Z",
        "links": [{
            "rel": "profile",
            "href": "http://www.opengis.net/spec/owc-geojson/1.0/req/core",
            "title": "This file is compliant with version 1.0 of OWS Context"
        }],
        "authors": [{
            "name": "Envitia",
            "email": "support@envitia.com",
            "uri": "http://www.envitia.com"
        }]
    },
    "id": 1475248710263,
    "bbox": [
        -154.30193347887,
        -20.206335142339,
        57.363088142915,
        91.295774461995
    ],
    "features": [
        {
            "properties": {
                "title": "Intervisibility",
                "updated": "2016-09-30T16:18:30Z",
                "content": "Intervisibility",
                "categories": [
                    {
                        "term": true,
                        "scheme": "http://www.opengis.net/owc/active"
                    },
                    {
                        "term": 1,
                        "scheme": "http://www.envitia.com/horizon/layer/
opacity"
                    }
                ],
                "offerings": [{
                    "code": "http://www.opengis.net/spec/owc-geojson/1.0/req/
wps",
                    "operations": [
                        {
                            "code": "GetCapabilities",
                            "method": "GET",
                            "type": "text/xml",
                            "href": "http://10.68.2.68:11080/
MapLinkOGCServices/OGC?REQUEST=GetCapabilities&SERVICE=WPS&VERSION=1.0.0"
                        },
                        {
                            "code": "Execute",
                            "method": "POST",
```

```
                            "href": "http://10.68.2.68:11080/
MapLinkOGCServices/OGC?",
                            "request": {
                                "type": "text/xml",
                                "request": "<wps:Execute service=\"WPS\"
version=\"1.0.0\"
                                    xmlns:wps=\"http://www.opengis.net/wps/
1.0.0\"
                                    xmlns:ows=\"http://www.opengis.net/ows/
1.1\"
                                    xmlns:xlink=\"http://www.w3.org/1999/
xlink\"
                                    xmlns:xsi=\"http://www.w3.org/2001/
XMLSchema-instance\"
                                    xsi:schemaLocation=\"http://www.opengis.
net/wps/1.0.0 ..\\schemas\\wps\\1.0.0\\wps\\Execute_request.xsd\">
                                    <ows:Identifier>MultiViewShed</ows:
Identifier>
                                    <wps:DataInputs><wps:Input>
                                       <ows:Identifier>source</ows:Identifier>
                                       <ows:Title>source</ows:Title>
                                       <wps:Data>
                                          <wps:LiteralData>britsouthlatlon</
wps:LiteralData>
                                       </wps:Data>
                                    </wps:Input>
                                    <wps:Input>
                                       <ows:Identifier>view_htype</ows:
Identifier>
                                       <ows:Title>view_htype</ows:Title>
                                       <wps:Data>
                                          <wps:LiteralData>groundHeight</wps:
LiteralData>
                                       </wps:Data>
                                    </wps:Input>
                                    <wps:Input>
                                       <ows:Identifier>view_maxRadius</ows:
Identifier>
                                       <ows:Title>view_maxRadius</ows:Title>
                                       <wps:Data>
                                          <wps:LiteralData>10000</wps:
LiteralData>
                                       </wps:Data>
                                    </wps:Input>
                                    <wps:Input>
                                       <ows:Identifier>target_height</ows:
Identifier>
                                       <ows:Title>target_height</ows:Title>
                                       <wps:Data>
                                          <wps:LiteralData>0</wps:LiteralData>
                                       </wps:Data>
                                    </wps:Input>
                                    <wps:Input>
                                       <ows:Identifier>target_htype</ows:
Identifier>
                                       <ows:Title>target_htype</ows:Title>
                                       <wps:Data>
                                          <wps:LiteralData>groundHeight</wps:
LiteralData>
                                       </wps:Data>
                                    </wps:Input>
                                    <wps:Input>
```

```
                                    <ows:Identifier>requiredDisplayWidth</
ows:Identifier>
                                    <ows:Title>requiredDisplayWidth</ows:
Title>
                                    <wps:Data>
                                       <wps:LiteralData>143</wps:
LiteralData>
                                    </wps:Data>
                                 </wps:Input>
                                 <wps:Input>
                                    <ows:Identifier>requiredDisplayHeight</
ows:Identifier>
                                    <ows:Title>requiredDisplayHeight</ows:
Title>
                                    <wps:Data>
                                       <wps:LiteralData>89</wps:LiteralData>
                                    </wps:Data>
                                 </wps:Input>
                                 <wps:Input>
                                    <ows:Identifier>displayStyle</ows:
Identifier>
                                    <ows:Title>displayStyle</ows:Title>
                                    <wps:Data>
                                       <wps:LiteralData>redGreen</wps:
LiteralData>
                                    </wps:Data>
                                 </wps:Input>
                                 <wps:Input>
                                    <ows:Identifier>viewPoints</ows:
Identifier>
                                    <ows:Title>viewPoints</ows:Title>
                                    <wps:Data>
                                       <wps:LiteralData>&#x3c;gml:
LineString srsName=&#x22;EPSG:4326&#x22;&#x3e;&#x3c;gml:posList srsDimension=
&#x22;3&#x22;&#x3e;51.588791004021 -3.0247492773309 0   &#x3c;/gml:
posList&#x3e;&#x3c;/gml:LineString&#x3e;</wps:LiteralData>
                                    </wps:Data>
                                 </wps:Input>
                                 <wps:Input>
                                    <ows:Identifier>requiredDisplayExtent</
ows:Identifier>
                                    <ows:Title>requiredDisplayExtent</ows:
Title>
                                    <wps:Data>
                                       <wps:BoundingBoxData crs=
\"EPSG:4326\"><ows:LowerCorner>-3.11458080574287 51.532798732668745</ows:
LowerCorner><ows:UpperCorner>-2.934917748918966 51.644713993337504</ows:
UpperCorner></wps:BoundingBoxData>
                                    </wps:Data>
                                 </wps:Input>
                              </wps:DataInputs>
                              <wps:ResponseForm>
                                 <wps:ResponseDocument>
                                    <wps:Output mimeType=\"image/png\"
asReference=\"true\">
                                       <ows:Identifier>image</ows:
Identifier>
                                    </wps:Output>
                                 </wps:ResponseDocument>
                              </wps:ResponseForm>
                           </wps:Execute>"
                        }
                     }
```

```
                    ]
                }]
            },
            "type": "Feature",
            "id": "OpenLayers_Layer_Image_135270",
            "geometry": {
                "type": "Polygon",
                "coordinates": [[
                    [
                        -3.11458080574287,
                        51.532798732668745
                    ],
                    [
                        -3.11458080574287,
                        51.644713993337504
                    ],
                    [
                        -2.934917748918966,
                        51.644713993337504
                    ],
                    [
                        -2.934917748918966,
                        51.532798732668745
                    ],
                    [
                        -3.11458080574287,
                        51.532798732668745
                    ]
                ]]
            }
        },
        {
            "properties": {
                "title": "us__countiescountiesType",
                "updated": "2016-09-30T16:18:30Z",
                "content": "us__countiescountiesType",
                "categories": [
                    {
                        "term": true,
                        "scheme": "http://www.opengis.net/owc/active"
                    },
                    {
                        "term": 1,
                        "scheme": "http://www.envitia.com/horizon/layer/
opacity"
                    }
                ],
                "offerings": [{
                    "code": "http://www.opengis.net/spec/owc-geojson/1.0/req/
wfs",
                    "operations": [
                        {
                            "code": "GetCapabilities",
                            "method": "GET",
                            "href": "http://demo.luciad.com:8080/LuciadFusion/
wfs?REQUEST=GetCapabilities&SERVICE=WFS"
                        },
                        {
                            "code": "GetFeature",
                            "method": "GET",
                            "href": "http://demo.luciad.com:8080/LuciadFusion/
wfs?REQUEST=GetFeature&SERVICE=WFS&VERSION=1.0.0&BBOX=-154.30193347887,-
```

```
20.206335142339,57.363088142915,91.295774461995&NAMESPACES=xmlns(feature,
null)&TYPENAME=feature:us__countiescountiesType"
                        }
                    ]
                }]
            },
            "type": "Feature",
            "id": "OpenLayers_Layer_Vector_134151",
            "geometry": {
                "type": "Polygon",
                "coordinates": [[
                    [
                        -180,
                        -90
                    ],
                    [
                        -180,
                        90
                    ],
                    [
                        180,
                        90
                    ],
                    [
                        180,
                        -90
                    ],
                    [
                        -180,
                        -90
                    ]
                ]]
            }
        },
        {
            "properties": {
                "title": "us_counties",
                "updated": "2016-09-30T16:18:30Z",
                "content": "us_counties",
                "categories": [
                    {
                        "term": true,
                        "scheme": "http://www.opengis.net/owc/active"
                    },
                    {
                        "term": 1,
                        "scheme": "http://www.envitia.com/horizon/layer/
opacity"
                    }
                ],
                "offerings": [{
                    "code": "http://www.opengis.net/spec/owc-geojson/1.0/req/
wms",
                    "operations": [
                        {
                            "code": "GetCapabilities",
                            "method": "GET",
                            "href": "http://demo.luciad.com:8080/LuciadFusion/
wms?REQUEST=GetCapabilities&SERVICE=WMS"
                        },
                        {
                            "code": "GetMap",
                            "method": "GET",
```

```
                            "href": "http://demo.luciad.com:8080/LuciadFusion/
wms?REQUEST=GetMap&SERVICE=WMS&TRANSPARENT=true&LAYERS=us_counties&FORMAT=
image/png&VERSION=1.1.1&STYLES=&SRS=EPSG:4326&WIDTH=1680&HEIGHT=885&BBOX=-
154.30193347887,-20.206335142339,57.363088142915,91.295774461995"
                        }
                    ]
                }]
            },
            "type": "Feature",
            "id": "OpenLayers_Layer_WMS_133883",
            "geometry": {
                "type": "Polygon",
                "coordinates": [[
                    [
                        -178.21502685547,
                        18.924781799316
                    ],
                    [
                        -178.21502685547,
                        71.406646728516
                    ],
                    [
                        -66.969848632813,
                        71.406646728516
                    ],
                    [
                        -66.969848632813,
                        18.924781799316
                    ],
                    [
                        -178.21502685547,
                        18.924781799316
                    ]
                ]]
            }
        },
        {
            "properties": {
                "title": "BlueMarbleCov",
                "updated": "2016-09-30T16:18:30Z",
                "content": "BlueMarbleCov",
                "categories": [
                    {
                        "term": true,
                        "scheme": "http://www.opengis.net/owc/active"
                    },
                    {
                        "term": 1,
                        "scheme": "http://www.envitia.com/horizon/layer/
opacity"
                    }
                ],
                "offerings": [{
                    "code": "http://www.opengis.net/spec/owc-geojson/1.0/req/
wcs",
                    "operations": [
                        {
                            "code": "GetCapabilities",
                            "method": "GET",
                            "type": "text/xml",
                            "href": "http://ows.rasdaman.org/rasdaman/ows?
REQUEST=GetCapabilities&SERVICE=WCS&VERSION=2.0.0"
                        },
```

```
                        {
                            "code": "GetCoverage",
                            "method": "GET",
                            "href": "http://ows.rasdaman.org/rasdaman/
ows?SCALEFACTOR=10.174297058223718&format=image/png&CoverageId=
BlueMarbleCov&request=GetCoverage&version=2.0.0&service=WCS"
                        }
                    ]
                }]
            },
            "type": "Feature",
            "id": "OpenLayers_Layer_Image_134365",
            "geometry": {
                "type": "Polygon",
                "coordinates": [[
                    [
                        -180,
                        -90
                    ],
                    [
                        -180,
                        90
                    ],
                    [
                        180,
                        90
                    ],
                    [
                        180,
                        -90
                    ],
                    [
                        -180,
                        -90
                    ]
                ]]
            }
        }
    ]
}
```

6

APPLICATIONS AND TOOLS

6 APPLICATIONS AND TOOLS

6.1. Applications

A number of OWS Context Implementations exist. Initial implementors include:

6.1.1. CREAF MiraMon Client

The Centre de Recerca Ecològica i Aplicacions Forestals (CREAF) in Spain has developed a generic client (MiraMon) used for research purposes which is able to read and write OWS Context Documents using the Atom Encoding. This client supports the OWS Context Core and the WMS and WMTS Extensions.

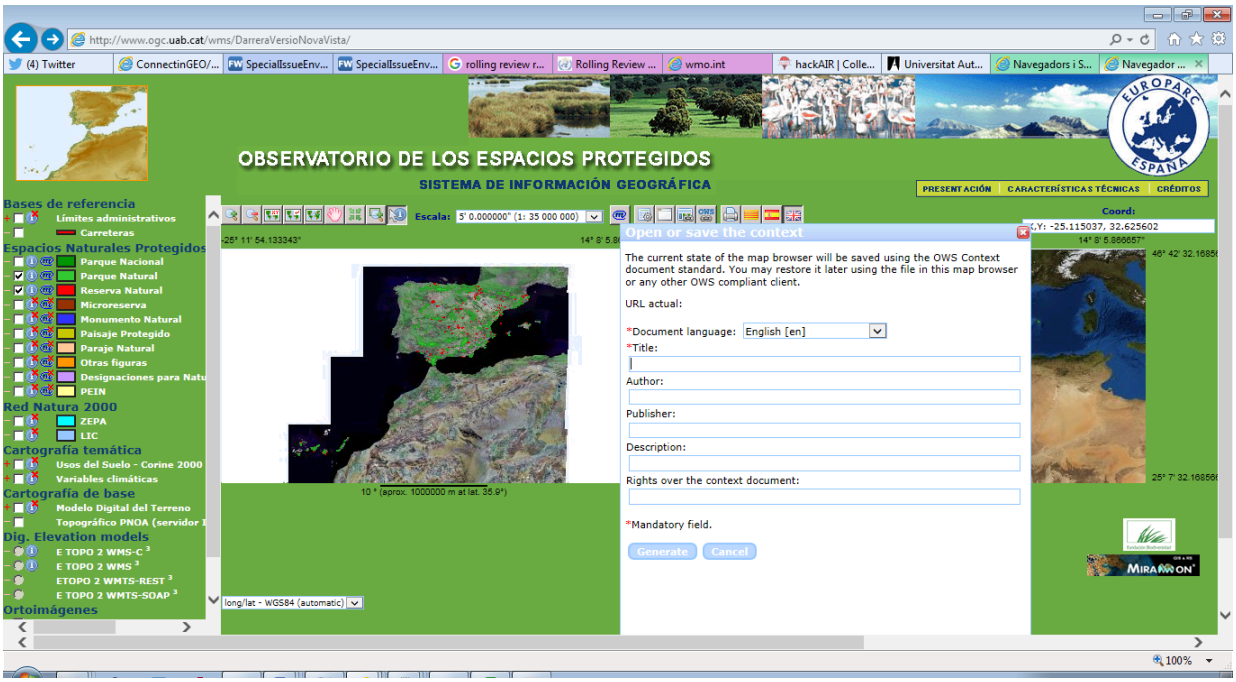

Figure 16 — CREAF MiraMon Client

For more details of the MiraMon client see: http://www.creaf.uab.es/MiraMon/

6.1.2. Compusult

Compusult have added support for OWS Context within their Web Enterprise Suite product line. This includes cataloging, reading and writing OWS Context Documents.

6.1.3. Envitia Ltd/Envitia Inc

Envitia, a UK/US SME operating in the Defence and Intelligence space has implemented support for OWS Context within many of its products.

6.1.3.1. Envitia Horizon GeoPortal

Envitia Horizon supports both the ATOM/XML encoding and Draft GeoJSON Encoding of OWS Context. This is a browser based client) using JavaScript/HTML5 with supporting Java Services. Horizon supports the OWS Context core, the WMS, WMTS, WFS, GML and GeoTIFF extensions. It is also, in prototype form able to support WCS and WPS. All of these extensions are available on both ATOM XML and GeoJSON.

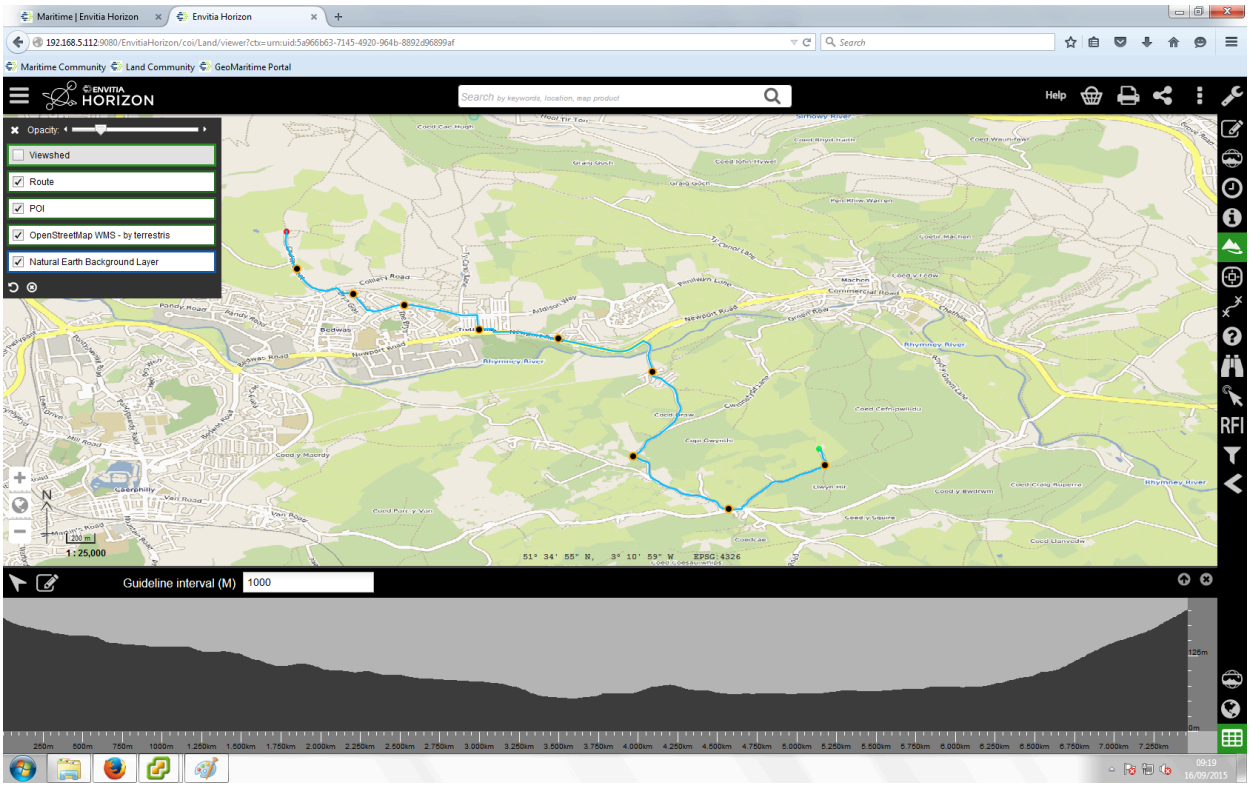

Figure 17 — Envitia Horizon Browser Client

Envitia Horizon is being used, together with other Envitia and Open Source software in the UK Defence Geographic Services system a major UK Defence infrastructure programme being delivered to UK MOD by Hewlett Packard.

6.1.3.2. Envitia MapLink Pro

Envitia MapLink Pro is a C++ based developer toolkit, available on Windows, Linux and Android which includes an OWS Context SDK capable of reading and writing OWS Context Documents.

By default it supports WMS, WMTS and GML but supports the capability to read/write any OWS Context Offering. It can be seen below displaying an OWS Context Document on an Android Tablet.

Figure 18 — Envitia MapLink Pro Android Client

6.1.4. ESRI

ESRI are developing capability based on the GeoJSON encoding of OWS Context, including supporting services such as WMS as well as the ESRI GeoServices REST model.

6.1.5. TerraDue

OWS Context Documents are being used by TerraDue in the platforms being developed for ESA. All the thematic apps are defined using OWS Context documents. See https://hydrology-tep.eo.esa.int/#!thematic] for more information.

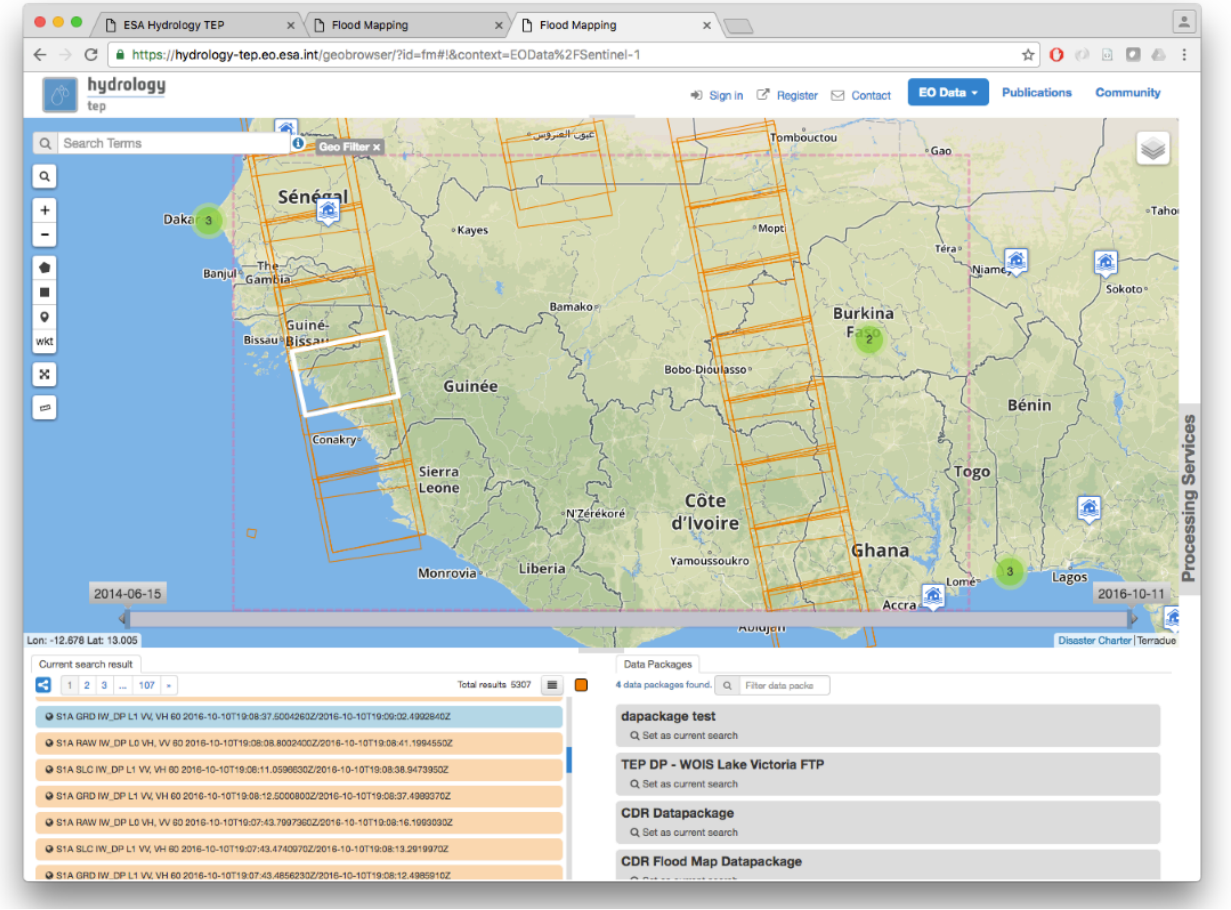

Figure 19 — TerraDue Implementation for ESA

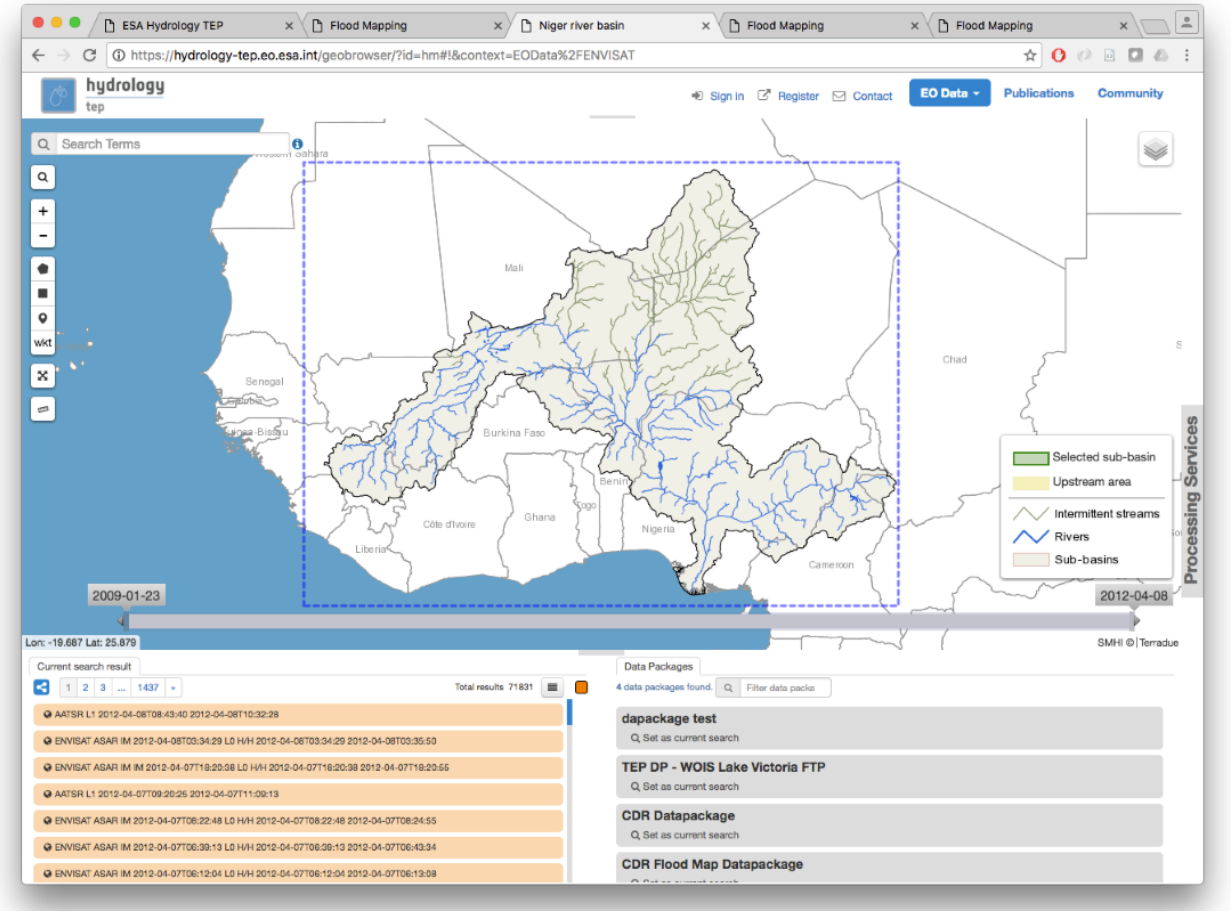

Figure 20 — TerraDue Implementation for ESA

Use of OWS Context is also being extended to the Geo-Hazard TEP area (https://geohazards-tep.eo.esa.int/) and Urban TEP (https://urban-tep.eo.esa.int/)

6.1.6. European Satellite Centre

The European Satellite Centre is using the OWS Context Document to describe collections of imagery relevant to a given scenario or situation. For more information contact the Chief Technology Officer at SATCEN https://www.satcen.europa.eu/

6.2. Tools

6.2.1. OGC CITE Test Validator for OWC Atom/XML Encoding

The OGC CITE test program includes the ability to validate an ATOM/XML OWS Context document.

The tests partially test the following requirements classes defined in the OWS Context Atom Encoding Standard (OGC 12-08r2):

- Requirement: http://www.opengis.net/spec/owc-atom/1.0/req/core
- Requirement: http://www.opengis.net/spec/owc-atom/1.0/req/atomRules
- Requirement: http://www.opengis.net/spec/owc-atom/1.0/req/owcEncoding
- Requirement: http://www.opengis.net/spec/owc-atom/1.0/req/wms/content

At present the validator only tests the OWS Context Core requirement class and the WMS Offering requirement class.

Details of the OWS Context tests can be found at:

http://cite.opengeospatial.org/teamengine/

6.2.2. OWS Context Atom/XML to GeoJSON Converter

Within The OGC Testbed program, a prototype OWS Context to ATOM/XML to GeoJSON converter was developed. These scripts can be executed using any XSLT 2.0 parser and are available at:

http://cite.opengeospatial.org/te2/

The main script takes an ATOM/XML document and generates a valid GeoJSON encoded version.