



Open
Geospatial
Consortium

Training Session on Authoring OGC Standards with Metanorma

Part 2

Getting to know Metanorma for OGC

Meeting sponsor



2022-09-28

Overview

- How to add images
- How to add tables
- How to add code listings
- How to add mathematical formulae
- Specifying ModSpec elements (requirements classes, requirements, conformance classes, and conformance tests)
- Populating the bibliography and references sections

Introduction

- Part 1 of the training session introduced you to *Metanorma for OGC* by:
 - explaining the basics of metanorma
 - providing a hands-on exercise that installs metadata and compiles a document
- Part 2 will provide a hands-on exercise on how to add images, tables, code listings, mathematical formulae, as well as elements of OGC Standards and other specifications

Recap: Difference between asciidoctor and metanorma for OGC

Feature	Asciidoctor	Metanorma for OGC
Document metadata (e.g. docnumber, keywords, etc)	Added as values in table cells	Added as document attributes (More info)
Requirements Classes	Presentation and content specified as tables	Created using a definition list, and then automatically rendered as tables. (More info)
Requirements	Presentation and content specified as tables	Created using a definition list, and then automatically rendered as tables. (More info)
Conformance Classes	Presentation and content specified as tables	Created using a definition list, and then automatically rendered as tables. (More info)
Conformance Tests	Presentation and content specified as tables	Content added as a definition list, and then automatically rendered as tables for Presentation. (More info)

Hands-on Exercise

Recap: Compiling a draft OGC Standard with a docker-containerized Metanorma instance

- To convert the draft standard from asciidoc format to HTML and PDF formats, we use the metanorma software to compile the document.

1. From the folder containing the document .adoc file, run the following command.

```
docker run -v "$(pwd)"/:/metanorma -v  
${HOME}/.fontist/fonts:/config/fonts metanorma/metanorma  
metanorma compile --agree-to-terms -t ogc -x  
xml,html,doc,pdf document.adoc
```

How to add images to an AsciiDoc file

- Preferably use PNG encoded images
- Ideally place the images in the images folder (be consistent)
- Recommended width of 800 pixels or more
- Place an anchor (e.g. `[[my_anchor]]`) above the image so that it can be referenced
- Always add a caption (preceded by dot on the example below)

```
[[img_architecture]]  
.High Level Overview of the Sprint Architecture  
image::images/architecture.png[align="center",width=800]
```


How to add tables

- Place an anchor (e.g. `[[my_anchor]]`) above the table so that it can be referenced
- Configure to number and size of the columns (cols) and other options
- Always add a caption (preceded by dot on the example below)

example.adoc

```
1  [[tbl_capital_cities]]
2  [cols="4,6",options="header"]
3  .Capital cities
4  |===
5  | City Name | Country
6
7  | London | United Kingdom
8  | Madrid | Spain
9  | Ottawa | Canada
10 | Rome | Italy
11 |===
12
```

example.adoc Preview

Table 1. Capital cities

City Name	Country
London	United Kingdom
Madrid	Spain
Ottawa	Canada
Rome	Italy

Advanced tables

Span multiple columns by use of N+|, where N is the number of columns to span

Span multiple rows by use of .n+|, where .n is the number of rows to span

```
example.adoc
1  [[tbl_capital_cities]]
2  [cols="2,4,6",options="header"]
3  .Capital cities
4  |===
5  | Timezone | City Name | Country
6
7  3+| Europe
8  | UTC | London | United Kingdom
9
10 .2+| UTC+1
11 | Madrid | Spain
12 | Rome | Italy
13
14 3+| North America
15
16 | UTC -5 | Ottawa| Canada
17
18 |===
```

example.adoc Preview		
Table 1. Capital cities		
Timezone	City Name	Country
Europe		
UTC	London	United Kingdom
UTC+1	Madrid	Spain
	Rome	Italy
North America		
UTC -5	Ottawa	Canada

How to add code listings

Use the [source,format] directive where 'format' identifies the format of content

```
06-other-clauses.adoc
1 == Other clauses
2
3 This is an example listing in JavaScript Object Notation
4 |
5 [%unnumbered%]
6 [source,json]
7 ----
8 {
9   "type": "Feature",
10  "geometry": {
11    "type": "Point",
12    "coordinates": [-2.682513,63.261372]
13  },
14  "properties": {
15    "title": "SE1_OPER_SEA_GEC_1P_19780927T010430_19780927T010430"
16  }
17 }
18
19 ----
```

6. OTHER CLAUSES

This is an example listing in JavaScript Object Notation (JSON).

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [-2.682513,63.261372]
  },
  "properties": {
    "title": "SE1_OPER_SEA_GEC_1P_19780927T010430_19780927T010430"
  }
}
```

NOTE: The [%unnumbered%] is optional. In this example it is used to prevent metanorma from numbering this particular code listing.

How to add mathematical formulae

- Metanorma AsciiDoc accepts mathematical input in: AsciiMath, LaTeX math, or MathML
- For simplicity, we recommend using AsciiMath

```
4 == Annex title
5
6 This is an example.
7
8 === AsciiMath
9
10 The derivative of a distribution stem:[D] is another
11 • distribution
12 stem:[D'] defined for any function stem:[f](stem:[x])
13 • by
14 stem:[D^( ' ) ( f ) = - D ( d f // d x )].
15
16 [stem]
17 +++++
18 f -= lambda x (a * x + b)
19 +++++
20
21 === LaTeX math
22 .....
23 The only change from the above example would be the
24 .....
25 nondimensionalization of viscosity, which would
26 .....
27 • become,
28 .....
29 latexmath:[\tilde{\tilde{\mu}}] = mu / (rho_infty
30 .....
31 .....
32 .....
33 .....
34 .....
35 .....
36 .....
37 .....
38 .....
39 .....
40 .....
41 .....
42 .....
43 .....
44 .....
45 .....
46 .....
47 .....
48 .....
49 .....
50 .....
51 .....
52 .....
53 .....
54 .....
55 .....
56 .....
57 .....
58 .....
59 .....
60 .....
61 .....
62 .....
63 .....
64 .....
65 .....
66 .....
67 .....
68 .....
69 .....
70 .....
71 .....
72 .....
73 .....
74 .....
75 .....
76 .....
77 .....
78 .....
79 .....
80 .....
81 .....
82 .....
83 .....
84 .....
85 .....
86 .....
87 .....
88 .....
89 .....
90 .....
91 .....
92 .....
93 .....
94 .....
95 .....
96 .....
97 .....
98 .....
99 .....
100 .....
```

PUBLISHED

OGC COMMUNITY PRACTICE

ANNEX A

(NORMATIVE)

ANNEX TITLE

This is an example.

A.1. AsciiMath

The derivative of a distribution D is another distribution D' defined for any function $f(x)$ by $D'(f) = -D(df/dx)$.

$$f \equiv \lambda x(a \cdot x + b) \text{ (A.1)}$$

A.2. LaTeX math

The only change from the above example would be the nondimensionalization of viscosity, which would become, $\tilde{\mu} = \text{mul}(\text{rho_infty}, \text{c_infty}, L)$.

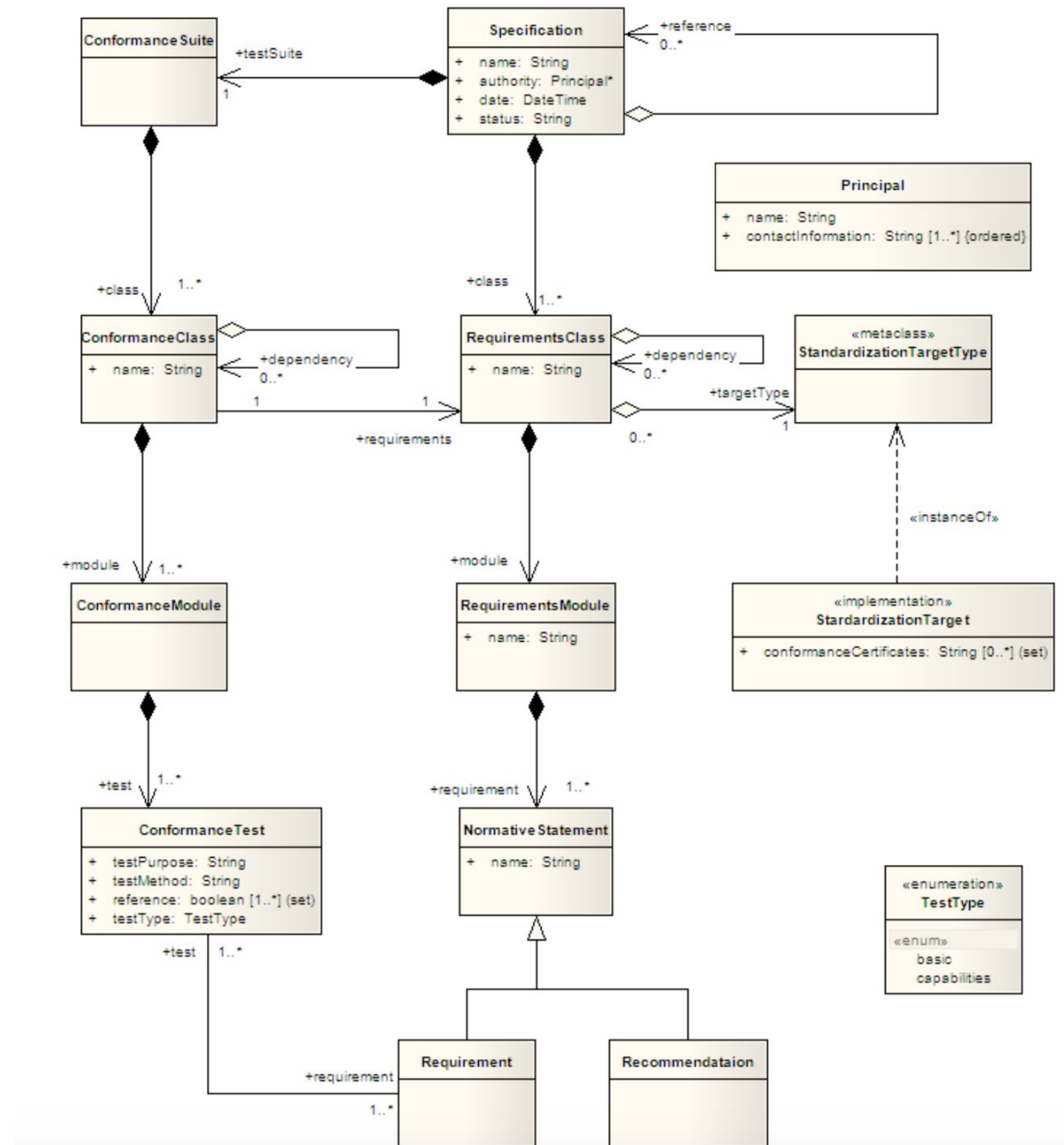
Top

The ModSpec — An OGC Standard for Modular specifications

The ModSpec (OGC 08-131r3) lays requirements against other specifications by using a set-theoretic description of those specifications based on their structure as organized sets of criteria:

- Requirements classes
- Requirements
- Conformance classes
- Conformance tests

<https://www.ogc.org/standards/modularspec>



Documenting requirements classes in metanorma AsciiDoc

An example of a Requirements Class, documented using a definition list, is shown below.

```
[requirements_class]
====
[%metadata]
label:: /req/req-class-building
subject:: Implementation Specification
inherit:: <<rc_core,/req/req-class-core>>
inherit:: <<rc_construction,/req/req-class-construction>>
====
```

Requirements Class rendered

Requirements class 1	
/req/req-class-building	
Target type	Implementation Specification
Dependency	/req/req-class-core
Dependency	/req/req-class-construction

Documenting requirements in metanorma AsciiDoc

An example of a Requirement, documented using a definition list, is shown below.

```
[requirement]
====
[%metadata]
label:: /req/xsd-xml-rules/abc
part:: Data models faithful to the original UML model.
part:: Metadata models faithful to the original UML model.
description:: Logical models encoded as XSDs should be faithful
to the original
UML conceptual models.
====
```

Requirement rendered

Requirement 1	
/req/xsd-xml-rules/abc	
A	Data models faithful to the original UML model.
B	Metadata models faithful to the original UML model.
Logical models encoded as XSDs should be faithful to the original UML conceptual models.	

Notice that the parts are automatically numbered. This is a feature of metanorma.

Documenting conformance classes in metanorma AsciiDoc

An example of a Conformance Class, documented using a definition list, is shown below.

```
[conformance_class]
====
[%metadata]
label:: /conf/crs
subject:: <<rc_crs,Requirements Class 'Coordinate Reference
Systems by Reference'>>
inherit::
http://www.opengis.net/doc/IS/ogcapi-features-1/1.0#ats\_core
classification:: Target Type:Web API
=====
```


Conformance class rendered

Conformance class 1	
/conf/crs	
Requirements class	Requirements Class 'Coordinate Reference Systems by Reference'
Dependency	http://www.opengis.net/doc/IS/ogcapi-features-1/1.0#ats_core
Target type	Web API

Documenting conformance tests in metanorma AsciiDoc

An example of a Conformance Test (Abstract Test), documented using a definition list, is shown below.

```
[abstract_test]
====
[%metadata]
label:: /conf/landing-page/root-op
subject:: <<req_landing-page_root-op,/req/landing-page/root-op>> +
<<req_landing-page_root-success,/req/landing-page/root-success>>
test-purpose:: Validate that a landing page can be retrieved from the expected location.
test-method::
+
--
. Issue an HTTP GET request to the URL {root}/
. Validate that a document was returned with a status code `200`
. Validate the contents of the returned document using test
<<ats_landing-page_root-success,/conf/landing-page/root-success>>.
--
=====
```

Conformance Test rendered

Abstract test 1

/conf/landing-page/root-op

Requirement [/req/landing-page/root-op](#)
 [/req/landing-page/root-success](#)

Test purpose Validate that a landing page can be retrieved from the expected location.

Test method

1. Issue an HTTP GET request to the URL {root}/
2. Validate that a document was returned with a status code 200
3. Validate the contents of the returned document using test [/conf/landing-page/root-success](#).

Populating the References section

- Normative References pulled from a central register, using the relaton toolkit
- Saves editing time, and reduces the possibility of human error

The image shows a development environment with two windows. The left window is a code editor showing a file named `03-references.adoc` with the following content:

```
1 [bibliography]
2 == References
3
4 * [[[openapi,openapi]]] Open API Initiative: OpenAPI Specification
  • 3.0.3, https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md
  • https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md
5 * [[[rfc3896,rfc3896]]] Berners-Lee, T., Fielding, R., Masinter, L:
  • IETF RFC 3896, Uniform Resource Identifier (URI): Generic Syntax,
  • https://tools.ietf.org/rfc/rfc3896.txt
6 * [[[iso19115-1,ISO 19115-1]]]
7 * [[[iso19115-2,ISO 19115-2]]]
8 * [[[iso19115-3,ISO 19115-3]]]
9
```

The right window is a web browser displaying a draft report titled "Joint OGC and ISO Code Sprint 2022 Summary Engineering Report". The report has a table of contents and a section for normative references.

CONTENTS

- I. EXECUTIVE SUMMARY
- II. KEYWORDS
- III. SECURITY CONSIDERATIONS
- IV. SUBMITTERS
- V. ABSTRACT
- 1. SCOPE
- 2. NORMATIVE REFERENCES**
- 3. TERMS, DEFINITIONS AND ABBREVIATED TERMS
 - 3.6. Abbreviated terms
- 4. HIGH-LEVEL ARCHITECTURE
 - 4.1. Approved and Draft Standards
 - 4.2. Open Source Software Projects
 - 4.3. Proprietary products
 - 4.4. Other solutions
- 5. RESULTS
 - 5.1. Leaflet
 - 5.2. MariaDB CubeWex CubeSERV
 - 5.3. Idproxy

2. NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Open API Initiative: OpenAPI Specification 3.0.3, <https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md>

Berners-Lee, T., Fielding, R., Masinter, L: IETF RFC 3896, Uniform Resource Identifier (URI): Generic Syntax, <https://tools.ietf.org/rfc/rfc3896.txt>

ISO: ISO 19115-1, *Geographic information – Metadata – Part 1: Fundamentals*. International Organization for Standardization, Geneva <https://www.iso.org/standard/53798.html>.

ISO: ISO 19115-2, *Geographic information – Metadata – Part 2: Extensions for acquisition and processing*. International Organization for Standardization, Geneva <https://www.iso.org/standard/67039.html>.

ISO: ISO/TS 19115-3, *Geographic information – Metadata – Part 3: XML schema implementation for fundamental concepts*. International Organization for Standardization, Geneva <https://www.iso.org/standard/32579.html>.

Populating the Bibliography section

- Use the LNCS Style as per OGC Editorial Guidance
- Number the anchors of the bibliographic items using [[[anchor_id,serial_number]]]

az-bibliography.adoc

1

2

3

4

5

6

7

8

9

10

11

12

13

[appendix,obligation=informative]

[[annex-bibliography]]

[bibliography]

== Bibliography

* [[[ogc20-004,1]]], Vretanos, P.A, Kralidis, T., Heazel, C.: OGC 20-004: Draft OGC API – Records – Part 1: Core, <http://docs.ogc.org/DRAFTS/20-004.html>

* [[[ogc21-045,2]]], Portele, C., Vretanos, P.A: OGC 21-045: OGC Features and Geometries JSON – Part 1: Core, <https://docs.ogc.org/DRAFTS/21-045.html>

* [[[stacref,3]]], STAC Community: SpatioTemporal Asset Catalog, <https://stacspect.org/en>

* [[[Holmes2021,4]]], Holmes, C.: Static SpatioTemporal Asset Catalogs in Depth, Available at <https://medium.com/radiant-earth-insights/static-spatiotemporal-asset-catalogs-in-depth-710530934a84>

OGC EN

5.8. pycsw

5.9. OWSLib

5.10. Geopython stack

5.11. ISO 19115 activity by OpenWork

5.12. 3DGI CityJSON and JSON-FG Viewer

5.13. Secure and Asynchronous Catalogue

5.14. University of Manchester Natural Language Processing for Spatial Analysis

6. DISCUSSION

6.1. Harmonization between STAC and OGC API Records

6.2. Harvesting

6.3. ISO 19115 metadata and OGC API Records

6.4. JSON-FG

7. CONCLUSIONS

7.1. Future Work

ANNEX A (INFORMATIVE) REVISION HISTORY

BIBLIOGRAPHY

2022-09-26	0.1	G. Hobona	all	initial version
------------	-----	-----------	-----	-----------------

BIBLIOGRAPHY

[1] Vretanos, P.A, Kralidis, T., Heazel, C.: OGC 20-004: Draft OGC API – Records – Part 1: Core, <http://docs.ogc.org/DRAFTS/20-004.html>

[2] Portele, C., Vretanos, P.A: OGC 21-045: OGC Features and Geometries JSON – Part 1: Core, <https://docs.ogc.org/DRAFTS/21-045.html>

[3] STAC Community: SpatioTemporal Asset Catalog, <https://stacspect.org/en>

[4] Holmes, C.: Static SpatioTemporal Asset Catalogs in Depth, Available at <https://medium.com/radiant-earth-insights/static-spatiotemporal-asset-catalogs-in-depth-710530934a84>

Top

Thank You

Community

500+ International Members
110+ Member Meetings
60+ Alliance and Liaison partners
50+ Standards Working Groups
45+ Domain Working Groups
25+ Years of Not for Profit Work
10+ Regional and Country Forums

Innovation

120+ Innovation Initiatives
380+ Technical reports
Quarterly Tech Trends monitoring

Standards

65+ Adopted Standards
300+ products with 1000+ certified implementations
1,700,000+ Operational Data Sets
Using OGC Standards

