OGC Testbed-14

WMS QoSE Engineering Report
# Table of Contents

1. Summary ................................................................................................................. 4  
   1.1. Rationale ............................................................................................................. 4  
   1.2. Context ............................................................................................................... 4  
   1.3. Requirements & Research Motivation ................................................................. 4  
   1.4. What does this ER mean for the QoSE DWG and OGC in general .................... 7  
   1.4.1. QoE specific problem statements from the DWG Charter .......................... 7  
   1.5. Prior-After Comparison ...................................................................................... 7  
   1.6. Recommendations for Future Work .................................................................. 8  
   1.6.1. Recommended Future Tasks and Deliverables .............................................. 8  
   1.7. Document contributor contact points ................................................................. 10  
   1.8. Foreword ............................................................................................................ 11  
2. References ............................................................................................................... 12  
3. Terms and definitions .............................................................................................. 13  
   3.1. Abbreviated terms ............................................................................................ 13  
4. Overview .................................................................................................................. 14  
5. Components and Component Scenario ................................................................. 15  
   5.1. Component Overview Chart .............................................................................. 15  
   5.2. Quality of Service & Experience (QoSE) .......................................................... 15  
   5.3. Client with QoSE Support .................................................................................. 16  
   5.3.1. WMS Metadata Elements ........................................................................... 17  
   5.3.2. Operations .................................................................................................... 18  
   5.4. Test Suite for WMS Service Quality Assessment ............................................... 26  
   5.4.1. Methods ....................................................................................................... 27  
   5.4.2. Results ......................................................................................................... 28  
   5.5. WMS Stress Testing with JMeter ....................................................................... 31  
   5.5.1. On stress testing and benchmarking ............................................................... 31  
   5.5.2. JMeter introduction ....................................................................................... 32  
   5.5.3. Simulating concurrent users ......................................................................... 33  
   5.5.4. Separation from the tested infrastructure ..................................................... 34  
   5.5.5. Impact of Network and other tools .............................................................. 34  
   5.5.6. Synthetic requests versus real-world requests .............................................. 34  
   5.5.7. Randomization of requests parameters ....................................................... 34  
   5.5.8. Putting together the results .......................................................................... 36  
   5.5.9. JMeter case study example ......................................................................... 36  
   5.5.10. Integration with other QoS tools .................................................................. 39  
   5.6. GeoServer Extension for QoS .......................................................................... 39  
   5.6.1. GeoSolutions Methodological Approach on Monitoring QoS ..................... 43  
   5.7. OGC Web Service Landing Pages ..................................................................... 45
Publication Date: 2019-02-15

Approval Date: 2018-12-13

Submission Date: 2018-11-28

Reference number of this document: 18-028r2

Reference URL for this document: http://www.opengis.net/doc/PER/t14-D011

Category: Public Engineering Report

Editor: Guy Schumann (RSS)

Title: WMS Quality of Service & Experience

OGC Engineering Report

COPYRIGHT

Copyright (c) 2019 Open Geospatial Consortium. To obtain additional rights of use, visit http://www.opengeospatial.org/

WARNING

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.
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.
Chapter 1. Summary

1.1. Rationale

Quality of Service (QoS) and Quality of Experience (QoE) as they are intended and described at the OGC are two related concepts which require very specific treatment and characterization. Citing the definitions provided by the Domain Working Group (DWG) charter document:

- **Quality of Service**: Technical reliability and performance of a network service. Typically measured using metrics like error rates, throughput, availability and delay or request response time. This Engineering Report (ER) attempts to handle QoS aspects such as service availability, scalability and speed.

- **Quality of (User) Experience**: A holistic, qualitative measure of the customers’ experience of the application or service. It encompasses both the user experience and the customer support experience of the evaluated applications and/or services.

QoE focuses on the usability of the information that is conceived via OGC services to end users or other client application and therefore is concerned more with qualitative aspects of such services like presence of metadata, proper and descriptive namings, appropriate styling and so on (a more thorough treatment is present in the QoE discussion paper OGC 17-049 entitled "Ensuring Quality of User Experience with OGC Web Mapping Services" available at [https://portal.opengeospatial.org/files/?artifact_id=74403&version=1](https://portal.opengeospatial.org/files/?artifact_id=74403&version=1)).

QoS focuses on providing reliable (i.e. quantitative) measures of spatial data service metrics which can be used to characterize how a service (one or more specific datasets exposed by a certain service) is performing both in near real-time as well as historically. It touches concepts like availability, scalability (also known as capacity), absolute performance (i.e. speed) and can be used to assess also perceived performance by final clients. As mentioned above, it is typically measured using metrics like error rates, throughput, availability and delay or request response time.

Quite often the QoS and QoE aspects of spatial data services are underestimated if not simply ignored due to lack of resources as well as lack of awareness, resulting in services which are difficult to exploit (i.e. QoE very low) and/or unstable or very slow (i.e. QoS very low). The result is that few users end up using them after the initial launch and this is especially true for services targeting end users who are used to interact with services a-la Google Maps which delivers extreme performance and scalability as well as bullet-proof usability.

1.2. Context

The ability to combine and visualize location-based data using a Web Map Service (WMS) is a key value proposition of the Federal Geospatial Platform (FGP). The FGP is a collaborative online environment where a collection of the government of Canada’s most relevant geospatial information can be found easily and viewed on maps to support evidence-based decision-making, foster innovation, and provide better service for Canadians. The FGP includes the capability of selecting datasets with WMS to view individually or combined with other services in the FGP viewer.
This functionality, as is the general case with a WMS, is provided to allow users to immediately visualize and analyze geospatial data. Unfortunately, user feedback has proven that these geospatial web services are not always easy or intuitive to navigate, combine or understand. Because the FGP’s primary end users are government policy analysts who are not always highly familiar with mapping and web mapping technologies, it is important to make WMS, and the content they make available, as user-friendly as possible.

In 2016, to help alleviate this issue, the FGP developed a web service quality assessment methodology that supported WMS, ran an assessment and developed recommendations and best practices to support a user-friendly experience for all employees and citizens using WMS. Assessments to date have shown that key considerations are often very simple, but very impactful on QoE. The results of this study were used as the primary input into an OGC Discussion Paper created by the Quality of Service Experience (QoSE) DWG. The OGC QoSE-DWG has developed a discussion paper (OGC 17-049) entitled “Ensuring Quality of User Experience with OGC Web Mapping Services” [https://portal.opengeospatial.org/files/17-049] that identifies and describes issues often encountered by users of OGC WMS that affect the quality of their experience, and also provides an assessment framework for identifying issues and measuring quality, along with potential solutions and guidance to improve the usability of services.

The assessment framework for measuring QoE and the associated recommendations for improving service quality are intended to benefit human end-users who need to rapidly assimilate and use web mapping visualizations to answer questions or input into analysis. In other words, they need to be able to make sense of the information an OGC WMS provides them.

In addition, Testbed-13 addressed QoS aspects in the aviation domain. Though specific to a particular domain, The Testbed13: Data Quality Specification Engineering Report [http://docs.opengeospatial.org/per/17-018.html] addressed a number of general aspects that apply to this task nevertheless.

1.3. Requirements & Research Motivation

Testbed-14 has addressed the following WMS usability aspects:

1. Develop a revision of the OGC Discussion Paper 17-049: Develop a revision of the OGC Discussion Paper 17-049 as per the review and assessment of the current OGC Discussion Paper 17-049. This revision shall be performed in coordination with OGC QoSE DWG to address potential new requirements such as QoS metadata (Capabilities Extensions) server implementation and monitoring client implementations that allows improved service quality and measurement. The revision is part of this ER.

2. Quality of Service Experience Assessment Framework: The fourteen assessment criteria described in Discussion Paper 17-049 and summarized in Figure 1 are all aimed at assessing the quality of a web service in terms of the degree to which it conveys clearly understood information to the user. The user is assumed to be a non-expert in geospatial web services, but in most cases the criteria are equally valid for all classes of users. A selected number of services in the Testbed will be assessed against the QoSE assessment criteria, and retested once recommendations have been applied. Comparison of results will help to validate the effectiveness of the assessment criteria and the corresponding recommendations to improve usability, and allow for feedback, improvement or correction. Services are provided as Testbed-14 deliverable D115 and, a dedicated client as D116.
3. **Quality of Service Experience Practices to Alleviate Usability Issues**: The fourteen assessment criteria all have corresponding recommendations that describe practices that, once implemented, should help to alleviate user confusion and improve the usability of WMS as a means of visualizing and simple visual or query-based analysis of geospatial data. Once selected services have been assessed against assessment criteria, all or as many QoSE recommendations will be applied, then services will be subject to retesting.

4. **Quality of Service Experience Implementing Best Practices: Man or Machine?**: Analysis required here on the making of a geospatial WMS and the opportunities of human operator or programmatic response to make decisions that impact service quality of experience. Assessment of which usability issues are determined/caused by human input or what determined/caused by default via the programmatically generated aspects of the service (not due to direct human input or decisions) via the standard specification implementation. Results are captured in this ER.

5. **Quality of Service Experience Test Suite**: Develop test suite to programmatically test and validate that best practices have been successfully implemented. The test suite will automatically assess the quality of service according to the assessment criteria and validate or flag services that do not comply to best practices/recommendations. The results of the test suite (D117) are also captured in this ER.

Figure 2 illustrates the QoSE work items as reflected by the OGC Testbed-14 thread.
Note: The purpose of D122 is to provide a Client to meet requirements for QoSE and Portrayal as well as measure statistics on response time, number of layers, number of features, error codes generated, etc. within the client. Since subjective measurement criteria, focusing for example on legibility, intuitiveness, consistency of symbology, may be more difficult to automate, participant will gather QoSE statistics manually for those criteria or elements, and then propose recommendations for an automated service for those assessment criteria.

1.4. What does this ER mean for the QoSE DWG and OGC in general

The OGC QoSE DWG provides a forum for discussing issues related to QoS and QoE of spatial data services and applications relying on these services for delivering timely and accurate spatial information to the end-users. Key business goals of the Working Group include the following:

• Sharing implementation experience and ideas in evaluating and improving QoS and QoE of spatial data services.
• Collect best practice, create and promote guidance on evaluating and self-assessing the QoE of spatial data services, and practical means for improving the user experience of these services.
• Identify gaps in the existing standards and guidance related to QoS and QoE of spatial data services, and as appropriate, suggest new standardization activities within OGC to fill those gaps.

1.4.1. QoE specific problem statements from the DWG Charter

• Metadata, such as titles and keywords, is not always written in clear and understandable language considering the end users. Further, metadata does not always describe the provided services and data sets in necessary detail. This makes it more difficult for users to fully take advantage of the provided service and its datasets. There should be better guidance and a checklist to assist data and service providers to record human-readable metadata.
• The ancillary information, such as legends for WMS, is not always clear and human-readable. Missing or ambiguous legend information may easily lead to misinterpretation of the presented data. There should be better guidance for data providers on specifying good and readable legends.
• There is a lack of methods and best practice on evaluating and improving user experience and human interaction of processes involving discovery, initial evaluation of the fitness-for-purpose of spatial data services, as well as access to change records during the services’ lifetime.
• There is a need for more accurate understanding of how spatial data services are used and perceived in the larger web service end-user community.
• Evaluating, comparing and improving the QoE of spatial data services is difficult without commonly agreed and well-defined metrics for measuring the QoE.

1.5. Prior-After Comparison

The task described in this ER addressed the QoE assessment criteria laid out in the DWG Discussion
paper referred to earlier and also implemented a QoS performance assessment. Specifically, an interactive user-friendly Graphical User Interface (GUI) was developed based on the QoE assessment criteria defined by the DWG. This GUI also lists performance statistics from a WMS server.

1.6. Recommendations for Future Work

A goal of this task and its analysis was also to suggest potential future activity where these results could be investigated through new tasks in future testbeds.

1.6.1. Recommended Future Tasks and Deliverables

Item Related to QoSE Discussion Papers

As per OGC Policies and Procedures, the Discussion Papers are not versioned documents. The option discussed in the QoSE DWG has been that a new document (e.g. perhaps a Best Practice) would be produced based on the original Discussion Paper and the results of the Testbed-14.

Item Related to Future OGC Work

There should be a discussion on how QoS/QoE will fit into next generation OGC service interface specifications (i.e. WFS3, CAT4) given the clean break and move to REST/JSON/etc.

Quality of Experience (QoE) Items

- Extend QoSE evaluation work to test other geospatial services

Although current QoSE indicators are specifically designed for the Web Map Service, some of them could be directly used to measure the quality of other OGC web services (OWS). For example, the title or similar elements in nearly all OWS types could be considered, resulting in the title meaningfulness and related criteria being mapped to the elements. In addition, the feature-related QoSE indicators could be used to evaluate the quality of the WFS. And most of them could be directly used to measure the WMTS.

- Improve current QoSE indicators

The fifth QoSE criteria is named as Feature Attribution, but its description indicates that it is used to measure the number and relevance of attributes provided for each feature. Besides, the attribution is used to identify the source of the geospatial information. As a result, it might be worth considering to change the Feature Attribution to measures of Feature Attribute.

The fourteen (14) QoSE criteria only aim at assessing the quality of a web service in terms of the degree to which it conveys clearly understood information to the user. However, QoSE means identifying the usability issues associated with use of OGC web services. Hence, additional indicators beside the information understandability should be developed. For example, the service performance also affects user experience.

In the QoSE discussion paper, the user is assumed to be a non-expert. However, some indicators are difficult for them to understand. In my opinion, the evaluation on feature attribute needs some
expert knowledge, since it measures the number and relevance of attributes provided for each feature.

Not all items in the QoSE recommendations could find a matching item in the QoSE criteria, such as fees and access constraints, attribution, bounding box, and service interoperability. Thus, this needs improving.

Some future work related to D117

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Recommendation</th>
<th>Difficulty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Meaningfulness</td>
<td></td>
<td>The meaningfulness is too subjective for computers to measure it.</td>
</tr>
<tr>
<td>Consistency between Title and Map</td>
<td></td>
<td>It is hard to judge what the map content for computers.</td>
</tr>
<tr>
<td>Content</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature Attribute:</td>
<td></td>
<td>Feature Attributes:</td>
</tr>
<tr>
<td>The number and relevance of</td>
<td></td>
<td>a. The units used should relate to accuracy and common usage</td>
</tr>
<tr>
<td>attributes provided for each</td>
<td></td>
<td>b. Numerical precision should be given to correspond to and reflect actual accuracy.</td>
</tr>
<tr>
<td>feature</td>
<td></td>
<td>Common usage of units is too subjective.</td>
</tr>
<tr>
<td>Feature Attribute Values:</td>
<td></td>
<td>Actual numerical precision judgement needs some other description about the data provenance, which is usually hard to judge it based on the limited service description.</td>
</tr>
<tr>
<td>the understandability of</td>
<td></td>
<td>How to judge a value represents a code or real measure result is hard.</td>
</tr>
<tr>
<td>attribute values</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conveys meaning effectively</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not conveys meaning effectively</td>
<td></td>
<td></td>
</tr>
<tr>
<td>excessive precision, code given</td>
<td></td>
<td></td>
</tr>
<tr>
<td>but unclear as to what it means,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vague</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supporting Docs</td>
<td>Metadata, Series and Supporting</td>
<td>It is hard for computers to measure the understandability of one document.</td>
</tr>
<tr>
<td>Service Metadata</td>
<td>Documents</td>
<td></td>
</tr>
<tr>
<td>Legend</td>
<td></td>
<td>Legend legibility and understandability, and which numbers are codes are hard for computers to measure or judge.</td>
</tr>
<tr>
<td>Cartography</td>
<td></td>
<td>Color and symbols legibility measurement; visualization quality and understandability; Map content consistency measure across different scales with symbols size and overlap.</td>
</tr>
<tr>
<td>Bounding Boxes</td>
<td></td>
<td>Hard to get the accurate area of a given resource.</td>
</tr>
<tr>
<td>Service Interoperability:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a. In the case of data series,</td>
<td></td>
<td>How to define a data series.</td>
</tr>
<tr>
<td>one style should be visually</td>
<td></td>
<td></td>
</tr>
<tr>
<td>distinguishable from other data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>in the series, and another style</td>
<td></td>
<td></td>
</tr>
<tr>
<td>should be similar or identical to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>other data in the series.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Quality of Service (QoS) Items

- Investigate further towards the concept of a status page to report information about availability and status of OGC services endpoints;
- Investigate further, together with the QoSE DWG, the QoS extensions to GetCapabilities operations for WMS and WFS in order to streamline them and propose them as profiles for the respective OGC services;
- Better link the “E” (Experience) of QoSE with QoS since bad performance and/or availability means bad experience for the end-user; meaning that OGC should try and draw a link between checking the famous "14 rules" that are more towards naming and metadata, and that OGC should check for decent performance and decent stability when assigning scores to layers (ideally the score should become dynamic at this stage since performance and availability should be monitored over time);
- Advance performance testing of WMS services by executing the load tests through the containerized architectures: Load testing is crucial for WMS that have to serve many users in real time. Load testing involves simulating similar actions, where computer-generated virtual users mimic real users. In most cases, this kind of testing cannot be generated by a single server due to limited hardware resources. Additionally, physically distributed servers are a must if user behavior is desired from different regions of the world. For these large-scale load tests containerized cloud services offer a good alternative. Cloud services that make use of containerized architectures such as Kubernetes, rkt, etc, offer an alternative to standalone machines, running full virtual machine instances for applications and are well-suited for scaling of simulated clients;
- Extend performance tests by expanding to other geospatial services: There is a variety of open geospatial standards to provide online access to geospatial data, for example, Web Coverage Service (WCS), Web Map Service (WMS), Web Feature Service (WFS), and Web Map Tile Service (WMTS). OGC Testbed-14 focused on performance tests for WMS, but WCS and WFS services are more sensitive to data resolution. For WFS services, due to their verbose nature, transferring large amounts of data might be problematic and lead to higher latencies and lower performance. Extending performance tests to other geospatial services offering data from a sponsor would provide a broader view of how well other geospatial services are provided.

1.7. Document contributor contact points

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

Contacts

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guy Schumann</td>
<td>Remote Sensing Solutions Inc.</td>
</tr>
<tr>
<td>Albert Kettner</td>
<td>Consultant for Remote Sensing Solutions Inc./INSTAAR, DFO, CU Boulder</td>
</tr>
<tr>
<td>Simone Giannecchini</td>
<td>GeoSolutions SAS</td>
</tr>
</tbody>
</table>
1.8. 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 normative documents are referenced in this document.


- OGC: OGC 09-025r2, OGC® Web Feature Service 2.0 Interface Standard – With Corrigendum, July 2014

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

- Quality
  
  degree to which a set of inherent characteristics fulfills requirements [SOURCE: ISO 9000:2005, 3.1.1, modified - Original Notes have been removed.]

- Usability
  
  The degree to which something is able or fit to be used

3.1. Abbreviated terms

NOTE: The abbreviated terms clause gives a list of the abbreviated terms and the symbols necessary for understanding this document.

- DWG Domain Working Group
- QoE Quality of Experience
- QoS Quality of Service
- QoSE Quality of Service and Experience
- WMS Web Map Service
Chapter 4. Overview

This ER describes all the different components that were implemented to advance WMS QoSE. Section 5 describes the following components in detail:

- GUI for WMS Service Quality Assessment
- Test Suite for WMS Service Quality Assessment
- WMS Stress Testing with JMeter
- GeoServer Extension for QoS
- OGC Web Service Landing Pages
- TIE and Scenario for Demonstration
Chapter 5. Components and Component Scenario

5.1. Component Overview Chart

Figure 4 illustrates the connections and interoperability between the various component parts discussed in this ER.

5.2. Quality of Service & Experience (QoSE)

A discussion paper entitled "Ensuring Quality of User Experience with OGC Web Mapping Services" provides an assessment framework for identifying issues and measuring quality, which can be downloaded at public link [https://portal.opengeospatial.org/files/?artifact_id=74403&version=1]. It outlines fourteen criteria to measure QoSE, and recommendations are given for each of them, which are shown as components using a rectangle in Figure 5.
5.3. Client with QoSE Support

D116 requires implementation of a client dedicated to testing the assessment criteria of WMS instances. Therefore, a Graphical User Interface (GUI) was designed and is shown in Figure 6. It consists of three basic components, including a WMS layer selector, metadata information viewer, and QoSE assessment panel. Users can select a layer of interest from the WMS layer selector. Once a layer is selected, its metadata is loaded and shown in the metadata information viewer, which is located in the bottom-left. Finally, users can give their evaluation for each QoSE indicator based on the layer metadata contents. At the same time, users can also leave feedback about the QoSE framework or the client using the feedback panel.
5.3.1. WMS Metadata Elements

In Figure 5, this ER also explains and works out the necessary metadata elements for the QoSE assessment. According to the WMS specifications, all metadata elements could be extracted as shown in Figure 7.
Figure 7. WMS Metadata Elements Extraction

Metadata cache, through pre-parsing the metadata elements and storing them, could enhance the user experience by reducing user wait time. Therefore, some metadata elements contained in the capability documents are pre-parsed and stored in the local database. However, some metadata elements are not suitable for this procedure. For example, the layer map may be configured with different rendering strategies at different zoom levels, but it is difficult to know how many rules exist. Also, the WMS only supports pixel-wise GetFeatureInfo operation, which makes it hard to cache all feature information, and it is also a lot of effort to manage the vector information. For this, the live request operation is provided which will be described in the sections below.

5.3.2. Operations

5.3.2.1. Layer Selection

In the WMS specification, the title element is described as more understandable for humans than the layer name which is designed for computers to identify. Therefore, the layer title is adopted to assist users in layer selection. But layers from different WMS services may share the same title. As a result, our portal requires users to specify a WMS before the layer selection.
5.3.2.2. Map Window

The map window consists of a background map from OpenStreetMap (OSM), the layer map, and the layer legend. It supports some basic map operations including roam, zoom in/out. In addition, the map click operation is valid for queryable layers with a crosshair cursor shape. If the layer is not queryable, the click operation will be hidden, and the mouse cursor on the map will be changed to a move shape. If it is too small for users to view or operate, a full screen operation is implemented like that in Figure 11.
5.3.2.3. Feature Attributes

As described in the above sections, the feature information is loaded in real time instead of being cached together with other layer metadata elements. Therefore, the feature information value is left empty when users select a WMS layer (Figure 12). The feature information value is filled, once users click a feature pixel in the map window, provided the layer is queryable. In this process, a GetFeatureInfo request is constructed in real time, then the response of the request is shown as the feature information value. If no feature information is returned or the user clicks on an empty pixel, the feature information value is shown like that in Figure 13. At the same time, a purple cross symbol will appear on the map, whose center means the location of the selected pixel (Figure 13, Figure 14).
5.3.2.4. Criteria Assessment

There are two types of QoSE indicators, criteria and recommendations. For the criteria, the items are mutually exclusive. Therefore, a radio button group is adopted in our client (Figure 15). For QoSE recommendations, users can suggest multiple items belonging to the same recommendation (Figure 15). Once users give their assessments, the corresponding statistical graphs of past results will change immediately.
There are some friendly prompt functions implemented to assist users’ operations. For example, users can view the detailed description of each QoSE indicator and their items by moving the mouse on the (?) symbols (Figure 18, Figure 19). If the operation is layer dependent, users should first select a layer of interest, otherwise a prompt like Figure 20 will popup.
If users or service providers want to get a complete QoSE report for a layer or a service, they can send an email to us using the address listed in the bottom in the Figure 21.
5.3.2.6. User Management

A user profile is sometimes important for analyzing the QoSE results. For example, an expert in Geographic Information Systems (GIS) is usually more professional than a common user for a QoSE assessment. Therefore, a user management module is developed, and the profile information list in Figure 22 will be collected. In addition, the client also allows users to share their assessment as a guest. Users can check their logged status in the location shown in Figure 24.
Figure 22. Sign Up

Create a new account

Account

Password

Confirm password

Full name

Institution

Discipline

Degree

E-mail

Confirm e-mail

Create a account

Already registered? Sign In

Figure 23. Sign In
5.3.2.7. Feedback Collector

If users want to share their opinions on the QoSE indicators or our client, the feedback menu tab allows them to describe and submit them (Figure 25). Some friendly prompt functions are also developed like those in Figure 27.

Figure 25. Feedback

Figure 26. No Feedback Content Prompt

Figure 27. Successful Feedback Prompt

5.4. Test Suite for WMS Service Quality Assessment

The test suite is implemented to programmatically test and validate whether best practices have been successfully implemented, and it can automatically assess part of the quality of service.
according to the assessment criteria and validate or flag services that do not comply to best practices/recommendations.

5.4.1. Methods

There are tens of assessment criteria and recommendations for assessing the quality of service. Some of them are easy to programmatically test, but some others are too difficult to automatically test, even for people. This section describes which QoSE indicators have been implemented in the current test suite and briefly describes how to test them programmatically. The other QoSE indicators remain unsolved and reasons are summarized in the future research section.

5.4.1.1. Title

The WMS specifications allow WMS providers to edit layer titles as free text. This section removes the special symbols first, and then conducts word segmentation to split the title into individual words.

5.4.1.1.1. Title Length and Uniqueness

The title length could be easily calculated by counting words. Title uniqueness requires the title to start with the most unique or important aspect of the data first. Because different WMS providers organized their layers based on different indicators, the layer title uniqueness calculation on service level is more reasonable than that across all WMS. In addition, it would be a 'big task' to consider all WMS layer titles as more WMS come. To calculate the title uniqueness, the number of each word occurrence is counted among all layer titles in the same WMS. Then a new title could be generated by changing word sequences based on their occurrence time for each title. Finally, the uniqueness is calculated by comparing the new title and the old one.

5.4.1.1.2. Title Readability & Meaningfulness

Title readability plays a key role in measuring the title meaningfulness, and a less readable title is hard to be meaningful. In our test suite, the title readability is measured by the ratio of words in the title, which could be found in the English and French dictionary. Then two thresholds on the ratio are given to split the title as not meaningful, less meaningful and readable. To check whether the title is meaningful, human involved assessment is required.

5.4.1.2. Feature

In WMS specifications, only queryable layers support the GetFeatureInfo operation. Therefore, the queryable attribute value could be used to determine whether corresponding QoSE indicators are applicable. It is hard to download all feature information, because the GetFeatureInfo operation only allows users to request the feature information on one pixel at one time. Here, 100 feature pixels are randomly selected to get their feature attributes in each queryable layer map. Then the duplicated records will be removed. Finally, the feature information table will be used to evaluate the corresponding QoSE indicators.

5.4.1.3. Metadata, Supporting Docs and Abstract

In our current test suite, it can test whether the metadata and supporting docs are available. As for their understandability, it still needs human involvement.
5.4.1.4. Legend

Legend Necessity: An InceptionResnetV2 model is trained to check the legend necessity using manually labeled maps.

5.4.1.5. Map

Map Scaling Consistency: It could be tested by checking whether the minimum and maximum scale denominator exist.

5.4.1.6. Other

5.4.1.6.1. Fees, Access Constraints and Attribution

The recommendations related to the above metadata elements could be tested by comparing the elements' values with corresponding descriptions in recommendations.

5.4.1.6.2. Bounding Boxes

To test whether the spatial coverage of a layer map is global, a VGG19 model was trained using some manually labeled maps. As for maps with non-global spatial coverage, it still needs humans to determine.

5.4.2. Results

Because the test suite adopts the same QoSE indicators set as that in the client, the human evaluation results from the client could be used to assess that from the test suite. Therefore, a test suite results presentation module has been implemented, which is located below the statistical graph (Figure). Users can open or close it by checking or unchecking the checkbox named D117 Result in the top-right.
In group discussion, some QoSE indicators are too subjective or too hard for computers to programmaticaly evaluate them. For example, the feature attribute criteria measure the number and relevance of attributes provided for each feature. To assess it, an expert knowledge database is in demand which should contain attributes' requirements for each feature type in different
applications. In the test suite result presentation module, descriptions of difficulty for programmatically assessing them are presented instead of test suite results.

Figure 30. An example of Test Suite Results on Feature Related Criteria

Figure 31. An Example of Test Suite Results on Feature Related Recommendations

Note that test results of the JMeter (see next section) component have been integrated into the portal, under the tab “Performance Load Reports”. GeoSolutions’ QoS monitor, outlined later, is also integrated into the portal, under the tab “QoS Monitor”.

30
5.5. WMS Stress Testing with JMeter

Note that the decision to use JMeter as the tool for running the stress tests was not a requirement by the sponsors, rather it was based on a QoSE Testbed-14 team discussion. Other software alternatives, such as GeoHealthCheck (mentioned later in this report) and Spatineo Monitor, could of course be used to accomplish the same or similar results. Apart from stress testing, these tools can be used for monitoring the OGC Web Service availability and its changes over longer periods; however, here integration with (these) other tools was not considered.

5.5.1. On stress testing and benchmarking

Stress testing is crucial to investigating the behavior of infrastructure or an application under heavy load. It usually involves measuring the following:

- Scalability, as in the ability to keep performing under increasing loads
- Speed, as in the ability to respond with acceptable and stable response time to requests

Note: There needs to be a discussion or agreement on stress testing against operational services with the server and service provider in order to avoid formal blocking of requests by the provider.

From an implementation/deployment standpoint the goal of stress testing is to gather concrete numbers about the characteristics that can be used over time to check the effect of evolutions. It can also be used to dimension hardware infrastructure and redundant deployments as well as to impose QoS restrictions to protect production deployment (e.g. limit number of parallel requests to a value that is closer but lower than the maximum throughput).

From a developer standpoint stress testing is also important in order to investigate software and architectural bottlenecks. As a developer trying to assess scalability, performance and availability of an infrastructure the goal is to hit the latter with a load that can force it to its limits (i.e. maximum resource utilization) and then analyze the effect on the underlying resources, find the bottleneck, fix it and then move to the next one. As an instance, bottlenecks can be identified in a WMS implementation (e.g. unnecessary synchronizations or inefficient code of sort) by not being able to fully exploit Central Processing Unit (CPU) and Memory under huge load. By experience, if at increasing load the throughput falls and the response time significantly increases but CPUs and/or memory are not fully utilized this is a clear indication that unless the implementation is being affected by slow Input/Output (slow disk, slow DBMS), then the implementation has some bottlenecks that need fixing.

Given what is written above, stress testing and benchmarking is an activity that should encompass the entire lifecycle of an infrastructure; it should be done during development. To do a sanity check on written code and chosen components, stress testing should also be done at deployment time to ensure proper dimensioning, it should also be done periodically when in production to double check and reconcile assumptions and expectation with reality. All this falls under the umbrella of Application Performance Management (https://en.wikipedia.org/wiki/Application_performance_management).

A useful case for stress testing is whenever server software, data sources or their configurations are changed: stress testing before and after modifying the service would verify if everything is still working as expected, or if the made changes had a real effect on service performance.
5.5.2. JMeter introduction

Apache JMeter (http://jmeter.apache.org/) is a popular open source tool used to load test (or stress test) web applications. JMeter lets us send an arbitrary amount of custom tailored OGC WMS requests to server applications to derive information about response time, throughput and availability of the services against heavy load conditions. Some of JMeter points of strength are mentioned below:

- Open source and free to use software
- It ships with an easy-to-use GUI;
- It allows to set up multiple thread groups, different parallelism and request count, to ramp up the load;
- It can use Comma-Separated Value (CSV) files to generate semi-randomized requests (needed to bypass server cache);
- It can execute parameterized tests;
- It reports results in simple tables and text files;
- It uses assertions for checking test results;
- It can be executed in headless mode (i.e. without the GUI) as part of a larger script.
- It is supported by a large active community. Multiple open source plugins have been developed by its community that make the tool applicable for tests to many different websites.

Within JMeter, a web performance test can be built, executed and analyzed.

1. Building web performance test plan: Building a test plan in JMeter for a WMS is fairly straightforward. First, define a set of variables that describe the specifics of the WMS and the global parameters for the entire run (in the case of Testbed-14 the path to the tiled or untiled variable domains, the throughput and the number of loops each ‘user’ should query the WMS. Depending on the type of test, specify a set of ‘threads’ or concurrent users to simulate parallel querying of the WMS. And finally define a number of reports and graphs to capture the results of the test.

2. Executing your test: Once a test plan has been built, there are various ways to run the test. This can be done either through the GUI or through the command line. The most preferable way is using the command line but due to limited time Testbed-14 chose the GUI to run the performance tests. The test environment can influence the performance test results (see below). Setup the JMeter tests outside the WMS infrastructure to avoid the test influencing the behavior of the WMS and if possible, a cloud service should be used simply as the load cannot be generated by a single server to simulate a large number of real users due to limited hardware resources of a single computer. Testbed-14 chose to use an Amazon Cloud service made available by GeoSolutions to run the JMeter load tests.

3. Analyzing test results: Load test results can be saved as CSV files. Testbed-14 chose to analyze how the ‘transaction throughput’ would behave with increasing the number of threads, and the same for response time. For this test, the result reports created include the raw data as well as graphs of: a) transaction throughput versus threads and, b) Response time versus threads. These are created both for WMS tiled and untiled services of the same WMS.
Figure 32 shows an example of a JMeter load test plan, showing the global variables, a set of threads (1-100) and some result output options (summary reports and graphs).

For a more thorough discussion on JMeter and its capabilities, the reader is referred back to the official JMeter documentation as well as to the GeoSolutions training which contains a section on how to stress test GeoServer using JMeter (https://geoserver.geo-solutions.it/edu/en/enterprise/jmeter.html).

As mentioned above, performance tests have the objective of probing the swiftness and behavior of a service under various load levels. Performance tests also reconstruct the throughput and average response time curves under increasing concurrent load to simulate different real-world conditions. In order to do so, JMeter test plans send an increasingly higher number of concurrent WMS requests to test services performance under various load conditions.

Below the most important points to consider when setting up a stress test plan; this ER will then discuss how one of these test plans is set up.

5.5.3. Simulating concurrent users

Generally speaking, software developers tend to associate a single user to a single thread in JMeter to send requests. When it comes to WMS this is partially true since, if an end-user is interacting with a WMS service via a web-based applications that uses map-tiling, a single user can send up to 6 requests in parallel (depending on the web browser of choice) and even more if tricks like setting up multiple names for the same cluster under tests have been set up.

So the equivalent between threads in a JMeter ThreadGroup and end-users needs to be evaluated on a case by case basis. However, when in doubt it is safe to assume that JMeter can simulate a single user interaction with a single thread.
5.5.4. Separation from the tested infrastructure

JMeter uses multithreading to simulate concurrent users, by having multiple threads sending requests in parallel and therefore when trying to simulate a huge load it can become quite heavy on the underlying infrastructure. Rule of thumb is therefore to avoid adding extra load to the infrastructure that is being stress tested by having JMeter run on a different one.

It is important to remark on the fact that a number of online services that can be used to generate a huge load based on a JMeter plan. In the past GeoSolutions have used https://flood.io/ for testing a large infrastructure, but there are other options like https://www.blazemeter.com/. In addition, JMeter can be used to run distribute tests by exploiting multiple slave machines from a single master as explained in the JMeter documentation (https://jmeter.apache.org/usermanual/jmeter_distributed_testing_step_by_step.pdf).

5.5.5. Impact of Network and other tools

When planning and executing a stress test with JMeter, carefully evaluate the impact of all the tools that sit between JMeter and the infrastructure that is being tested. This includes:

- Network capacity and eventual QoS configurations (firewall and so on). It is necessary to make sure that the network bandwidth and network devices in between JMeter and the infrastructure to test do not become bottlenecks and then generate results that are worse than could be achieved. The typical examples are poor outbound bandwidth on the server-side or anti Distributed Denial-of-Service (DDoS) protections at the firewall level which could impede huge parallel loads on the infrastructure when testing.

- It is necessary to make sure that, while running the tests, no other component apart from those that are being tested is generating load on the underlying hardware infrastructure. Failure to do so could lead to false results. The opposite is also true, it is necessary to make sure that the tests will not impact other critical parts of the overall infrastructure.

5.5.6. Synthetic requests versus real-world requests

It is obviously crucial that the requests used to generate the stress test load are as representative as possible of real-world usage patterns. This is possible for systems that have already been launched and made accessible to end-users (e.g. by collecting and curating access logs) but it is not always possible for pre-flight stress tests; moreover it is always important to somehow randomize requests (see also below) in order to discover bad behaviors that are not always obvious during normal usage.

In general, so-called synthetic requests tend to be created starting from valid WMS requests and then randomizing in a controlled manner some of the parameters like BBOX, width, height and so on. Often, mixed stress tests tend to be used where some of the requests are real-world requests and others are synthetic requests.

5.5.7. Randomization of requests parameters

Some of the parameters of each request (width, height, bounding box) are varied continuously to avoid aggressive caching on the server-side, this is done using JMeter “CSV Data Set Config” element
that allows us to fetch the above-mentioned parameters from pre-generated Comma-Separated Value (CSV) files containing pseudo-random values for each of the parameters as described below.

A way to perform this randomization is given by using a feature within JMeter that allows use of an external CSV file to override certain default parameters of a WMS request using the values taken from the CSV file itself. An example is shown below (Figure 33). Web Map Services can be set up as tiled and untiled services. For each an external CSV with changing boundary boxes has to be specified.

![Figure 33. CSV Data Set Config element within JMeter](image)

The config in the previous figure is responsible for fetching parameters from CSV files (file system path set in the TILED_CSV_FILE variable). An example of an external CSV file is given in Figure 34.

| 1441;643,-138.06979 | 48.683384 | -126.54939 | 53.823995;EPSG:4326 |
| 1119;757,-144.73178 | 48.596305 | -135.70632 | 54.701995;EPSG:4326 |
| 629;726,-132.53598 | 47.054275 | -127.90511 | 52.399285;EPSG:4326 |
| 1040;780,-145.30502 | 45.893408 | -132.51715 | 55.484311;EPSG:4326 |
| 1865;881,-144.50615 | 47.004977 | -126.20842 | 55.648566;EPSG:4326 |
| 1156;1161,-135.34703 | 45.550824 | -125.78543 | 55.153779;EPSG:4326 |
| 762;543,-142.80009 | 49.454692 | -137.28598 | 53.289787;EPSG:4326 |
| 1869;763,-144.37061 | 46.92431 | -127.00252 | 54.014655;EPSG:4326 |
| 1930;729,-139.294 | 49.888074 | -124.24607 | 55.571984;EPSG:4326 |
| 834;1021,-142.45204 | 45.916199 | -136.34296 | 53.395064;EPSG:4326 |

![Figure 34. Example of an external CSV file](image)

Fields are separated by a semicolon, the first two represent the values for width and height, next is the coordinates for the bounding box followed by the reference system for these coordinates.
A number of scripts exist already for the generation of the randomized parameters. GeoSolutions has put together Python scripts that can be used to generate parameters for WMS requests (and also for other services). These are available in a public GitHub repository (https://github.com/geosolutions-it/scripts/tree/master/performance_tests/WMS).

5.5.8. Putting together the results

JMeter can produce the results in Hypertext Markup Language (HTML) outputs but this output is somewhat cumbersome and requires some plugins and an installation hosted on a web server. GeoSolutions typically uses spreadsheets to record the output of the JMeter plans (shown below in Figure 35) using the number of threads as the independent variable and the throughput and response time as the dependent variable. Below are some examples charts for throughput and response time.

![Summary report from a JMeter run](image)

![Charting the throughput over increasing threads](image)

Figure 35. Output of the JMeter

5.5.9. JMeter case study example

To demonstrate the use of JMeter for this study we choose to do a load performance test on the WMS that is made available through: https://open.canada.ca/data/en/dataset/be0a3350-f755-418e-b04b-7ff9fd2ebeac. And choose a layer named “0”. This layer was selected using QGIS and represents geospatial data of “Pelagic Seabird Atlas, West Coast of Canada - Average Grid Cell Density, 2009” and has the following boundary box: -147.0,45.1666669999999968 : -123.0833329999999961,56.0 (see Figure 36).
This maximum boundary box was used to generate tiled and untiled variable boundary boxes with changing height and with running the script provided by GeoSolutions to a CSV file, similar as is shown in Figure 34. Once the tiled and untiled CSV files are in place the load performance test plan is built in JMeter (similar as shown in Figure 32). All in-place test runs should be conducted to make sure that JMeter receives data about the WMS when querying for tiled and untiled services. Make sure to log successful connections. These test runs can be done on a desktop in the JMeter GUI environment, and if all works well should show a snapshot of the geospatial data each time JMeter sends out a data request, see Figure 37.
Potentially, parameters need to be tweaked to be able to receive spatial data. TestBed-14 participants found that they could not create the variable boundary boxes captured by the CSV files for all WMS instances provided by Natural Resources Canada. Especially for datasets that cover only a very small spatial area these variable boundary boxes could not be generated and therefore the WMS site could not be tested. Also, for some WMS the tiled option was not activated and therefore this WMS capability could not be tested either.

Successful load performance tests are then moved and executed in the cloud. Running in the cloud, a summary report informs (see Figure 38) the user of the state of the execution and a graphical report (see Figure 39) provides a first indication of the performance of the WMS instance over time.

Figure 38. JMeter Summary report of a WMS instance, showing the number of loops for each thread it has processed already

Figure 39. JMeter graphical representation of the performance of a WMS instance.

Two informative diagrams (Transaction Throughput and Response Time) are generated for each WMS instance that a load performance test is created for in the cloud (Figure 40). The graph on the left shows how transaction throughput time increases with increasing number of threads (concurrent users) until it levels out. So basically, it shows the statistical maximum possible number of transactions based on number of users accessing the WMS. Any more users that query the WMS
at the same time will not affect the transaction throughput time anymore. In this case that happens after 35 to 40 users for the tiled service (orange line) and already at between 5 to 8 users for the untiled service (blue line). The graph on the right shows 2 interesting things. First, the untiled service (blue line) has significant slower responses with increasing threads (from 2 to 14 seconds). Secondly, for the tiled WMS service, the response time does start to increase with increasing users at around 30-35 threads but stays below 2 seconds, even with a 100 threads (users, each querying the WMS with 50 requests).

![Graph showing transaction throughput and response time](image)

*Figure 40. Transaction Throughput and Response Time, both per increasing number of threads.*

### 5.5.10. Integration with other QoS tools

JMeter stress tests are ran, over time, for a variety of WMS provided by Natural Resources Canada (NRCan). NRCan is in the process of migrating some of their provided WMS to another platform, and this provides a perfect opportunity to see if the hosted WMS will perform better when migrated over. The stress tests reports will be made available online, such that they can be easily integrated into the “GUI for WMS Service Quality Assessment” effort mentioned above.

### 5.6. GeoServer Extension for QoS

Over the past year, the OGC QoSE DWG has established a set of basic QoS indicators as acknowledged key performance indicators of spatial data service instances or endpoints. The aim is to improve the quality of the web services and Application Programming Interfaces (APIs), including factors like availability, capacity and performance by using well-defined metrics in order to provide comparable QoS measurements. The declared QoSE metadata will contribute to a complete picture of the QoS of the entire Spatial Data Infrastructure (SDI), and more effectively maintain high QoE for its end-users.

GeoSolutions has implemented an extension for WMS and Web Feature Service (WFS) functionality in GeoServer that allows the administrator to declare several statements of the service such as:

- **Operating Info** - [Video Tutorial](https://youtu.be/EQaeDAkLySc)
  - Operational status (test/demo/beta/production etc.)
  - Operating days & hours (default: 24/7)
Figure 41. Operating Info

- QoS statements of the entire service - Video Tutorial [https://youtu.be/JpimmBmmxAU]
  - Metrics & minimum values to be expected, performance, availability, capacity etc.

Figure 42. Metrics Statements Declaration

- Operation Anomaly Feed - Video Tutorial [https://youtu.be/6zFrfhz6cEM]
  - Maintenance periods, downtimes

Figure 43. Operation Anomaly Feed
• Representative operations - Video Tutorial [https://youtu.be/FS-h7wjf0TI]
  ◦ QoS statements for given operations & limited request parameters
  ◦ Auto-configuration for QoS monitoring tools
Figure 44. Representative operations
Documentation on how to install and configure the module can be found [here](http://docs.geoserver.org/latest/en/user/community/qose-module/index.html).

The latter statements declared by the user will be embedded in the Extensible Markup Language (XML) GetCapabilities file of the WMS or the WFS service.

A few examples for extended GetCapabilities responses are provided below and more can be found in the GitHub repository for the QoSE DWG [1: https://github.com/opengeospatial/QoSE-DWG/tree/master/QoSMetadata].

**Declaring operating hours for a WMS service**

```xml
<qos-wms:QualityOfServiceMetadata>
  <qos:OperatingInfo>
    <qos:ByDaysOfWeek>
      <qos:On>Monday Tuesday Wednesday Thursday Friday</qos:On>
      <qos:StartTime>09:00:00+01:00</qos:StartTime>
      <qos:EndTime>18:00:00+01:00</qos:EndTime>
    </qos:ByDaysOfWeek>
  </qos:OperatingInfo>
</qos-wms:QualityOfServiceMetadata>
```

**Declaring expected Quality of Service metrics for a WMS service**

```xml
<qos:QualityOfServiceStatement>
  <qos:MoreThanOrEqual uom="%">99</qos:MoreThanOrEqual>
</qos:QualityOfServiceStatement>
```

**Declaring a Live Monitoring Service**

```xml
  <ows:Abstract>Monitoring Summary for WMS in Cloudsdi</ows:Abstract>
  <ows:Format>html</ows:Format>
</qos:OperationAnomalyFeed>
```

**For Expert Users:**

The same operations can be done via the GeoServer REST Interface. A video tutorial can be found [here](https://youtu.be/zNivPjSTRik).

### 5.6.1. GeoSolutions Methodological Approach on Monitoring QoS

It is very important to make the QoS service very reliable, credible and transparent in terms of
metadata declaration. Thus, GeoSolutions have built a Live Monitoring Service [http://monitoring.geosolutions.it/?lang=en] based on GeoHealthCheck [http://docs.geohealthcheck.org/en/latest/] that monitors the health and the availability of the GeoServer services which will be declared in turn in the QoS module of GeoServer and therefore in the GetCapabilities file of the service.

![Figure 45. Monitoring Services and QoSE Module in Geoserver](image)

The figure above shows a schema of how to explicitly monitor external services in the QoSE module in GeoServer in order to offer a reliable service to the users.

The upper side of the diagram in Figure 45 represents an external monitoring service which continuously sends GetCapabilities or GetMap requests to GeoServer on the entire workspace or on single layers to control the endpoints of the WMS and WFS services of GeoServer.

![Figure 46. WMS and WFS Live Monitoring with GeoHealthCheck](image)

The lower side shows that the monitoring service can be declared in the QoSE module of GeoServer linking it in the Operation Anomaly Feed section of the module to demonstrate the reliability of the service to the users.
Accordingly, once a user requests the WMS or WFS GetCapabilities of the server, the user will have a way to directly check the monitoring service.

**Monitoring Service Link in the GetCapabilities XML File**

```
  <ows:Abstract>Monitoring Summary for WMS in Cloudsdi</ows:Abstract>
  <ows:Format>html</ows:Format>
</qos:OperationAnomalyFeed>
```

### 5.7. OGC Web Service Landing Pages

It is difficult for an end-user to directly ascertain the quality of many of the existing OGC Web Services due to the fact that the specifications for these services define only machine-oriented operations. That is, they define the set of operations that a client application can use to discover and interact with the various offerings of the server, but they do not specify any standard way for an end-user to discover the offerings of the server directly using only a web browser. A sophisticated end-user who is well versed in the specification(s) for the service can manually construct a GetCapabilities request and browse the XML response. However, this is far from user friendly, and is not a viable approach for all but the most sophisticated of end-users.

The ‘base’ Uniform Resource Locator (URL) of an OGC Web Service is typically a URL to which one or more **mandatory** parameters must be added. Without such parameters, the service is required by its specification to return an exception report. Use of such parameters is not user-friendly. One such required parameter is the “SERVICE” parameter indicating the type of service being invoked. This is typically not part of the base URL in order to allow a single base URL to support multiple services. Another such required parameter is the “REQUEST” parameter indicating the operation being invoked. The current OGC Web Service specifications (with the exception of WFS 3.0) simply do not define any human-readable landing page for allowing an end-user to browse the offerings of an OGC Web Service via a web browser.

The solution to this is pretty obvious. The OGC Web Service specifications should be augmented to provide this functionality. To this end, this engineering report makes the following recommendations.

1. All future OGC Web Service specifications should require the base URL of the service to be sensitive to the HTTP "Accept" header. If this header indicates that text/html is preferred, then the server should be required to return an HTML landing page presenting the offerings of the
server in a human-readable format, providing hyperlinks as necessary for navigation. This landing page should provide as much metadata as it can about each offering, including (but not limited to):

- Titles and abstracts, ideally in the language that is the closest match to the language requested by the HTTP "Accept-Language" header.
- Spatial extents where applicable. Ideally these spatial extents should be displayed graphically in a map.
- Source attribution, where applicable and available.
- Any other available measure(s) of data quality.

2. If the OGC Web Service specification defines a traditional machine-readable capabilities document (like the classic OGC Web Services do), then the capabilities-document endpoint (regardless of whether it is a RESTful or service-oriented endpoint) should be sensitive to the HTTP "Accept" header and should be required to return the HTML landing page if the header indicates that text/html is preferred. In general, the HTML landing page should provide roughly the same information that is available through the capabilities document, as well as extra measures of data quality where available.

3. If the OGC Web Service specification defines a set of service-oriented operations that indicate the type of operation via SERVICE and REQUEST parameters (like the classic OGC Web Services do), it should make the REQUEST parameter optional and require that the absence of the REQUEST parameter be interpreted as a request for the HTML landing page. Perhaps it could be sensitive to the HTTP "Accept" header and return either the HTML landing page or the machine-readable capabilities document as appropriate (in which case the absence of a REQUEST parameter becomes the equivalent of REQUEST=GetCapabilities).

4. If the OGC Web Service specification defines a GetCapabilities request with an AcceptFormats parameter (like the classic OGC Web Services that are based on the OGC Web Services Common Specifications do), then it should recognize an AcceptFormats value of text/html to indicate a request for the HTML landing page.

Change requests have been submitted to augment the behaviors of the commonly-used existing OGC Web Services specifications to include this functionality. Specifically, change requests have been made against the following existing specifications:

- Request 559, for OGC Web Services Common Standard 2.0.0 (OGC 06-121r9). All OGC Web Services that are based on this specification will therefore automatically inherit this augmented behavior. These services include, but are not necessarily limited to, the WCS 2.0 Interface Standard (OGC 09-11-r4, etc.) and the Web Integration Service (OGC 16-043). This change request should make the SERVICE parameter optional in the situation where a Web Integration Service is available. It should require that the absence of the SERVICE parameter in this situation be interpreted as a request for the HTML landing page of the Web Integration Service. This, in addition to making the REQUEST parameter optional, allows the raw base URL (i.e., without any parameters) of an OGC Web Service to return an HTML landing page for the suite of services that it provides.

- Request 560, for OGC Web Services Common Specification 1.1.0 (OGC 06-121r3). All OGC Web Services that are based on this specification will therefore automatically inherit this augmented behavior. These services include, but are not necessarily limited to, the Web Map Tile Service
(WMTS) Implementation Standard (OGC 07-057r7) and the OpenGIS Web Feature Service (WFS) 2.0 Interface Standard (OGC 09-025r1).

- Request 561, for Web Map Server Implementation Specification 1.3.0 (OGC 06-042). Unfortunately, this latest WMS specification predates the OGC Web Services Common specifications, so a separate change request had to be made specifically for it.

No change request needs to be made against the WFS 3.0 specification, since it already defines this behavior.

A proof-of-concept implementation of these recommendations has been made available at https://tb14.cubewerx.com/cubewerx/cubeserv for the duration of the OGC Testbed-14 initiative (tb14guest credentials need to be provided to see the full suite of services available).

5.8. TIE and Scenario for Demonstration

5.8.1. TIE Component Implementation

The component overview graphic at the beginning of this section illustrates how the different components described above fit together. The following scenario will be used to demonstrate this in an example demo video.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Service Request</th>
<th>Tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>D115: HTML Landing Page</td>
<td>Additional quality measures to the capabilities document</td>
<td>WMS server</td>
<td>Y: TIE on 07/04/18</td>
</tr>
<tr>
<td>D116/D117: GUI of 14 Assessment Criteria</td>
<td>Qualitative Experience Evaluation: draft implementation and testing</td>
<td>WMS instances</td>
<td>Y: TIE on 09/18/18</td>
</tr>
<tr>
<td>D116/D117: GUI of 14 Assessment Criteria</td>
<td>Qualitative Experience Evaluation: implementation of NRCan recommendations</td>
<td>WMS instances</td>
<td>Y: TIE on 10/23/18</td>
</tr>
<tr>
<td>D121: WMTS Portrayal with QoSE support</td>
<td>GeoServer Extension for QoS, including service performance, operating capabilities (for QoSE)</td>
<td>WMS server</td>
<td>Y: TIE on 09/25/18</td>
</tr>
<tr>
<td>D122: Additional Client Support</td>
<td>Service performance testing (Quantitative; JMeter)</td>
<td>WMS server(s)</td>
<td>Y: TIE on 10/30/18</td>
</tr>
</tbody>
</table>

5.8.2. Scenario "Blueprint" for the Result Demonstration Video

After setting the story for the demonstration video including the material and short individual
recordings describing each component as outlined at the start of this chapter, the Technology Integration Experiment (TIE) showcasing followed the steps below:

**Step 1:** Set up a couple of WMS instances that will be used to demonstrate the workflow.

**Step 2:** Introduce (briefly) a couple of WMS instances

**Step 3:** Use those instances to run through the Assessment GUI for user-based evaluation/rating

**Step 4:** Pull in the JMeter performance statistics of those WMS instances/services

**Step 5:** Showcase the GeoServer Extension for those WMS instances

**Step 6:** Illustrate the HTML Landing Page and its value to the community.
## Appendix A: Revision History

### Table 2. Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Editor</th>
<th>Release</th>
<th>Primary clauses modified</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 15, 2016</td>
<td>I. Simonis</td>
<td>.1</td>
<td>all</td>
<td>initial version</td>
</tr>
<tr>
<td>July 22, 2016</td>
<td>I. Simonis</td>
<td>.9</td>
<td>all</td>
<td>comments integrate</td>
</tr>
<tr>
<td>September 7, 2016</td>
<td>S. Simmons</td>
<td>1.0</td>
<td>various</td>
<td>preparation for publication</td>
</tr>
<tr>
<td>March 23, 2017</td>
<td>I. Simonis</td>
<td>2.0</td>
<td>all</td>
<td>template simplified</td>
</tr>
<tr>
<td>January 18, 2018</td>
<td>S. Serich</td>
<td>2.1</td>
<td>all</td>
<td>additional guidance to Editors; clean up headings in appendices</td>
</tr>
<tr>
<td>October 4, 2018</td>
<td>G. Hobona</td>
<td>2.1</td>
<td>all</td>
<td>comments to Editors</td>
</tr>
<tr>
<td>October 16, 2018</td>
<td>G. Schumann</td>
<td>2.1</td>
<td>all</td>
<td>implementation of comments to Editors</td>
</tr>
<tr>
<td>October 30, 2018</td>
<td>G. Schumann</td>
<td>2.1</td>
<td>all</td>
<td>Completion of Future Work section</td>
</tr>
<tr>
<td>November 13, 2018</td>
<td>A. Kettner</td>
<td>2.1</td>
<td>all</td>
<td>Completion/Revision of JMeter section</td>
</tr>
<tr>
<td>November 20, 2018</td>
<td>G. Schumann</td>
<td>2.1</td>
<td>all</td>
<td>ER revision &amp; implemented comments from NRCan and the QoSE DWG</td>
</tr>
<tr>
<td>November 22, 2018</td>
<td>Zelong</td>
<td>2.1</td>
<td>all</td>
<td>Completed description of QoSE GUI</td>
</tr>
<tr>
<td>November 28, 2018</td>
<td>G. Schumann</td>
<td>2.1</td>
<td>all</td>
<td>Second ER revision</td>
</tr>
</tbody>
</table>