Testbed-12 Web Service Implementation Engineering Report

Table of contents

Publication Date: 2017-05-12

Approval Date: 2016-12-07

Posted Date: 2016-11-15

Reference number of this document: OGC 16-027

Reference URL for this document: http://www.opengis.net/doc/PER/t12-A090

Category: Public Engineering Report

Editor: Johannes Echterhoff, Clemens Portele

Title: Testbed-12 Web Service Implementation Engineering Report

---

Testbed-12 Web Service Implementation Engineering Report (16-027)

COPYRIGHT

WARNING

LICENSE AGREEMENT

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

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

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

This license is effective until terminated. You may terminate it at any time by

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

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

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

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

Abstract

This document is a deliverable of the OGC Testbed-12. It describes the results of analyzing the Testbed-12 web service implementations.

OGC has been developing web service specifications since the OGC Web Mapping Testbed in 1999. In particular, the original OGC Web Map Service specification has been developed during that testbed. 17 years later most current OGC web service standards still follow the general approach that had been developed in 1999 (the capabilities document, the remote procedure call via HTTP paradigm, etc).

Over time, the OGC web service approach has been amended and extended in different ways by different OGC standards and profiles. In addition, some of the more flexible mechanisms have been used in practice in different ways by different software vendors or communities. The OGC Web Service Common standard had been a response by OGC to these developments and aimed at maintaining a consistent approach across the different OGC web service standards. However, this effort has been only partially successful for several reasons, including shortcomings in the OWS Common standard, the existence of multiple incompatible OWS Common versions and a reluctance by working groups and communities to introduce incompatible changes to existing service types in order to harmonize. All attempts in recent years to continue the work on OWS Common have not seen much traction. While there seems to be general agreement that the current situation is not optimal and that consistency is desirable, it is unclear how to improve in a way that meets market demands.

This document summarizes information about the web service implementations in Testbed-12. It is not and should not be understood as a general analysis or assessment of the OGC web service architecture, but a low-key effort to gain some insights from looking at a significant number of web service implementations and their use in interoperability experiments and demos.

During the years since 1999 not only the OGC standards baseline has evolved, but also the Web itself. The W3C has been working on identifying Best Practices for Data on the Web and W3C and OGC are jointly working on extending this with Best Practices for Spatial Data on the Web. The analysis also includes an assessment about the OGC approach to web services with respect to the draft best practices at the time of writing of this report.

To the extent possible, we draw conclusions and recommendations from the

information that has been gathered. These fall into three categories:

- Improving the interoperability of OGC web services as they are today
- Support for new requirements in a consistent way across service types
- Improvements to the standardization process

In addition, there is also a specific case that does not fit into these general categories.

Business Value

This document provides information that should be valuable for the Architecture Domain Working Group and the OWS Common Standards Working Group when discussing the evolution of the architecture underpinning the OGC web service standards.

The context of the work is described in the Abstract.

Technology Value

See the sections Abstract and Business Value.

Keywords

OGC, Testbed-12, Web Services

Proposed OGC Working Group for Review and Approval

OGC Architecture DWG

Chapter 1. Introduction

1.1. Scope

This OGC® Engineering Report documents the following information for each Web service implementation in the OGC Testbed-12:

- specifications / profiles / conformance classes / options implemented;
- specific requirements, where known;
- security technologies used, if any.

Based on this overview, the report documents - to the extent that the information is available to the authors - the following information:

- which capabilities are really used by the clients (including how OGC Capabilities documents are used);
- interoperability issues that were encountered during TIEs;
- the results of CITE tests executed on the service, if any.

For web services that make spatial data available on the Web, the document will also discuss in how far implementations follow the (draft) best practices identified by the W3C/OGC Spatial Data on the Web working group.

Recommendations for consideration by the OGC membership are identified, where possible.

This report does not document the entire Testbed-12 architecture.

1.2. Document Contributor Contact Points

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

Table 1. Contacts

Name	Organization
Johannes Echterhoff	interactive instruments
Clemens Portele	interactive instruments

1.3. Future Work

No future work is planned to this document. It is expected that this document will inform the OGC Architecture DWG about general issues and recommendations regarding OGC web services. This may result in changes in other documents.

Some of the recommendations could be addressed in future OGC testbeds. An example is the recommendation regarding SDI Profiling.

1.4. Foreword

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.

Chapter 2. References

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

OGC Engineering Reports:

- OGC Testbed-6 SWE Event Architecture Engineering Report, OGC document number 09-032
- OGC Testbed-12 Asynchronous Services Response Engineering Report, OGC document number 16-023
- OGC Testbed-12 LiDAR Streaming Engineering Report, OGC document number 16-034
- OGC Testbed-12 Web Integration Service Engineering Report, OGC document number 16-043
- OGC Testbed-12 Data Broker Engineering Report, OGC document number 16-045
- OGC Testbed-12 OWS Common Security Extension Engineering Report, OGC document number 16-048
- OGC Testbed-12 Imagery Quality and Accuracy Engineering Report, OGC document number 16-050
- OGC Testbed-12 Compression Techniques Engineering Report, OGC document number 16-055
- OGC Testbed-12 Semantic Portrayal, Registry and Mediation Engineering Report, OGC document number 16-059
- OGC Testbed-12 Catalogue and SPARQL Engineering Report, OGC document number 16-062

NOTE OGC Engineering Reports can be found at http://www.opengeospatial.org/docs/er

OGC standards:

- OpenGIS Filter Encoding 2.0 Encoding Standard, OGC document number 09-026r1
- OGC Web Services Common
  - Version 1.0.0: OpenGIS Web Services Common Specification, OGC document number 05-008
  - Version 1.1.0: OGC Web Services Common Specification, OGC document number 06-121r3
  - Version 2.0.0: OGC Web Services Common Standard, OGC document number 06-121r9
- OGC Publish/Subscribe Interface Standard 1.0 - Core, OGC document number 13-131r1
- OGC Sensor Planning Service Implementation Specification 1.0, OGC document number 07-014r3
- OGC Sensor Planning Service Implementation Standard 2.0, OGC document number 09-000
- Styled Layer Descriptor profile of the Web Map Service Implementation Specification, OGC document number 05-078r4

NOTE OGC Standards can be found at http://www.opengeospatial.org/docs/is

OGC supporting documents:

- OGC Web Notification Service, OGC document number 06-095
- Technical Committee Policies and Procedures, OGC document number 05-020r24

DGIWG standards:

- DGIWG – Web Coverage Service Profile, edition 0.5.0
- DGIWG - Web Feature Service 2.0 Profile, edition 2.0.0
- DGIWG – Web Map Service 1.3 Profile – Revision, edition 2.1.0

NSG standards:

- NSG Web Feature Service Implementation Profile, version 0.0.9
- NSG Web Feature Service Implementation Profile, version 0.2.0
- NSG Web Map Service Implementation Profile, version 1.3.0
- NSG Web Map Tile Service 1.x.x Implementation Interoperability Profile, version 1.x.x

ISO standards:

- ISO 19115 - Geographic information - Metadata, first edition (2003)

Chapter 3. Conventions

3.1. Abbreviated Terms

API	Application Programming Interface
CSW	Catalogue Services for the Web
DGIWG	Defense Geospatial Information Working Group
ER	Engineering Report
EXI	Efficient XML Interchange
FES	Filter Encoding Standard
KVP	Key-Value Pairs
NSG	National System for Geospatial-Intelligence
POX	"Plain Old XML"
PubSub	Publish and Subscribe
SOA	Service Oriented Architecture
SOS	Sensor Observation Service
SWG	Standards Working Group
T12	Testbed-12
TIE	Technical Integration Experiment
W3C	World Wide Web Consortium
WFS	Web Feature Service
WMS	Web Map Service
WMTS	Web Map Tiling Service
WOS	Web Object Service
WSDL	Web Services Description Language

Chapter 4. Overview

OGC has been developing web service specifications since the OGC Web Mapping Testbed in 1999. In particular, the original OGC Web Map Service specification has been developed during that testbed. 17 years later most current OGC web service standards still follow the general approach that had been developed in 1999 (the capabilities document, the remote procedure call via HTTP paradigm, etc).

Over time, the OGC web service approach has been amended and extended in different ways by different OGC standards and profiles. In addition, some of the more flexible mechanisms have been used in practice in different ways by different software vendors or communities. The OGC Web Service Common standard had been a response by OGC to these developments and aimed at maintaining a consistent approach across the different OGC web service standards. However, this effort has been only partially successful for several reasons, including shortcomings in the OWS Common standard, the existence of multiple incompatible OWS Common versions and a reluctance by working groups and communities to introduce incompatible changes to existing service types in order to harmonize. All attempts in recent years to continue the work on OWS Common have not seen much traction. While there seems to be general agreement that the current situation is not optimal and that consistency is desirable, it is unclear how to improve in a way that meets market demands.

A large number of web services were implemented in OGC Testbed-12. This report summarizes information about these components and documents the results of a general analysis. It is not and should not be understood as a general analysis or assessment of the OGC web service architecture, but a low-key effort to gain some insights from looking at a significant number of web service implementations and their use in interoperability experiments and demos.

The information retrieved from the web service implementations, extracted from Draft Engineering Reports and received from Testbed participants is documented in Clause 5.

During the years since 1999 not only the OGC standards baseline has evolved, but also the Web itself. The W3C has been working on identifying Best Practices for Data on the Web and W3C and OGC are jointly working on extending this with Best Practices for Spatial Data on the Web. The analysis in Clause 6 documents a brief assessment about the OGC approach to web services with respect to the draft best practices at the time of writing of this report.

The information in Clauses 5 and 6 was analyzed to derive conclusions and recommendations. These are documented in Clause 7.

Chapter 5. Web Service Implementations

5.1. Overview

This chapter documents information about the Testbed-12 web service implementations. The information was collected as follows:

- Metadata from Testbed-12 web services was harvested and parsed (using XQuery) to gather information such as versions, profiles, conformance classes, and options supported by these services.
- Client and service component providers were asked to provide feedback on specific aspects, for example results of CITE tests, implemented security technologies, use of capabilities documents, and encountered interoperability issues. The feedback was analyzed and summarized.

Documents based on the OWS Common standard supported the application of a common parsing function. Likewise, the "Filter_Capabilities" section defined by the OGC Filter Encoding standard is re-used by multiple OGC web services (in Testbed-12, these were CSW, PubSub, SOS, WFS, and WOS instances). It was therefore also parsed with a common function. For other documents, and to gather service type specific information from the capabilities document, additional functions were applied.

The following information was parsed from capabilities elements defined by OWS Common:

- Name of the service provider
- Version of the capabilities document
- Specification versions supported by the service
- Profiles supported by the service
  - The information was retrieved from "Profiles" elements in the "ServiceIdentification" section of the capabilities document.
- Operations supported by the service, including:
  - HTTP methods (GET, POST)
  - Security constraints - for further details, see the Testbed-12 OWS Common Security Extension ER
  - Aysnchronous response handling capabilities - for further details, see the Testbed-12 Asynchronous Services Response ER
  - Support for EXI compression - for further details, see the Testbed-12 LiDAR Streaming ER and the Testbed-12 Compression Techniques ER
- If extended capabilities were defined

Information parsed from FES filter capabilities includes supported:

- conformance classes
- identifier filtering capabilities

- logical, comparison, spatial, temporal, and any additional operators
- functions
- geometry operands
- temporal operands

NOTE Documentation of service type specific information that was parsed from service metadata can be found in the following sections. Note that the sections do not directly contain the tables with this information, since the tables are often too big for inclusion in a document with usual page size. Instead, the sections contain links to the tables in the HTML version of this report.

NOTE Testbed-12 component providers may not always have populated the capabilities documents of their services with correct values for the fields that are automatically parsed for this analysis. This can be due to time and resource constraints in Testbed-12 but also to creating an accurate capabilities document not being part of their tasks in this Testbed. Consequently, the tables presented in the following sections may not be complete. This should be taken into account when reviewing these tables.

5.2. Catalogue Service (CSW)

The following table was created using the common parsing functions described in the overview.

Results of parsing service metadata of Testbed-12 CSW instances

The table is available online.

NOTE
- The service provided by CNR also implements PubSub functionality. More specifically, it implements the "Basic Publisher" conformance class.

5.3. PubSub Publisher

The following table was created using the common parsing functions described in the overview.

Results of parsing service metadata of Testbed-12 PubSub Publisher instances

The table is available online.

NOTE

- The 52N PubSub service implemented the OGC PubSub 1.0 SOAP Extension. This information was not listed in the Capabilities document (under ows:ServiceIdentification/ows:Profile) and therefore does not appear in the auto-generated table).
- The CSW service provided by CNR also implements PubSub functionality. More specifically, it implements the "Basic Publisher" conformance class. It is not listed in the table, since the capabilities document of the service is based on CSW 2.0.2. The integration of multiple OGC service interfaces at the same endpoint is an aspect of the recommendation to pursue development of a mechanism to discover links between services as well as data and services.

5.4. Semantic Services

The set of web services developed and deployed in Testbed-12 includes four services focusing on semantic processing:

- Semantic Registry Service
- Semantic Portrayal Service
- Semantic Mediation Service

Automated processing of service metadata was not performed for these services, because they are new services that have been developed in the OGC IP Program (Testbed-11 and Testbed-12) and do not extend any established OGC web service standard. Detailed information about the services can be found in the Testbed-12 Semantic Portrayal, Registry, and Mediation ER.

5.5. Sensor Observation Service (SOS)

The following table was created using the common parsing functions described in the overview.

Results of parsing service metadata of Testbed-12 SOS instances

The table is available online.

5.6. Web Coverage Service (WCS)

The following table was created using the common parsing functions described in the overview. In addition:

- Support of the DGIWG WCS (draft) Profile was identified
- For a WCS v1.0.0 - whose capabilities document is not based on OWS Common, parsed information includes the:
  - Name of the service provider
  - Version of the capabilities document
  - Specification versions supported by the service
  - Operations supported by the service

Results of parsing service metadata of Testbed-12 WCS instances

The table is available online.

5.7. Web Feature Service (WFS)

The following table was created using the common parsing functions described in the overview. In addition:

- Support of the DGIWG WFS profile and NSG WFS profiles was identified
- Supported conformance classes were identified

Results of parsing service metadata of Testbed-12 WFS instances

The table is available online.

NOTE

- The WFS with ID WFS_Luciad_F006 is part of a Data Broker service that facades multiple services. For further details, see the Testbed-12 Data Broker Engineering Report.

5.8. Web Integration Service (WIS)

The following table was created using the common parsing functions described in the overview.

Results of parsing service metadata of Testbed-12 WIS instances

The table is available online.

5.9. Web Map Service (WMS)

Since the capabilities document of WMS is not based on OWS Common, the common parsing functions described in the overview could not be applied to retrieve relevant information from the service metadata. Instead, a specific function was implemented that retrieves the following pieces of information for each of the Testbed-12 services (listed in the following table):

- Name of the service provider
- Version of the capabilities document
- Support of the DGIWG WMS, the NSG WMS, and the OGC SLD WMS profiles was identified
- Operations supported by the service

Results of parsing service metadata of Testbed-12 WMS instances

The table is available online.

NOTE

- The WMS with ID WMS_Luciad_F006 is part of a Data Broker service that facades multiple services. For further details, see the Testbed-12 Data Broker Engineering Report.

5.10. Web Map Tile Service (WMTS)

The following table was created using the common parsing functions described in the overview. In addition:

- Support of the NSG WMTS profile was identified

Results of parsing service metadata of Testbed-12 WMTS instances

The table is available online.

5.11. Web Object Service (WOS)

The following table was created using the common parsing functions described in the overview.

Results of parsing service metadata of Testbed-12 WOS instances

The table is available online.

5.12. Web Processing Service (WPS)

The following table was created using:

- the common parsing functions described in the overview to be applied for non-RESTful WPS instances
- a specific function to parse the following information from the service metadata of a RESTful WPS (which is encoded in JSON that has been automatically converted to XML for processing with XQuery):
  - name of the service provider
  - version of the capabilities document
  - specification versions supported by the service
  - profiles supported by the service
  - WPS operations supported by the service
  - if extended capabilities were defined
  - the fact that a WPS instance is a RESTful WPS

Results of parsing service metadata of Testbed-12 WPS instances

The table is available online.

5.13. Security Technologies

Three types of security technology were used by Testbed-12 web services:

- HTTP Basic Authentication
- HTTPS with Client-Side-Certificate

- HTTPS + SOAP + WS-Security

For more details about the security technologies, see the Testbed-12 OWS Common Security Extension ER.

The following listing documents which technology was implemented by which services:

- HTTP Basic Authentication
  - CSW_Galdos_A053_F003
  - PubSub_52N_F009
  - WCS_CubeWerx_A045_Flood_sims_AND_surge_H
  - WCS_CubeWerx_A045_Flooding_MODIS
  - WCS_CubeWerx_A045_Floodzone_sealevel_rise
  - WCS_CubeWerx_A045_GPM_Imerge_20160201
  - WCS_CubeWerx_A045_GPM_Imerge_20160219
  - WCS_CubeWerx_A045_LiDAR_2m_SFSU
  - WCS_CubeWerx_A045_NED_elevation
  - WCS_CubeWerx_A045_Satellite_Soil_Moisture
  - WCPS_SecureDimensions_rasdaman
  - WFS_CubeWerx_A007
  - WFS_CubeWerx_A011
  - WFS_CubeWerx_A037
  - WFS_CubeWerx_A065
  - WFS_Luciad_F006
  - WFS_mclick_F005
  - WFS_mclick_F010
  - WFS_mclick_F012
  - WFS_TE_mclick
  - WFS_TE_mclick_E006
  - WFS_Snowflake_F005
  - WFS_Snowflake_F010
  - WFS_Snowflake_F012
  - WIS_Compusult_A019
  - WMS_CubeWerx_A040
  - WMS_CubeWerx_A041
  - WMTS_CubeWerx_A042
- HTTPS with Client-Side-Certificate

    - WCS_SecureDimensions_DigitalGlobe
    - WFS_SecureDimensions_DigitalGlobe
    - WMTS_SecureDimensions_DigitalGlobe
- HTTPS + SOAP + WS-Security
    - WFS_SecureDimensions_CubeWerx_A004

In addition, some services implemented specific security measures:

- Most of the services deployed by CubeWerx support both HTTP Basic Authentication and "CwAuth credentials".
- A proxy service acts as a Policy Enforcement Point (PEP) for CSW_Envitia_A051, preventing specific operations from being executed with specific parameters. The proxy service can also be set to act as an IP filter.

NOTE	This list of additional security measures is based on information that was available on the Testbed-12 components twiki page and that was received as feedback to a questionnaire from component providers; the list may therefore not be complete.

Chapter 6. OGC Web Services and the Best Practices for Spatial Data on the Web

Access to data is increasingly happening over the Web and it is clear that the trend will continue. OGC started using web services already in the late 1990s and has significant experience in this regard. However, the Web and the related practices and expectations with respect to data “on the Web” have evolved during this period, too.

Through joint work by W3C and OGC, best practices for publishing Spatial Data on the Web are currently being documented. This section documents a brief assessment how the typical web service implementation based on an OGC standard (using KVP and POX bindings) that provides data on the Web (i.e. a WMS, WMTS, WFS, WCS, SOS, or CSW) fits with the emerging best practices (or not).

6.1. Data on the Web Best Practices

The Spatial Data on the Web Best Practices extend the generic Data on the Web Best Practices with additional aspects related to geographic information. The Data on the Web Best Practices are applicable to geographic information, too.

This proposed W3C standard is in the last stage of the consensus process (“Candidate Recommendation”). The latest version is available at https://www.w3.org/TR/dwbp. In total, 35 best practices are identified. The following table lists each best practice and a brief assessment about its relevance for OGC and whether the OGC web services implement the best practice.

Best Practice	Assessment
1: Provide metadata Provide metadata for both human users and computer applications. 2: Provide descriptive metadata Provide metadata that describes the overall features of datasets and distributions.	Capabilities documents provide metadata for applications, not for human users. Additional metadata may be available through other operations. The metadata about the data is in general limited to information that is needed to compile requests to retrieve the data. Other metadata (typically in a catalogue) is referenced from the capabilities, but often it is not consistent or reliable what is referenced and applications cannot rely on metadata being provided. In practice, the metadata in the capabilities is often also inconsistent with the referenced dataset metadata in catalogues (different contact information, bounding boxes, etc.).
3: Provide structural metadata Provide metadata that describes the schema and internal structure of a distribution.	Implemented through capabilities (feature types, layers, etc.) and through additional operations (DescribeFeatureType, DescribeRecord, DescribeCoverage, etc.).

Best Practice	Assessment
4: Provide data license information Provide a link to or copy of the license agreement that controls use of the data.	Implemented in general, but in a very weak way. Fees and access constraints as free text fields. Often not set (correctly).
5: Provide data provenance information Provide complete information about the origins of the data and any changes you have made. 6: Provide data quality information Provide information about data quality and fitness for particular purposes.	Not implemented in general. While the metadata includes fields for lineage information and data quality, information about the origins, the processing and the data quality are currently often not included.
7: Provide a version indicator Assign and indicate a version number or date for each dataset.	Not implemented in general, services do not provide version information about the content they provide. If available, it may be part of the reference metadata.
8: Provide version history Provide a complete version history that explains the changes made in each version.	Currently not implemented. No information about the change history in the dataset is provided.
9: Use persistent URIs as identifiers of datasets Identify each dataset by a carefully chosen, persistent URI. 10: Use persistent URIs as identifiers within datasets Reuse other people's URIs as identifiers within datasets where possible. 11: Assign URIs to dataset versions and series Assign URIs to individual versions of datasets as well as to the overall series.	Not really implemented, in general. There will be URIs that represent datasets or more fine-grained elements (features, records, etc. through the OWS Common GetResourceById request or similar), but persistence is not considered and in practice these URIs will change with time (new service URLs, changes in how the data is organized, changes in the service version supported by a service, etc.)
12: Use machine-readable standardized data formats Make data available in a machine-readable, standardized data format that is well suited to its intended or potential use.	Implemented. XML is used in OGC web services. Services may support other formats, too, but the capabilities are always in XML.
13: Use locale-neutral data representations Use locale-neutral data structures and values, or, where that is not possible, provide metadata about the locale used by data values.	In general not implemented in OGC web services.
14: Provide data in multiple formats Make data available in multiple formats when more than one format suits its intended or potential use.	Implemented, all service specifications support providing multiple formats. However, HTTP content negotiation is not supported, instead special parameters like outputFormat are used.
15: Reuse vocabularies, preferably standardized ones Use terms from shared vocabularies, preferably standardized ones, to encode data and metadata.	Implemented. ISO and OGC provide a number of vocabularies / schemas that are reused in the OGC web services including Filter Encoding, GML, ISO 19139 metadata, etc.
16: Choose the right formalization level Opt for a level of formal semantics that fits both data and the most likely applications.	This assessment is in the eye of the beholder…

Best Practice	Assessment
17: Provide bulk download Enable consumers to retrieve the full dataset with a single request. 18: Provide Subsets for Large Datasets If your dataset is large, enable users and applications to readily work with useful subsets of your data.	In some cases, this may be supported, but in general the OGC web services aim at providing selected subsets of datasets through queries. The full dataset is often large.
19: Use content negotiation for serving data available in multiple formats Use content negotiation in addition to file extensions for serving data available in multiple formats.	Not implemented. OGC web services do not support content negotiation, but use OGC-specific mechanisms like a parameter outputFormat.
20: Provide real-time access When data is produced in real time, make it available on the Web in real time or near real-time.	Can be supported by all service types. PubSub supports active notifications, updates too.
21: Provide data up to date Make data available in an up-to-date manner, and make the update frequency explicit.	It depends on the provider, how up-to-date a dataset is.
22: Provide an explanation for data that is not available For data that is not available, provide an explanation about how the data can be accessed and who can access it.	In general, not supported. GML and ISO 19139 metadata support nil values with an associated reason providing additional information.
23: Make data available through an API Offer an API to serve data if you have the resources to do so.	This is what OGC web services do.
24: Use Web Standards as the foundation of APIs When designing APIs, use an architectural style that is founded on the technologies of the Web itself.	Most of the current OGC web service standards predate the Web as it is used today and thus are founded only partially on the technologies of the Web. The services often do not conform to HTTP 1.1 although it is a normative reference. In recent years, REST bindings are being added to several of the OGC web service standards, too.
25: Provide complete documentation for your API Provide complete information on the Web about your API. Update documentation as you add features or make changes.	All versions of the standards are published on the OGC website, including revision notes.
26: Avoid Breaking Changes to Your API Avoid changes to your API that break client code, and communicate any changes in your API to your developers when evolution happens.	Reflected in the OGC policies.
27: Preserve identifiers When removing data from the Web, preserve the identifier and provide information about the archived resource. 28: Assess dataset coverage Assess the coverage of a dataset prior to its preservation.	Not addressed, in general.

Best Practice	Assessment
29: Gather feedback from data consumers Provide a readily discoverable means for consumers to offer feedback. 30: Make feedback available Make consumer feedback about datasets and distributions publicly available.	New work items in OGC are looking into this topic (the proposed OGC Quality of Service and Experience Domain Working Group and the Geospatial User Feedback Standards Working Group).
31: Enrich data by generating new data Enrich your data by generating new data when doing so will enhance its value.	It depends on the provider, how the service is set up.
32: Provide Complementary Presentations Enrich data by presenting it in complementary, immediately informative ways, such as visualizations, tables, Web applications, or summaries.	Not addressed by OGC standards and typically not implemented for OGC web services.
33: Provide Feedback to the Original Publisher Let the original publisher know when you are reusing their data. If you find an error or have suggestions or compliments, let them know. 34: Follow Licensing Terms Find and follow the licensing requirements from the original publisher of the dataset. 35: Cite the Original Publication Acknowledge the source of your data in metadata. If you provide a user interface, include the citation visibly in the interface.	It depends on the provider, how the service is set up.

6.2. Spatial Data on the Web Best Practices

The Spatial Data on the Web Best Practices currently under development by a joint working group of W3C and OGC extend the Data on the Web Best Practices with additional guidance specific to spatial data. This is work in progress. The final version is expected during the first half of 2017.

During the development, the latest Editor's Draft is available at http://w3c.github.io/sdw/bp/. The table below uses the 15 best practices included in the latest Working Draft: https://www.w3.org/TR/2016/NOTE-sdw-bp-20161025/ (the 3 best practices in the "placeholder" section are ignored for now). Note that the best practices will change.

Best Practice	Assessment
1: Include spatial metadata in dataset metadata The description of datasets that have spatial features should include explicit metadata about the spatial coverage	Supported. See also Data on the Web Best Practices 1 and 2.
2: Provide context required to interpret data values Data values should be linked to spatial, temporal and thematic information that describes them.	Implemented.

Best Practice	Assessment
3: Specify Coordinate Reference System for high-precision applications A coordinate referencing system (CRS) should be specified for high-precision applications to locate geospatial entities.	Implemented, OGC has policy guidance related to the use and declaration of CRSs.
4: Make your data indexable by search engines Search engines should receive a metadata response to a HTTP GET when dereferencing the link target URI.	Not implemented.
5: Describe the positional accuracy of spatial data Accuracy and precision of spatial data should be specified in machine-interpretable and human-readable form.	See Data on the Web Best Practice 6.
6: How to describe properties that change over time Entities and their data should have versioning with time/location references	Supported, in general.
7: Use globally unique HTTP identifiers for spatial things Spatial things referred to within datasets should have unique, persistent HTTP or HTTP(S) URIs as identifiers.	See Data on the Web Best Practice 10.
8: Provide geometries on the Web in a usable way Geometry data should be expressed in a way that allows its publication and use on the Web.	Implemented through the use of GML (and other geometry encodings, if other formats than XML are used like WKT, GeoJSON, etc.).
9: How to describe relative positions Provide a relative positioning capability in which the entities can be linked to a specific position.	Not addressed, in general.
10: Use spatial semantics for Spatial Things The best vocabulary should be chosen to describe the available SpatialThings.	Implemented by using application schemas.
11: Expose spatial data through 'convenience APIs' If you have a specific application in mind for publishing your data, tailor the spatial data API to meet that goal.	Out of scope, in general, for standards-based services. However, this is to some extent supported through extension mechanisms, for example, stored queries.
12: Include search capability in your data access API If you publish an API to access your data, make sure it allows users to search for specific data.	Implemented.
13: Provide subsets for large spatial datasets Identify subsets of large spatial data resources that are a convenient size for applications to work with	Not addressed, in general.

Best Practice	Assessment
14: Publish links to related resources The data should be published with explicit links, including spatial links, to spatial things or other resources, both in the same dataset and in other datasets. 15: Use links to find related data Related data to a spatial dataset and its individual data items should be discoverable by browsing the links	Can be supported, but links are often not used in practice, in particular to information outside of a dataset.

Chapter 7. Conclusions & Recommendations

The information in Clauses 5 and 6 has been analyzed to identify potentially interesting facts or patterns and draw conclusions related to the web services implemented in Testbed-12. In addition, feedback from participants of Testbed-12 have been taken into account, too.

The main target of the conclusions and recommendations documented in this clause are the OGC membership and the OGC Architecture DWG in particular as well as future OGC testbeds.

The points raised in this clause should not be considered as results of a comprehensive analysis or as mature proposals that went through thorough scrutiny and in-depth discussions. They should be understood as one of the inputs to the continued discussions in the OGC membership related to the future evolution of the OGC standards baseline for geospatial web services. This report is an attempt to gain some insights from looking at a good number of web service implementations and their use in interoperability experiments and demos.

Whenever there is a recommendation for changes in standards, this has been done based on the assumption that such recommendations should only be considered during the next revision. The authors want to stress that none of the recommendations seems important enough to justify a revision just for the changes! Rather, the conclusions and recommendations should be seen as observations with regard to the future evolution of the OGC web service standards.

Therefore, none of the recommendations have been turned into Change Requests to specific OGC standards as there should be a broader discussion first.

Now that we have got this important message out of the way, we can take a closer look at the conclusions and recommendations. They have different charactistics and this clause is organized accordingly.

7.1. Improving Interoperability

Comparing the capabilities of the web services in Clause 5 where multiple implementations exist per service type using different software products, we can observe that in most cases - WMS is a notable exception - a client

- would have to support multiple service profiles due to a lack of a core profile,
- would have difficulties to exploit some aspects of the capabilities documents due to inconsistent ways in which information is provided.

For developers that have a good understanding of the OGC standards and sufficient resources this may be a smaller issue, but for someone who justs wants to use some geospatial web service as a likely minor part of a larger application this could make using location-based information more complicated than it would have to be (in a perfect world).

Regarding the first item, for most service types we can see different implementation choices and a client in general has to support all of them, unless it knows in advance the types of implementations it will use. Examples for the choices are:

one service supports HTTP GET for an operation, another HTTP POST

- different implementations support different versions of a standard, in some cases including only quite old versions
- one service may only support spatial queries by point, others by bounding box
- etc.

As mentioned above, this applies to most service types with a very long history in OGC (WFS, WCS, CSW), but less for WMS where all services in Testbed-12 support the same non-deprecated version (1.3.0), all support HTTP GET for the core operations, etc. This seems to be consistent with the experience that of all OGC web services WMS seems to have the least interoperability issues in practice. Of course, it is also the simplest of the services.

In the next sub-clauses, some specific observations and recommendations are documented.

7.1.1. Follow Web Rules

Recommendation: When implementing an OGC web service, follow the rules of the Web. In particular, conform to the HTTP standard. Consider to provide HTML, too.

When the initial OGC web service standards were developed, "web services" mostly meant using remote procedure calls using HTTP as a transport mechanism for request and response messages. However, the expectations are different today and following the practices of the Web (see also Clause 6) and the requirements as well as the spirit of the HTTP standard are important. The current activities related to RESTful OGC web service bindings reflect this evolution.

However, it is not uncommon for OGC web service implementations for the other bindings (KVP: Key-Value pairs, POX: plain old XML) to be inconsistent with the HTTP standard. This includes incorrect content-type headers, lack of support for the mandatory HEAD operation, etc. The related requirements are not explicitly documented in the OGC web service standards, but in all cases they are "imported" as HTTP 1.1 is a normative reference in all the OGC web service standards implemented in Testbed-12.

For example, TIE testing in Testbed-12 revealed a small interoperability issue: a WMS returned an exception (that access was not allowed) as an image, without specifying a content type in the response. The client was not able to correctly handle this particular case.

This is one example where web standards and rules were not followed by a web service implementation. HTTP relevant information needs to be set correctly in HTTP messages, like status codes and content type in service responses. Likewise, OGC standards should clearly specify relevant aspects, for example, the content types to use in responses, where appropriate.

It may also be an option to include relevant assertions in CITE tests for OGC web services to improve the consistency of web service implementations of OGC standards with the rules of the Web.

Providing HTML responses in addition to XML would support human users and search engines, too. A simple start would be to provide a HTML response with information about the service, if the base URL of the service is retrieved without parameters, or if the capabilities document is requested with HTTP headers indicating a preference for an HTML representation.

7.1.2. Use CITE Tests for Testbed Services

Recommendation: Perform available CITE tests on services developed during a testbed before TIEs take place.

The CITE tests are an important tool for the community as a basic test for an implementation, at least in the cases where the CITE tests can be used with any implementation and do not require loading a specific test dataset. The WFS tests are examples.

For example, the following interoperability issue was discovered - and fixed - in a TIE in Testbed-12: the schema returned by a WFS upon a DescribeFeatureType request did not match the GML features returned by the service. This was problematic for a consuming entity which uses schema-based parsing (and therefore needs a correct DescribeFeatureType response to be able to decode data).

CITE tests should be able to detect such issues. Surprisingly the CITE tests were apparently not used to test any of the web service implementations in Testbed-12 (except in the conformance testing thread, of course). Therefore, we recommend to perform available CITE tests on services developed during a testbed. These tests should be performed before TIEs take place. This can also help improve the CITE tests.

7.1.3. Conformance Statements

Recommendation: Agree on a common mechanism to publish conformance statements for an OGC web service.

One aspect of the web service implementation analysis was to identify the specifications, conformance classes, and profiles supported by the Testbed-12 services. It turned out that the required information is provided in different ways:

- Some services publish the conformance classes they support in their capabilities document, within "Profile" elements of the "ServiceIdentification" section. Each "Profile" element contains a URI that identifies a conformance class that the service supports. A good example can be found in the table with parsed information on Sensor Observation Services. It shows that a service can list conformance classes defined by the service standard it implements, but also conformance classes defined for the data it serves as well as extensions. WCS 2.0 also follows this approach.
- Web Feature Services primarily declare conformance statements through "Constraint" elements in the capabilities document - in the "OperationsMetadata" section (for WFS conformance classes) and in the "Filter_Capabilities/Conformance" section (for FES conformance classes). Services that support the (draft) WFS Temporality Extension, however, specify conformance with that extension in "Profile" elements of the "ServiceIdentification" section.
- Some of the services developed in Testbed-12 were required to implement DGIWG and NSG profiles for WCS, WFS, WMS, and WMTS. A single service declared conformance with http://www.dgiwg.org/service/wfs/1.0/profile/basic via a "Profile" element in the "ServiceIdentification" section. None of the services did so for the conformance class with URI http://www.nga.mil/service/wfs/1.0/profile/basic. However, some WFSs announced compliance with DGIWG and NSG profiles through an "Abstract" element in their capabilities document,

which provides a human readable description of the service. These abstracts contained statements that were actually required by the profiles, such as "This server implements the DGIWG BASIC WFS profile of WFS 2.0". The formulation of the abstracts in the service metadata was sometimes slighty different than required by the profiles, which makes the abstract a rather week solution for automatically parsing declarations of conformance. Then again, only some of the profiles declared URIs for conformance classes (which could be used to declare conformance via "Profile" elements, for example for WMTS) - and therefore the abstract was used as a fallback solution to identify compliance with the profiles.

Additional observations and comments:

- The definition of the "ServiceIdentification/Profile" element in table 11 of OWS Common 2.0 is: "Identifier of OGC Web Service (OWS) Application Profile". According to this definition, the "Profile" element was not intended to be used for declaring all sorts of conformance claims (for example about data served by the service).
- OGC service standards that are not based on OWS Common, such as WMS 1.3.0, do not support a "Profile" element to indicate identifiers of conformance classes in their capabilities document.
- The metadata of RESTful OGC services will likely not always contain an "OperationsMetadata" section. In these cases, the declaration of conformance via "Constraint" elements in that section - like it is done by WFSs - would not be possible.

Conclusion:

OGC web services use different ways to make conformance statements. A common solution to do so would facilitate service discovery across service types, more specifically: the discovery of services that support the functionality required by a given use case. This is important in an architecture, where a client wants to find suitable services (by inspecting service metadata published in a registry or catalogue) before binding them.

As described before, each of the current solutions has a drawback. Therefore, the OGC should explore the possibilities of a common approach to publishing conformance statements for an OGC web service.

7.1.4. SDI Profiling

Recommendation: Consider development of SDI profiling capability.

This report provides an overview of the features and capabilities that are offered by the web services implementations in Testbed-12. An attempt was made to identify which of these capabilities were actually used by client components to construct service requests. The only viable approach in Testbed-12 to get the necessary information was to ask the client component developers. However, feedback was minimal, which is an obstacle for drawing conclusions or for developing recommendations.

Observing the variations in how information is provided in practice in capabilities, it seems reasonable to assume that most clients will only process a subset of the capabilities provided by a web service.

One approach to gather more information about the service capabilities used by clients to construct

service requests would be to monitor the messages during TIEs.

In software development, *profiling* is a way to identify which parts of a program are actually used when executing it. Profiling also helps identifying how often certain methods are invoked, and how much time is spent during their execution. This information helps understanding the program.

When applied to a Spatial Data Infrastructure based on OGC standards, *profiling* could help understanding how the various SDI components are used, how they communicate, and which functions and capabilities are used - or not used at all. Profiled information about requests and responses could be analyzed to identify usage patterns and trends. It would also help understanding which versions of OGC standards are in use, as well as which specific capabilities supported by them are used.

An SDI profiling capability could be useful to providers of OGC services as well as the OGC in general:

- For service providers, especially in an environment where the clients that use a service are not pre-defined, such as in INSPIRE and GEOSS, service profiling would provide information which capabilities of a service are used and which are not used. This gives an indication about which capabilities are essential for clients and therefore must be maintained, and which capabilities (especially from older versions of the service specification) can be deprecated or removed in future iterations of the service deployment. Knowing which service capabilities are essential for clients, and which capabilities are nice-to-have, can also be useful when developing standard profiles and procuring service software.
- In an OGC testbed, service profiling could help identifying which capabilities of a specific OGC service standard have been used in TIEs and demos. This can be useful input for deciding if (a specific version of) an OGC standard - or specific capabilities defined by it - can be deprecated or retired/removed. Service profiling could be done using a proxy between client and service in TIEs, similar to a proxy for access control.

7.2. New Requirements

Testbeds explore potential new capabilities and Testbed-12 is no exception. This sub-clause captures observations related to such cases.

7.2.1. HTTP Headers with Metadata Links

Recommendations:

1. Develop a common mechanism to provide metadata for data whose format does not allow inclusion of metadata (like some image formats).
2. Re-use available Web solutions where appropriate.

The Testbed-12 Imagery Quality and Accuracy ER describes how HTTP headers can be used to communicate metadata about a resource (in this case, an image tile returned by a WMTS). Two alternatives are presented:

a. One approach is to encode the metadata within an HTTP header.

b. The other approach is to use an HTTP header to link to the actual metadata.

The ER describes some limitations with providing the metadata directly in HTTP headers. Approach b appears to be the better alternative:

- It supports the provision of metadata in any format (XML, RDF, CSV, ...).
- It supports metadata of any length and complexity.
- It allows pointing to registries/catalogues (that could serve as a single point of entry to metadata harvested from multiple stores).

In general, the approach is useful to convey metadata for data whose format cannot convey the metadata itself (like some image formats, depending on the type of metadata). Such a mechanism would be useful for any OGC service that returns such data.

The ER suggests to establish the metadata link using a "Link" HTTP header as defined by IETF RFC 5988. The W3C Note “PROV-AQ: Provenance Access and Query” uses this header as well to link to provenance information (which is a kind of resource metadata). The Testbed-12 Imagery Quality and Accuracy ER correctly uses an available Web solution for the issue of communicating metadata about a resource. We recommend this course of action (since it helps preventing redundancy).

7.2.2. Advertising and Requesting Compression

Recommendation: Develop a common way to advertise and request compression of service responses.

In Testbed-12, support for EXI compression was advertised in different ways:

- A SOS advertised that it supports EXI compression by including a "Content-Type" constraint in the metadata of its operations, with value "application/exi".
- A WFS used the "outputFormat" parameter in the operation’s metadata - with values "application/json2exi", "json2exi", and "gml2exi" - to state that EXI compression is supported.

The Testbed-12 Compression Techniques ER recommends the development of a "Compression Profile for WFS". With EXI compression also having been investigated in Testbed-12 for another OGC service (SOS), we think that a common solution is necessary, and therefore recommend the development of a "Compression Profile for OWS".

7.2.3. Discovery of Links Between Services and Data

Recommendation: Pursue development of a mechanism to discover links between services as well as data and services.

The Testbed-12 data broker service (for details, see the Testbed-12 Data Broker ER) offers two OGC service interfaces, which happen to be accessible at the same endpoint/URL. The data broker can provide portrayal information for feature sets - either directly or by referencing services (WM(T)S, FPS) that can portray the features.

The service provided by CNR is another example for two OGC service interfaces being hosted at the same endpoint. The service implements both the CSW interface and the PubSub interface. An issue

there is that the PubSub core standard does not specify mechanisms for incorporating Publisher capabilities metadata into other OGC web services. Therefore, if a specific OGC service (like CSW) wanted to implement PubSub behavior, then an additional specification would be needed to define exactly how the PubSub capabilities metadata must be incorporated and advertised by that service.

The Testbed-12 Web Integration Service (WIS) (for details, see the Testbed-12 Web Integration Service ER) is a way to explore which services are offered at the same endpoint (which has similarities with WSDL). This could be a useful addition to the data broker. It could also be a solution for the combination of the PubSub interface with existing OGC services.

The common GetAssociations operation described in the Testbed-12 Web Integration Service ER is of more interest for this recommendation. The operation provides a way to explore which links exist between services as well as services and datasets. At the moment, these links are typically implicit. The GetAssociations operation makes this information explicit, making it accessible for harvesting by catalogues and discovery by clients.

The GetAssociations operation allows to, for example, indicate which services are cascaded by a given service. This allows a client to discover which services are used as data sources by that service - which can be of interest for data brokering.
The operation also allows to explore which features (provided by a WFS) are rendered by which WMSs. The GetAssociations operation does not provide a way to directly provide information for portraying a specific set of feature data retrieved from a WFS (as the Testbed-12 data broker does). However, it supports a scenario in which a client may have found WFSs with feature data the client is interested in, but the client prefers to directly portray the data via WMS before downloading the full set of feature data. Both approaches have unique benefits, and are not mutually exclusive.

The two Testbed-12 work items - data brokering and web integration - indicate that there is a need for a mechanism to discover links between services as well as data and services. The development of such a mechanism should be pursued. Additional use cases, and thus requirements, would help defining the mechanism.

7.2.4. Declaring New Functionality in Service Capabilities

NOTE	This recommendation is primarily intended for the OGC IP Program. That it applies to the OGC Standards Program is taken for granted.

Recommendation: When new functionality for OGC web services is developed during an OGC IP project, such as a Testbed, do not forget to develop a suitable way for the service to advertise that functionality.

In a testbed, there is often a hard-coded use-relationship between a client and a service component. In other words, the client can be pre-configured to use a specific web service. During our analysis, we recognized that this can lead to reduced attention to specifying how new functionality developed for the service (to be used by the client) should be declared by the service in its metadata.

Being able to identify that a service supports a particular function is important for dynamic discovery of and binding to that service - which is a key piece of the publish-find-bind paradigm of Service Oriented Architectures. This applies to RESTful and non-RESTful services.

If a service publishes what it is capable of, and also what a client may not be allowed to use (for example through security constraints), a client can a) determine if a service is useful for the task at hand and b) interact with the service using the features that are most useful for the task.

We acknowledge that an OGC testbed is primarily used to test and develop new functionality, typically as a proof-of-concept. This is only a first step to developing a new standard functionality. Maturation occurs in subsequent projects, which can be other OGC testbeds, but often is an OGC Standards Working Group (SWG). However, early development of a suitable way for declaring new service functionality in service metadata would facilitate subsequent steps of the maturation process.

7.2.5. Asynchronous Request-Response

Recommendation: Use features provided by the communication protocol to realize asynchronous request-response, if available. Otherwise, use an extension mechanism of the protocol (if available) to add asynchronous request-response capabilities in a common way.

Connection timeouts are a bane for service requests that require a long time to process. Asynchronous communication can solve the issue.

The Testbed-12 Implementing Asynchronous Services Response ER documents two approaches investigated in Testbed-12 for handling asynchronous interactions with OGC Web services:

- An OGC Web Processing Service (WPS) serves as a facade to another OGC service (e.g. WFS and WCS). The WPS manages the request to the OGC service and publishes the result so that the client can get it (a polling approach).
- A WFS has been extended so that the client can provide callback information with a request (KVP and POX), to which the service will send the actual response once it is available. In the KVP binding, the callback is provided using a parameter. In the POX binding, the callback is provided using a new element in the request (and thus requires a revision of the WFS schema). The extension also supports an option to cancel a request that is executed asynchronously. Finally, instead of sending the response to the callback, the extension also supports a polling option.

Previous work also dealt with asynchronous communication with OGC services:

- Asynchronous communication has been described in the OWS 6 SWE Event Architecture ER, (chapter 8.2). That Engineering Report recommends the use of WS-Addressing to realize asynchronous request-response in a SOAP binding. It also suggests using it in a POX binding. It refers to an article from Tilkov on how asynchronous request-response can be realized in a RESTful binding.
- The OGC Sensor Planning Service (SPS) v1.0 included a notification target (pointing to an OGC Web Notification Service) in operation requests that require asynchronous execution semantics. The OGC SPS v2.0 removed the coupling with OGC WNS. Instead, it relies on standardized extensions of the communication protocol to achieve this. SPS 2.0 only defines a SOAP binding. It uses WS-Addressing to realize asynchronous request-response in that binding.

Conclusions:

- The HTTP protocol relies on requests and responses sent via a single client-server connection.

This is still the case in version 2 of the protocol. A timeout can always break the connection. A mechanism to realize asynchronous request-response should therefore not rely on being able to return the response on the same connection via which the request was sent.

  - In the case of using a WPS as facade to another OGC service: if both services do not use asynchronous request-response themselves, then the request-response message exchange between the two services can be interrupted by a timeout as well.

- WS-Addressing is a standard for realizing asynchronous request-response in a SOAP binding. No standard exists yet which defines an extension for HTTP that fully realizes asynchronous request-response. However, implementations exist that use HTTP extension points (like HTTP headers, see this example) to convey information with which asynchronous request-response can be realized. The according approach can be represented - and implemented - as an additional layer between the HTTP and application layer.

  - The WFS extension developed in Testbed-12 requires a new parameter in KVP and XML encoded operation requests. When XML encoded requests are used in a SOAP binding, this creates an overlap with WS-Addressing (the standard solution for asynchronous request-response in a SOAP environment). Furthermore, it requires an extension of application layer messages with communication details that could be handled in the underlying communication layers.

- If a communication protocol natively supports (directly or through a standardized extension) asynchronous request-response, that mechanism should be used. An example is WS-Addressing for SOAP. Otherwise, if the communication protocol provides a suitable extension point, use it to add asynchronous request-response capabilities in a common way. Using HTTP headers could be a suitable solution for providing callback information and thus realizing asynchronous request-response in a common way for OGC services (in KVP, POX, and RESTful bindings).

7.3. Improving the Standardization Process

In addition to technical measures, the standardization process may also offer opportunities for improvement related to the aspects discussed in this Engineering Report.

7.3.1. Updating Service Profiles

Recommendation: When a new revision of an OGC web service standard is developed, require that updating service profiles has been considered.

The Testbed-12 Catalogue and SPARQL ER recommends that the release of CSW 3.0 means that profiles of CSW 2.0.2 should be adapted to support CSW 3.0. This recommendation can be generalized: If a service profile is still relevant for a new revision of an OGC web service standard then the profile should be updated, too, in a timely manner.

The SWG that develops a revision of an OGC standard should review the standardized OGC profiles for the current version and propose further changes to the OGC Standards Baseline. This can be as simple as listing known profiles that should be updated in the release notes or as additional input for the OAB review and RFC.

The OGC standards development process could include a check - for example in the OAB review -

that known profiles have been considered by the SWG.

7.3.2. Deprecation of OGC Standards

Many of the Testbed-12 web services support earlier versions of the respective OGC web service implementation standard.

We take a closer look at the services listed in the WCS table and the WFS table. Some WCS and WFS services do not support the latest version of the according OGC standard, which is 2.0 for WCS and 2.0 for WFS. In fact, there are WCSs that only support WCS version 1.0 or 1.1, and some WFSs that only support WFS version 1.1.

The big difference between the WCS and WFS standards is that all except the most recent version of WCS are deprecated (source: http://www.opengeospatial.org/standards/wcs), while none of the various versions of WFS are deprecated (source: http://www.opengeospatial.org/standards/wfs).

According to the OGC TC policies and procedures, a deprecated document is: *"An official standard of the OGC but no longer maintained. An OGC document shall be deprecated by a vote of the OGC Voting Members, usually as part of a standards adoption vote."*

The analysis showed that at least within this testbed, (WCS) services that only support a deprecated version of the respective OGC standard exist and are in use. This raises the question why that standard has been and whether it should be deprecated, which means that it is no longer maintained. If an error in the specification is identified, then for a deprecated OGC standard it seems like there is no room to correct the issue (which would result in a new version of that standard with an increased bugfix version identifier, for example going from 1.0.0 to 1.0.1).

What are the implications for interoperability? If the bug is about an interoperability issue, more specifically a way to implement a specific aspect of an OGC service differently, even though such diversity was not the intent, then by not fixing this bug the level of interoperability for that standard will decrease.

Which implications can the deprecation of OGC standards have for procurement decisions? Again, think about a case where a bug is detected in the standard that is implemented by software you bought. If this standard is deprecated by the OGC after you bought the software, then the bug will not be fixed in the version of the standard that your software implements. One could argue that such bugs could be solved in a new version of the standard with an increased major or minor version number (e.g. going from 1.0.0 to 1.1.0 or to 2.0.0). Upgrading the service implementation, however, would result in according costs - and that does not take into account the components (especially clients but also other services) that may depend on that service.

The OGC TC policies and procedures include the definition for a retired document, which is: *"An OGC document that, by Member approval, is no longer an official or supported document of the OGC. As such, retired documents should not be referenced in any procurement, policy statement, or other OGC document. Retired documents are made available on the OGC website for historical purposes."*

For retired documents the OGC recommends to not reference them in any procurements. Apparently this is not the case for deprecated documents, even though deprecated documents are no longer maintained by OGC.

This raises the question, if the OGC should even be concerned with deprecating and retiring OGC standards. Instead, it could let software implementers and users decide which standards suit their needs. Furthermore, resolving deficiencies and bugs in OGC standards should always be possible. Change requests (that include content to solve the issue) are a suitable way to achieve this.

Conclusion: Review the model for deprecating and retiring OGC standards?

7.4. Specific Cases

7.4.1. Catalogue Service Interoperability

The Testbed-12 LDS Thread investigated the interoperability between different profiles of CSW 2.0.2 and interoperability between different versions of CSW. The Testbed-12 Catalogue and SPARQL ER documents the results of this investigation, as well as a number of potential solutions. According to the ER, the interoperability in the aforementioned cases is limited. The LDS thread addressed this through implementation of transformers that transformed CSW responses into GeoDCAT and sending to a DCAT/SPARQL Service. This is described in the Testbed-12 Catalogue and SPARQL ER.

NOTE	The Testbed-12 Catalogue and SPARQL ER defines "interoperability" as the *"capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units"*.

Observations:

- Annex B of the Testbed-12 Catalogue and SPARQL ER shows that DCAT does not support a 1:1 mapping for the CSW profiles.
- GeoDCAT (GeoDCAT-AP, as published by the European Commission) supports a mapping of INSPIRE metadata to DCAT. The reverse mapping is not supported. Furthermore, the INSPIRE metadata profile is based on the core profile of ISO 19115 (see table 3 in ISO 19115). The INSPIRE metadata profile therefore does not cover the full range of metadata information defined by ISO 19115. Consequently, GeoDCAT does not support a mapping from all ISO 19115 profiles to DCAT.

Conclusions:

- Systems that implement OGC standards are not necessarily fully interoperable out-of-the-box, since OGC standards typically offer many options to choose from. A profile reduces the number of these options. It is built to serve a specific purpose (for example to identify the metadata items that are required within a certain community). Systems that implement the same service and information model profiles can be expected to have a high degree of interoperability. Interoperability is limited between systems that implement different service and information model profiles.
- DCAT supports a subset of the information items that are required by the metadata profiles investigated in Testbed-12. DCAT and GeoDCAT do not fully support a mapping to and from any profile of ISO 19115. This has to be understood when considering standardizing GeoDCAT in OGC.

- The Testbed-12 Catalogue and SPARQL ER recommends a change to the CSW standard to add a SPARQL endpoint in order to enable catalogue services to serve metadata in DCAT format in addition to the current CSW 3.0 bindings so that catalogues return the metadata in a common language (even though this language may not be able to represent all information in the source data). Such a step would impact many catalogue providers and it will be interesting to have a discussion of the costs and benefits of such a move.

Annex A: Revision History

Table 2. Revision History

Date	Release	Editor	Primary clauses modified	Descriptions
June 22, 2016	0.1	Clemens Portele	1 (Introduction)	initial version
September 30, 2016	0.2	Johannes Echterhoff	throughout	include initial results of web service metadata analysis
October 21, 2016	0.3	Johannes Echterhoff	throughout	update of web service metadata analysis, draft of initial recommendations
November 8, 2016	0.4	Johannes Echterhoff, Clemens Portele	throughout	update based on feedback from Testbed-12 participants, analysis of (S)DW best practice, additional conclusions and recommendations, revision of standard ER template sub-clauses
November 15, 2016	0.5	Johannes Echterhoff, Clemens Portele	throughout	update based on feedback from Testbed-12 participants and IP team, finalizing the ER