Testbed-12 OWS Common Security Extension ER

Table of Contents

Publication Date: 2016-03-10

Approval Date: 2016-12-07

Posted Date: 2016-11-15

Reference number of this document: OGC 16-048r1

Reference URL for this document: http://www.opengis.net/doc/PER/t12-A065

Category: Public Engineering Report

Editor: Andreas Matheus

Title: Testbed-12 OWS Common Security Extension ER

OGC Engineering Report

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 USERâ€™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

The OGC suite of standards address the interoperable exchange of geographic information. The Web Service Implementation Standards define the discovery, delivery, and processing services that make information exchange possible. Common aspects of those Web Service standards have been collected into the OGC Web Services Common standard. While there are multiple versions of OWS Common, and flexibility in how it is applied, this combination of standards does enable interoperability.

However, OWS Common neglected to address security. As soon as a service endpoint (an OGC Web Service instance) is secured, there is no guarantee of interoperability.

The OWS Common - Security Standards Working Group (SWG) was approved by the TC in September 2015 (http://www.opengeospatial.org/projects/groups/comsecurityswg). It held its first meeting during the December 2015 TC meetings. The objective of this SWG to define an extension to the existing OWS Common to ensure interoperability between a secured service instance and client. This "OWS Common Security Extension" adds content to the standard regarding the implementation of security controls in such a way as to preserve interoperability. These additions will be in two areas. The first extension will provide more detail on the use of the HTTP protocol, particularly as it related to security controls. The second extension will address discovery and negotiation of security controls. This will provide an annotation model for the Capabilities document to enable a service provider to specify the security implemented at a service instance (endpoint).

This ER shall serve as the technical background to the OWS Common - Security SWG to ensure that the standard that is to be created is comprehensive and suitable for all OGC Web Services standards, to overcome the interoperability hurdle, and - at the same time - maintain backwards compatibility.

Business Value

Many OGC services are deployed behind security controls. This constrains interoperability to those services designed to work within a common security infrastructure. The Business Value of this Engineering Report is that it provides a way to break that constraint while maintaining the security and infrastructure integrity. It provides the technical background and tests needed to standardize security across OGC Web Services. The use of the findings of this ER,

standardized by the OWS Common Security SWG, would then allow secured OGC Web Services to discover and negotiate mutually acceptable security controls. In that sense, OGC Services would again guarantee that access to protected geographic information can be assured. Use cases from the intelligence and commercial sector do benefit from this ER.

What does this ER mean for the Working Group and OGC in general

This ER is vetted by the OWS Common - Security SWG. In particular, this ER provides the technical background for the standardization of the interoperability aspects for OGC Web Services. The SWG has planned to release a normative standard, but also Best Practices and guidance. All of these topics are supported by the ER, but in particular the latter two topics.

This ER does have wider consequences for the OGC members as it provides options to ensure interoperability for secured OGC Web Services which extend the current use into other communities, such as commercial uptake. Assuming the approach described in this ER gets standardized by the OWS Common - Security SWG, it is expected that all future OGC Web Service standards address security and ensure compatibility with the approach.

How does this ER relate to the work of the Working Group

This ER provides the technical groundwork of outlining an interoperability framework to enable security on OGC Web Services. It is clear that the suite of OGC Web Services standards is supporting different versions of OWS Common and may define individual behavior that has an impact on how the OWS Common Security Extension must be worded.

In particular, the approach is to ensure that a client is capable to extract from a Capabilities document - via security annotations - which service operations it is capable of supporting.

Keywords

ogcdocs, testbed-12, Security, OWS Common, OWS Common Security Extension

Chapter 1. Introduction

1.1. Scope

The overall scope of this ER is to serve as technical background for the standardization of the "OWS Common Security Extension v1.0" as currently being worked by the OWS Common - Security SWG.

During the last OGC Testbeds, various ERs were written that examine the situation of security for OGC Web Service standards. The most recent and dominant ER is OGC 15-022. It provides a comprehensive overview to the problem space "interoperability of protected" OGC Web Services but also introduces an approach regarding possible solutions.

Starting off with the recommendations from OGC 15-022, the following scope for this ER becomes reasonable:

- Clarification of the OGC Web Services model with regard to security;
- Clarification of the HTTP protocol used with the OGC Web Services;
- Annotation of Capabilities and ISO metadata to outline security in a machine-understandable fashion; and
- Client requirements.

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**
Andreas Matheus	University of the Bundeswehr
Charles Heazel	WiSC Enterprises
Panagiotis Vretanos	Cubewerx
Robin Houtmeyers	Luciad
Ziheng Sun	GMU
Craig Coffin	Compusult
Marten Hogeweg	esri
Peter Baumann	rasdaman
Michael Leedahl	DigitalGlobe

1.3. Future Work

For the topic of security, two different origins for future work exist: (i) Security DWG and (ii) Testbed 12 (TB 12).

Future work items originating from the Security DWG session from the Dublin TC Meeting (June 2016) are:

- Tagging of imagery to control processing / use of the data;
- Fusion of tagged imagery;
- Security Quota based access;
- SAML and OAuth support in Esri ArcGIS / Envitia MapLinkPro desktop client;
- Multiple Levels or authentication (user / machine);
- Delegation of Identity when chaining secured OGC services;
- BPMN (Business Process Modelling Notation) to craft a workflow of protected services;
- REST and security (explore the Main Stream IT use of ALL HTTP verbs including OPTIONS, HEAD, etc.);
- TLS with client/user certificate (mutual TLS); and
- OWSContext with Information Flow / Access Control based on user characteristics.

Future work items originating from the work currently undertaken in TB12 are as follows.

- Client development to parse and use the security annotations introduced for secured services from TB12. This ER lists several examples of security annotations in the Capabilities document.
- Expansion of the <ows:Constraint/> element to ensure more expressiveness by introducing a CNF (Conjunctive Normal Form) structure.
- OGC Naming Authority to approve authentication code URNs, the security markup URNs and the URL for the Authentication Code List.
- Explore the use of annotated Capabilities in a services workflow with secured services.
- Related to the work described in OGC 16-049 (Testbed-12 Multi-Tile Retrieval ER): the GetTiles methodology described in this ER requires implementing security controls on a service chain if leveraging a WPS. The OWS Common - Security SWG will investigate the security requirements inherent in service chaining and the GetTiles methodology in particular.
- The availability of one and only DCP type "HTTP" in the OWS Common schema is problematic. For example, there is no way to indicate support for HTTPS. The OWS Common Security SWG will re-visit the Distributed Computing Platform (DCP) concept as originally documented in OGC Catalog 1.0 with the intent of more effectively using that concept to convey security related features and constraints.

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.

It is expected that this document may result in the draft standard or input to the informative annex of "OWS Common Security Extension Standard" currently developed in the OGC OWS Common - Security SWG.

Chapter 2. References

- OGC 15-022. OGC Testbed 11 Implementing Common Security Across the OGC Suite of Service Standards
- OGC 06-121r3.OGC® Web Services Common Standard
- OGC 04-095.OpenGIS® Filter Encoding Implementation Specification
- ISO 35.100.Open systems interconnection (OSI)
- ISO/IEC 10181-1.Information technology — Open Systems Interconnection — Security frameworks for open systems: Overview. ISO 1996: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=24404
- ISO/IEC 10181-2. Information technology — Open Systems Interconnection — Security frameworks for open systems: Authentication framework. ISO 1996: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=18198
- ISO/IEC 10181-3. Information technology — Open Systems Interconnection — Security frameworks for open systems: Access control framework. ISO 1996: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=18199
- ISO/IEC 10181-4. Information technology — Open Systems Interconnection — Security frameworks for open systems: Non-repudiation framework. ISO 1996: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=23615
- ISO/IEC 10181-5. Information technology — Open Systems Interconnection — Security frameworks for open systems: Confidentiality framework. ISO 1996: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=24329
- ISO/IEC 10181-6. Information technology — Open Systems Interconnection — Security frameworks for open systems: Integrity framework. ISO 1996: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=24330
- ISO/IEC 10181-7. Information technology — Open Systems Interconnection — Security frameworks for open systems: Security audit and alarms framework. ISO 1996: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=18200
- IETF RFC 2616. Hypertext Transfer Protocol — HTTP/1.1. http://tools.ietf.org/html/rfc2616
- IETF RFC 2617. HTTP Authentication: Basic and Digest Access Authentication. https://tools.ietf.org/html/rfc2617
- IETF RFC 2246. Transport Layer Security. http://tools.ietf.org/html/rfc2246
- IETF RFC 2818. HTTP Over TLS. http://tools.ietf.org/html/rfc2818
- ITU-T X.509 / PKI. Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks. ITU-T Standard. 08/2005: http://www.ietf.org/html.charters/pkix-charter.html
- IETF RFC 2396. Uniform Resource Identifiers (URI): Generic Syntax. https://tools.ietf.org/html/rfc2396
- IETF RFC 3280. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile – IETF RFC 3280: http://tools.ietf.org/html/rfc3280

- IETF RFC 2109. HTTP State Management Mechanism – IETF RFC 2109: https://tools.ietf.org/html/rfc2109
- OASIS. Web Services Security: SOAP Message Security 1.1 (WS-Security 2004) – OASIS Standard Specification. 1 February 2006: http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
- W3C. XML Digital Signature: XML-Signature Syntax and Processing – W3C Recommendation 12 February 2002: http://www.w3.org/TR/xmldsig-core/
- W3C. XML Encryption: XML Encryption Syntax and Processing – W3C Recommendation 10 December 2002: http://www.w3.org/TR/xmlenc-core/
- W3C. XML Signature Best Practices: XML Signature Best Practices – W3C Working Group Note 11 April 2013: http://www.w3.org/TR/2013/NOTE-xmldsig-bestpractices-20130411/
- W3C. XML Linking Language (XLink) Version 1.0. W3C Recommendation 27 June 2001: http://www.w3.org/TR/xlink/
- W3C. Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001: http://www.w3.org/TR/wsdl
- W3C. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Recommendation 26 June 2007: http://www.w3.org/TR/wsdl20/
- OASIS. UDDI Spec Technical Committee Draft. OASIS. Dated 20041019: http://www.uddi.org/pubs/uddi_v3.htm
- W3C. Web Services Policy 1.5 – Framework. W3C Recommendation 04 September 2007: http://www.w3.org/TR/2007/REC-ws-policy-20070904/
- W3C. Web Services Policy 1.5 – Attachment. W3C Recom-mendation. 04 September 2007: http://www.w3.org/TR/ws-policy-attach/
- W3C. WS-SecurityPolicy 1.2. OASIS Standard. 1 July 2007: http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf

Chapter 3. Terms and definitions

3.1. Authentication

ISO 10181-2 defines all basic concepts of authentication in Open Systems: it identifies different classes of authentication mechanisms, the services for their implementation and the requirements for supporting protocols. It further identifies requirements for the management of identity information.

3.2. Authorization

ISO 10181-3 defines all basic concepts for access control and authorization in Open Systems and the relation to other frameworks such as the Authentication and Audit Frameworks.

3.3. Confidentiality

ISO 10181-5 defines the basic concepts of confidentiality, identifies classes of confidentiality mechanisms and their maintenance. It further addresses the interactions of the confidentiality mechanisms with other services.

3.4. Integrity

ISO 10181-6 defines the basic concepts of integrity, identical to the Confidentiality Framework.

Chapter 4. Conventions

4.1. Abbreviated terms

- APIApplication Program Interface
- CNCommon Name (X.509 certificate)
- CNF Conjunctive Normal Form
- COTSCommercial Off The Shelf
- DCEDistributed Computing Environment
- DCOMDistributed Component Object Model
- DCPDistributed Computing Platform
- DNF Discrete Normal Form
- GeoXACMLGeospatial eXtensible Access Control Markup Language
- HTTPHypertext Transfer Protocol
- ISOInternational Organisation for Standardisation
- KVPKey Value Pair
- MIMEMultipurpose Internet Mail Extension
- OASIS Organization for the Advancement of Structured Information Standards
- OGC Open Geospatial Consortium
- OWSOGC Web Service
- W*SOGC Web Service
- PKIPublic Key Infrastructure
- SAML Security Assertion Markup Language
- URIUniform Resource Identifier
- UMLUniform Resource Locator
- URNUniform Resource Name
- WFSWeb Feature Service
- WMSWeb Map Service
- WMTS Web Map Tile Service
- WSDLWeb Services Description Language
- XACMLeXtensible Access Control Markup Language
- XMLExtensible Markup Language

Chapter 5. Overview

The current set of OGC Web Services specifications describe a large variety of interfaces to implement a service-type-specific API. All of these interfaces leverage the Hyper Text Transmission Protocol (HTTP). In particular, the existing OGC Web Service specifications describe the use of the HTTP methods GET and POST to interact with the service implementation at a network endpoint. Most of the commonalities regarding the use of HTTP methods and status codes are defined in OGC Web Services Common (OWS Common). Different versions of OWS Common exist. Each version describes sightly different restrictions how to use the HTTP protocol.

As pointed out in OGC 15-022, all OGC W*S based capabilities (with the exception of WMS) are based on the OWS Common structure.

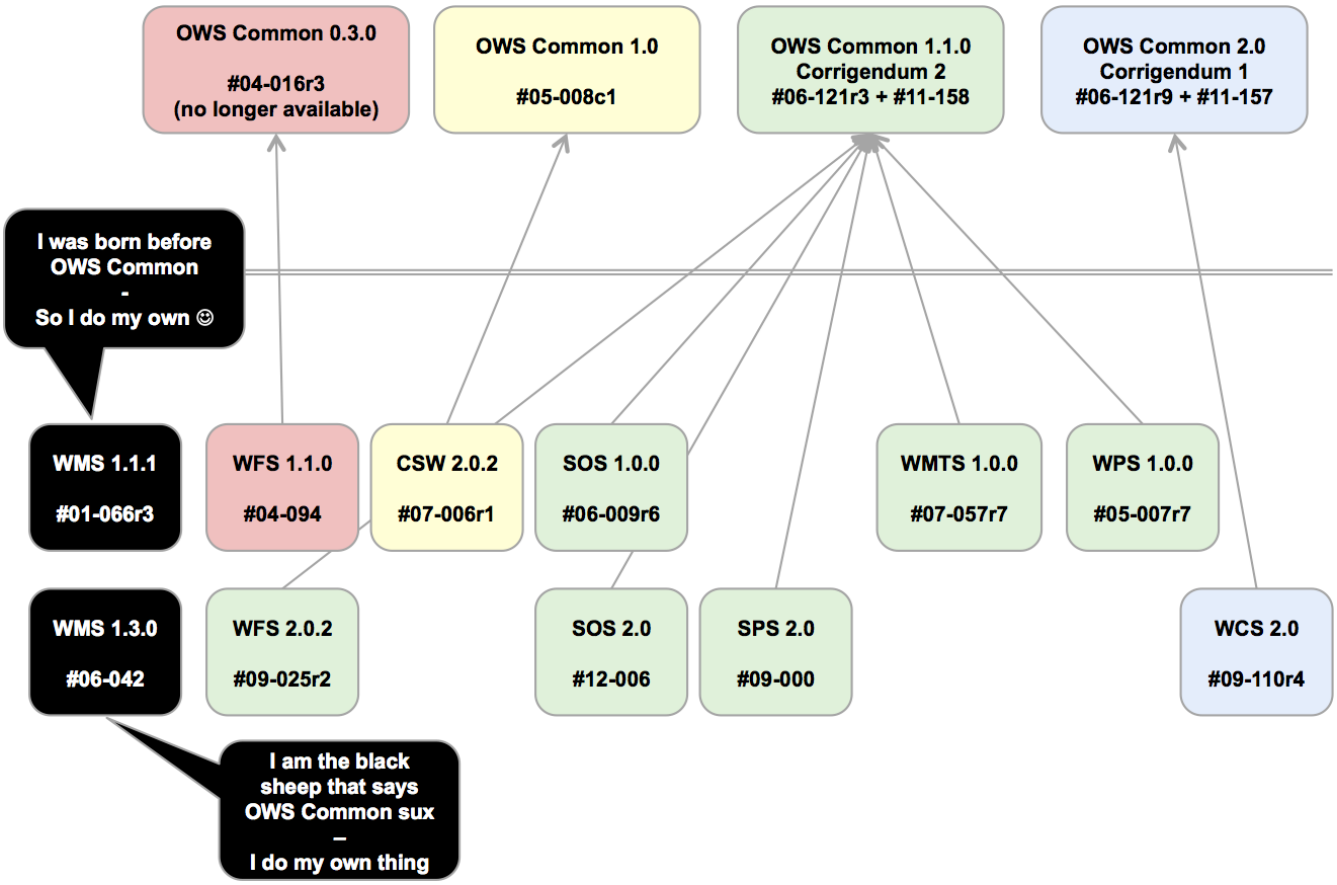

*Figure 1. Use of OWS Common in W*S Capabilities*

When implementing security on a service endpoint, it is important to inform the client of the security controls supported. This allows the client to determine which required "security handshake" the service supports and if it is possible to bind to the service in the first place. Obviously, only those service endpoints and operations can be executed for which the client has implemented support.

Informing clients of OGC Web Services about security controls is difficult. OWS Common has limitations which inhibit expressing this information. For example, OWS Common only supports the HTTP GET and POST methods. Also, the question arises as to how the service shall advertise the security requirements such as authentication, authorization, integrity and confidentiality.

This ER first explores the scope of security protocols and controls that are used in mainstream IT

systems. It then addresses the limitations of OWS Common when describing controls for the HTTP protocol. Then, it provides a proposal how the protected service endpoints can advertise security requirements to a client using the OGC Capabilities document. Finally, it illustrates a proof of concept by applying different security conditions to the Capabilities document from various W*S and versions and the results of various Technology Integration Experiments with existing client applications. The Annex contains a complete list of protected services and their annotated capabilities documents.

Chapter 6. OGC Web Services and Mainstream IT

NOTE

OGC Web Services are partly disconnected from mainstream IT

```
It is strongly recommended to apply security to OGC Web Services based
on mainstream Information Technology (IT) standards and solutions. But
in order to do so, it is important to understand where OGC Web Services
are similar to mainstream IT and where those services are not.
```

The foundation of interoperability for Web Services in general and OGC Web Services in particular is introduced by the OSI/ISO model (Open Systems Interconnection) as standardized in ISO 7498-1 (informative source: https://en.wikipedia.org/wiki/OSI_model). It is important to note that the interoperability introduced (ensured) by the OSI model is independent from the characteristics of the data exchanged.

For OGC Web Services, the communication protocol for Layer 7 (application) is limited to HTTP 1.1 as standardized by IETF as RFC 2616. A mainstream-compatible solution of a service and a client (e.g. a Web Browser) can be based on the fact that both are fully compliant. But, for OGC Web Services according to the W*S implementation specifications, no full support of HTTP is required. This has implications for applying security.

All OGC Web Services are stateless, which implies that any state must be managed by the client. When applying security controls, usually a security context is involved that can be used to manage access to a secured (protected) service endpoint. Typically, the security context contains (but is not limited to) authentication information about the user. This security context is typically stored at the server side and the client uses some mechanism to refer to it. When managing access to a protected service, this information might be considered. It is therefore relevant, that the client can provide a "pointer" to the security context. Different authentication models implement this pointer differently. For example HTTP Basic Authentication, as standardized in RFC 2617, uses the HTTP Header "Authorization" as the pointer. More complex authentication models require the support of HTTP Cookies as standardized in IETF RFC 6265 (former 2965 and 2109).

In terms of confidentiality and integrity, it is recommended practice to constrain the communication to HTTPS to ensure confidential communication between the entities. HTTPS is standardized in IETF 2818 and basically defines the use of HTTP 1.1 over an encrypted communication channel such as TLS. Transport Layer Security (TLS) version 1.2 (which is the de-facto version to use) is standardized by IETF RFC 5246.

6.1. Common Security Concepts

The ISO 10181 series, which is identical to ITU-T X.81* body of standards, defines a collection of security frameworks for open distributed processing. These standards are part of the ITU-T X-Series Recommendations for Data Networks and Open Systems Communications. While there are many other security frameworks available, the scope and history of the ITU-T (formerly CCITT) makes their frameworks particularly applicable to the Web Services domain.

This section provides a summary of these frameworks. An understanding of these concepts will be assumed throughout the rest of this Engineering Report.

6.1.1. Authentication (ITU-T X.811)

Many Open Systems applications have security requirements which depend upon correctly identifying the principals involved. Such requirements may include the protection of assets and resources against unauthorized access, for which an identity based access control mechanism might be used, and/or the enforcement of accountability by the maintenance of audit logs of relevant events, as well as for accounting and charging purposes. The process of corroborating an identity is called authentication.

ITU-T combines the concepts of identification and authentication into a single framework. Identification asserts an identity for a party in a transaction. Authentication validates that the assertion is correct. For example, a user name asserts an identity. Authentication is only achieved once the password has been validated.

6.1.2. Access Control (ITU-T X.812)

The primary goal of access control is to counter the threat of unauthorized operations involving a computer or communications system; these threats are frequently subdivided into classes known as unauthorized use, disclosure, modification, destruction and denial of service.

Access control is arguably the most complex of the security frameworks. It typically consists of the authenticated identity of the party requesting access, metadata describing the access restrictions for a resource, and some form of access policy enforcement which governs whether access is granted or refused. Some common forms of access control are as follows.

- Access control lists: If the requesting party is not on the list, they do not get access.
- Attribute Based Access Control: Each user is assigned a set of security attributes. Based on those attributes and the security markings on the resource, access is granted or refused. See ITU-T X.509.
- Secure Token Service: A Secure Token Service makes the access control decision once, then issues a token which grants access to a resource for a period of time. Most of these services use an automated system to make the access decision. OAuth differs for the Authorization Code flow as the request for approving access to the resource is send to the resource owner. This owner typically is an individual. This allows an individual to control access to their resources, even if they do not own or control the service which is hosting that resource.

6.1.3. Non-Repudiation (ITU-T X.813)

The goal of the Non-repudiation service is to collect, maintain, make available, and validate irrefutable evidence regarding identification of originators and recipients involved in data transfers.

Non-repudiation is typically accomplished through a digital signature. A digital signature generates a hash of the message, then encrypts the hash and additional identifying information using the users private key. If they signature can be decrypted using the users public key, and the hash

values match, then whoever signed the message must have had access to the users private key. If the private key has been protected, then only the user could have created and signed the message. Therefore they cannot repudiate the message.

6.1.4. Confidentiality (ITU-T X.814)

Many Open Systems applications have security requirements which depend upon the prevention of disclosure of information. Such requirements may include the protection of information used in the provision of other security services such as authentication, access controls, or integrity, that, if known by an attacker, could reduce or nullify the effectiveness of those services. Confidentiality is the property that information is not made available or disclosed to unauthorized individuals, entities, or processes.

Confidentiality is commonly achieved through encryption. Encryption can be applied to the resource itself or to the channel the resource is exchanged over. HTTPS, for example, provides confidentiality by running the HTTP protocol over an encrypted transport link.

6.1.5. Integrity (ITU-T X.815)

Many Open Systems applications have security requirements which depend upon the integrity of data. Such requirements may include the protection of data used in the provision of other security services such as authentication, access control, confidentiality, audit, and non-repudiation, that, if an attacker could modify them, could reduce or nullify the effectiveness of those services. The property that data has not been altered or destroyed in an unauthorized manner is called integrity.

The digital signature discussed above also provides integrity. If the hash algorithm is of cryptographic grade, then the chances of two messages generating the same hash are very low. So if the hash values before and after transmission match, then you can have confidence in the integrity of the message.

6.2. Security Use Cases

During the second session at the TC Meeting in Sydney (2015), use cases were identified that are relevant to OGC Web Service security. These use cases illustrate issues such as the following.

- The discovery of a protected service from a CSW?
- The discovery of protected data?
- The binding of a client to a protected service?
- How are security constraints described in ISO metadata?
- How are security constraints described in Capabilities documents?

6.2.1. Use Case #1 "Public Catalogue / Open Service"

The open use case: it’s an open environment and all is shareable. Services are available via a public network (Internet) and service metadata is registered in a public catalogue. ISO Metadata must not include accessContraint elements. Capabilities must not list access constraints.

This is a common practice.

6.2.2. Use Case #2 "Public Catalogue / Protected Service"

The Business Use Case: a provider wants users to know that there is a protected service registered in a public catalogue. They also want users to know that the content served is protected but not classified. The ISO Metadata indicates that the service endpoint(s) is(are) access protected. The ISO metadata also points to a public version of the Capabilities document. This public Capabilities document includes the content section and describes the content available from the service. The Public Capabilities document also describes the access constraints on a protected GetCapabilities operation. This protected GetCapabilities operation returns additional service metadata which the owner may not want to have publicly available.

6.2.3. Use Case #3 "Public Catalogue / Protected Service / Sensitive Data"

A Protected Service with sensitive data is registered in a Catalogue on a public network (i.e., the Internet). The ISO Metadata indicates that the service endpoint(s) is(are) access protected. The ISO metadata points to a public version of the Capabilities. The Public Capabilities document does not include a content section. The Public Capabilities document also describes the access constraints on a protected GetCapabilities operation. This protected GetCapabilities operation returns a full capabilities document including content section.

6.2.4. Use Case #4 "Public Catalogue / Protected Service / Dataset Partially Public / partially sensitive"

A service is registered in a public catalogue. Access to the service as an anonymous user provides knowledge about the non-sensitive data provided. Authenticated users can access knowledge about those data sets / layers that the user is authorized to see. Public metadata / capabilities outline the access constraints content section partially inline (non-sensitive data sets) partially using external links (sensitive data sets).

6.2.5. Use Case #5 "Protected Catalogue / Protected Service / Dataset sensitive"

A service that is just providing sensitive information, which is not meant for public / anonymous consumption, is registered in a protected catalogue. Users searching the catalogue must be authenticated. Access Control might additionally restrict the results provided to a user by enforcing the need-to-know principle.

The metadata describing the search results from the catalogue can be constructed according to the user’s permissions and therefore might different for users with different permissions / privileges.

6.3. Authentication Scenarios

6.3.1. Scenario "HTTP Authentication"

HTTP Authentication is specified by IETF RFC 7235. It is a challenge-response authentication scheme using HTTP headers and response codes to exchange the challenge and response. The

following example illustrates its use. This HTTP Authentication does not require a security context to be saved on the service side. Once valid authentication credentials have been established, the client can include those credentials with subsequent requests using the Authorization header. The authentication schemes supported by RFC 7235 are registered on the IANA HTTP Authentication Scheme Registry (http://www.iana.org/assignments/http-authschemes/http-authschemes.xhtml). Two commonly used authentication schemes are Basic and Digest. Basic authentication credentials are an unencrypted user name and password. Digest authentication also uses a user name and password however the values are encrypted prior to being exchanged. HTTP Authentication is illustrated by the sequence diagram in figure 2.

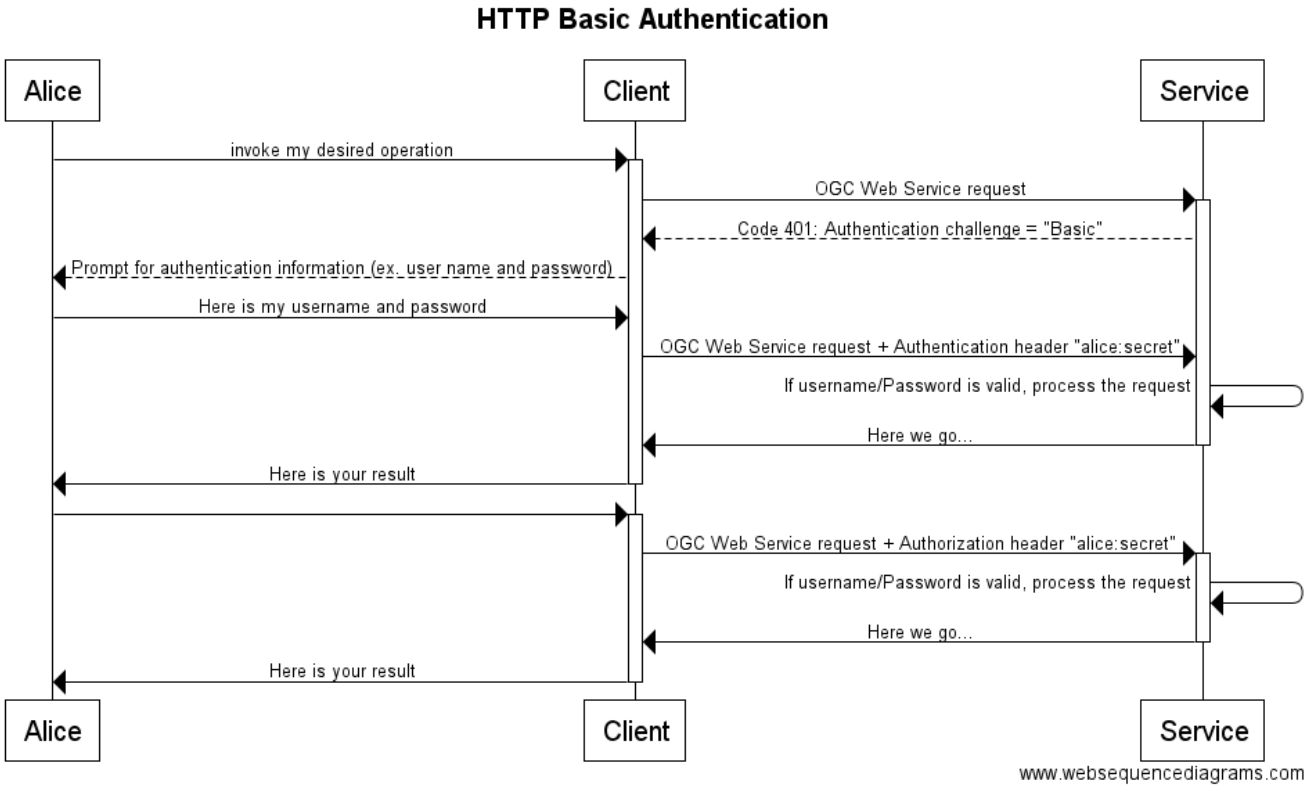

Figure 2. HTTP Authentication

```
title HTTP Basic Authentication

Alice->+Client:invoke my desired operation
Client->+Service: OGC Web Service request
Service-->Client: Code 401: Authentication challenge = "Basic"
Client-->Alice: Prompt for authentication information (ex. user name and password)
Alice->Client: Here is my username and password
Client->Service: OGC Web Service request + Authentication header "alice:secret"
Service->Service: If username/Password is valid, process the request
Service->-Client: Here we go...
Client->-Alice: Here is your result

Alice->+Client:
Client->+Service: OGC Web Service request + Authorization header "alice:secret"
Service->Service: If username/Password is valid, process the request
Service->-Client: Here we go...
Client->-Alice: Here is your result
```

6.3.2. Scenario "SAML Attribute Sharing Profile for X.509 Authentication-Based Systems"

The following example illustrates the use different "features" of the RFC 2818, 6265 and 2616:

- TLS client certificate handshake: The client must be able to provide a user (or client application) certificate to establish a secure channel with the service.
- HTTP response set-cookie: The service requests that the client "bakes" a cookie.
- HTTP response status code 302: The client must follow the redirect to the new location and - in this case applicable - submit the cookie with the redirect.
- HTTP request + Cookie: Client must submit the HTTP Cookie with the request.

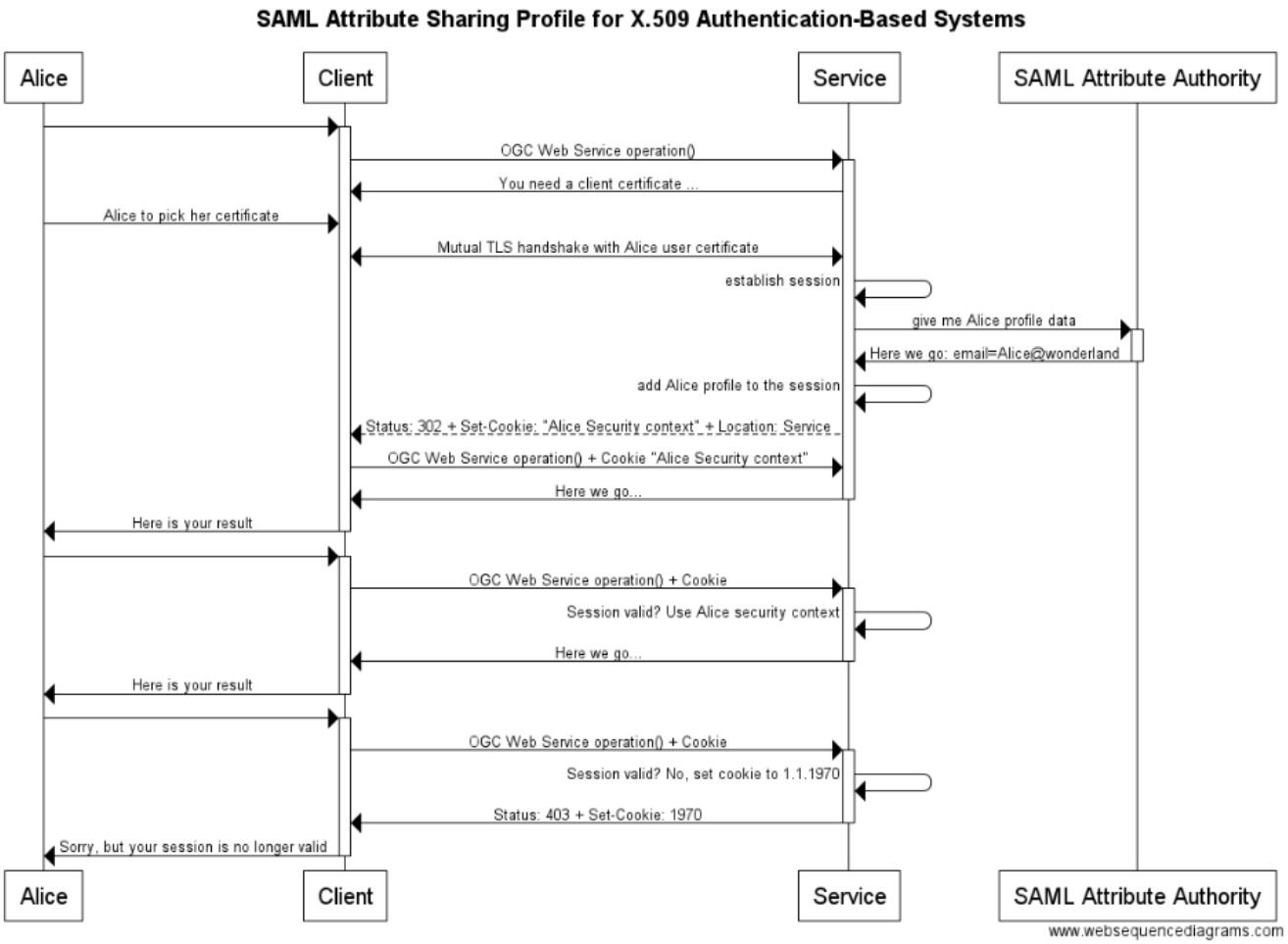

Figure 3. SAML Attribute Sharing Profile for X.509 Authentication-Based Systems

```
title SAML Attribute Sharing Profile for X.509 Authentication-Based Systems

Alice->+Client:
Client->+Service: OGC Web Service operation()
Service->Client: You need a client certificate ...
Alice->Client: Alice to pick her certificate
Client<->Service: Mutual TLS handshake with Alice user certificate
Service->Service: establish session
Service->+SAML Attribute Authority: give me Alice profile data
SAML Attribute Authority->-Service: Here we go: email=Alice@wonderland
Service->Service: add Alice profile to the session
Service-->Client: Status: 302 + Set-Cookie: "Alice Security context" + Location:
Service
Client->Service: OGC Web Service operation() + Cookie "Alice Security context"
Service->-Client: Here we go...
Client->-Alice: Here is your result

Alice->+Client:
Client->+Service: OGC Web Service operation() + Cookie
Service->Service: Session valid? Use Alice security context
Service->-Client: Here we go...
Client->-Alice: Here is your result

Alice->+Client:
Client->+Service: OGC Web Service operation() + Cookie
Service->Service: Session valid? No, set cookie to 1.1.1970
Service->-Client: Status: 403 + Set-Cookie: 1970
Client->-Alice: Sorry, but your session is no longer valid
```

6.3.3. Scenario "SAML Web Browser SSO + POST Binding"

The following example illustrates the use of HTTP redirect, HTTP cookies and HTTPS to leverage the SAML2 profile "Web Browser SSO" in conjunction with the "POST Binding."

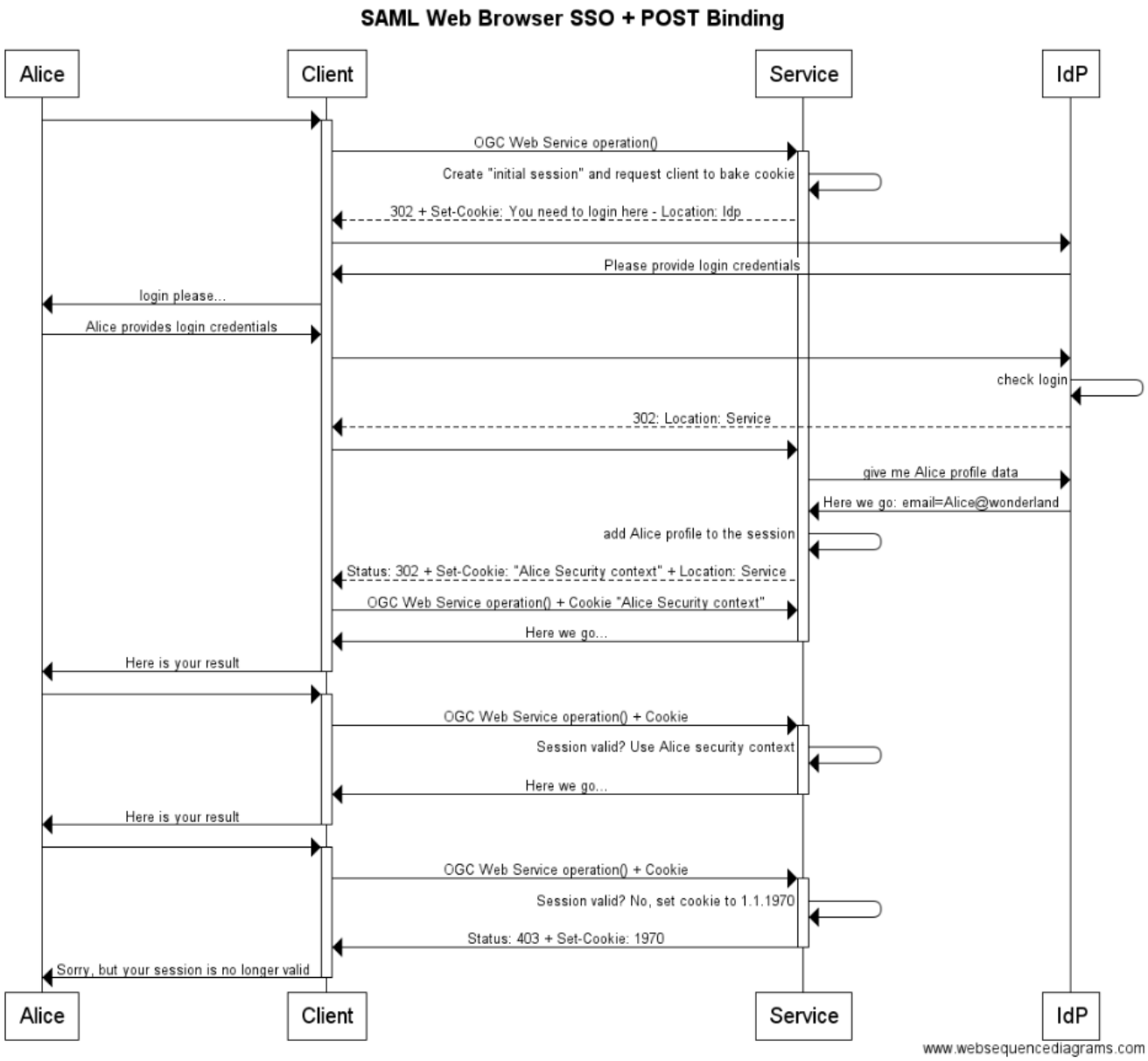

Figure 4. SAML Attribute Sharing Profile for X.509 Authentication-Based Systems

- Service creates "first visit" session and returns HTTP response status 302 + set-cookie: The user client gets redirected to another URL where the user shall login (IdP).
- IdP HTTP response status 302: Redirect the client back to the Service.
- Service returns HTTP response 302: Redirect to itself to fetch the cookie and update the session (e.g. with user profile information).
- The SAML message exchange between the service, the IdP and the service again are done using XHTML Form elements where the action is "onLoad()" of the HTML document. ⇒ The client must support JavaScript and it must be enabled to achieve automatic processing.

```
title SAML Web Browser SSO + POST Binding

Alice->+Client:
Client->+Service: OGC Web Service operation()
Service->Service: Create "initial session" and request client to bake cookie
Service-->Client: 302 + Set-Cookie: You need to login here - Location: IdP
Client->IdP:
IdP->Client: Please provide login credentials
Client->Alice: login please...
Alice->Client: Alice provides login credentials
Client->IdP:
IdP->IdP: check login
IdP-->Client: 302: Location: Service
Client->Service:
Service->IdP: give me Alice profile data

IdP->Service: Here we go: email=Alice@wonderland
Service->Service: add Alice profile to the session
Service-->Client: Status: 302 + Set-Cookie: "Alice Security context" + Location:
Service
Client->Service: OGC Web Service operation() + Cookie "Alice Security context"
Service->-Client: Here we go...
Client->-Alice: Here is your result

Alice->+Client:
Client->+Service: OGC Web Service operation() + Cookie
Service->Service: Session valid? Use Alice security context
Service->-Client: Here we go...
Client->-Alice: Here is your result

Alice->+Client:
Client->+Service: OGC Web Service operation() + Cookie
Service->Service: Session valid? No, set cookie to 1.1.1970
Service->-Client: Status: 403 + Set-Cookie: 1970
Client->-Alice: Sorry, but your session is no longer valid
```

6.3.4. Scenario "SAML ECP + PAOS Binding"

The following example illustrates the use of HTTP specific headers, HTTP cookies and HTTPS to leverage the SAML2 profile "ECP" in conjunction with the "PAOS Binding."

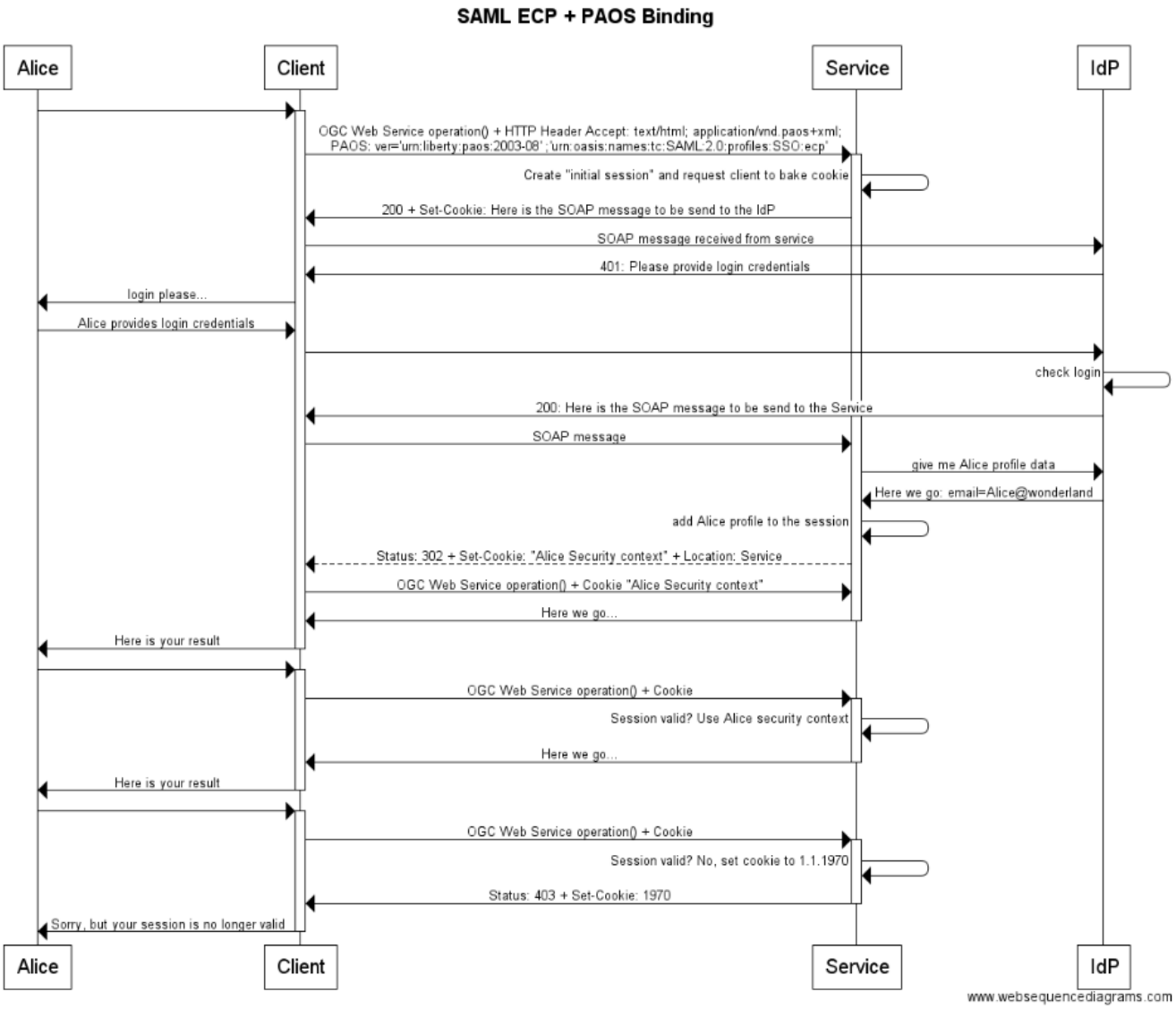

Figure 5. SAML ECP + PAOS Binding

- The client submits ACCEPT header: SAML ECP specific value "Accept: text/html; application/vnd.paos+xml; \nPAOS: ver='urn:liberty:paos:2003-08' ;'urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp'" indicates to service that the client is capable of the SAML2 ECP handshake.
- The service responses with a SOAP message + Set-Cookie.
- Client must submit HTTP cookie with requests.

```
title SAML ECP + PAOS Binding

Alice->+Client:
Client->+Service: OGC Web Service operation() + HTTP Header Accept: text/html;
application/vnd.paos+xml; \nPAOS: ver='urn:liberty:paos:2003-08'
;'urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp'
Service->Service: Create "initial session" and request client to bake cookie
Service->Client: 200 + Set-Cookie: Here is the SOAP message to be send to the IdP
Client->IdP: SOAP message received from service
IdP->Client: 401: Please provide login credentials
Client->Alice: login please...
Alice->Client: Alice provides login credentials
Client->IdP:
IdP->IdP: check login
IdP->Client: 200: Here is the SOAP message to be send to the Service
Client->Service: SOAP message
Service->IdP: give me Alice profile data
IdP->Service: Here we go: email=Alice@wonderland
Service->Service: add Alice profile to the session
Service-->Client: Status: 302 + Set-Cookie: "Alice Security context" + Location:
Service
Client->Service: OGC Web Service operation() + Cookie "Alice Security context"
Service->-Client: Here we go...
Client->-Alice: Here is your result

Alice->+Client:
Client->+Service: OGC Web Service operation() + Cookie
Service->Service: Session valid? Use Alice security context
Service->-Client: Here we go...
Client->-Alice: Here is your result

Alice->+Client:
Client->+Service: OGC Web Service operation() + Cookie
Service->Service: Session valid? No, set cookie to 1.1.1970
Service->-Client: Status: 403 + Set-Cookie: 1970
Client->-Alice: Sorry, but your session is no longer valid
```

6.3.5. Scenario "SOAP with WS-Security enabled Authentication-Based Systems"

The following example illustrates the use of security on the application layer by applying "WS-Security" to SOAP <Header> elements. The SOAP security enabled endpoint acts as a reverse proxy to an unsecured WFS.

- 1) The security facade inspects the received SOAP message to be compliant with the WS-Policy in place for the service.
- 2) For example, the validity of the digital signature for the Digest Username/Password token included in the header.

- 3) The security facade strips off the SOAP message parts and forwards the content of the <Body> element to the unprotected service.
- 4) The response from the WFS is included inside a SOAP envelope as the content of the <Body>.
- 5) Outbound message security is applied as specified in the WS-Policy.

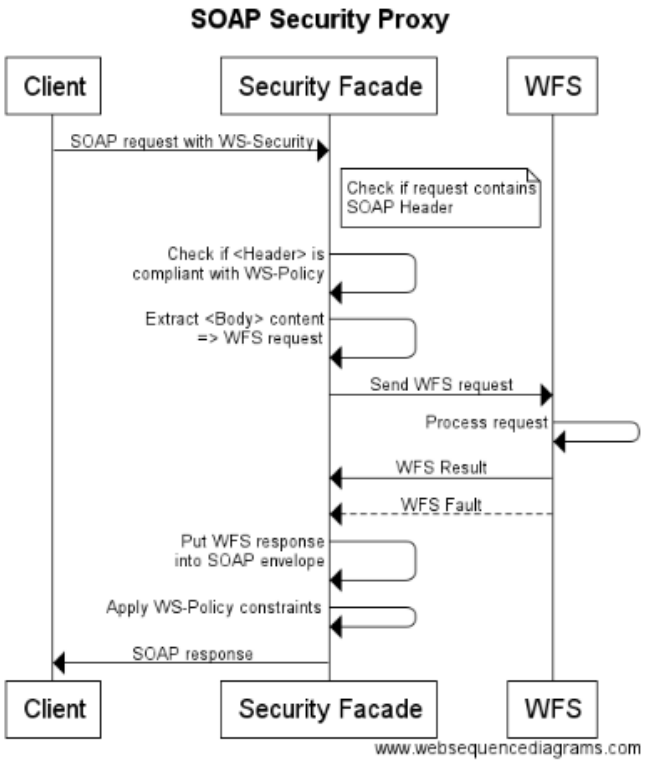

Figure 6. SOAP + Security

```
title SOAP Security Proxy

Client->Security Facade: SOAP request with WS-Security
note right of Security Facade: Check if request contains\nSOAP Header
Security Facade->Security Facade: Check if <Header> is\ncompliant with WS-Policy
Security Facade->Security Facade: Extract <Body> content\n=> WFS request
Security Facade->WFS: Send WFS request
WFS->WFS: Process request
WFS->Security Facade: WFS Result
WFS-->Security Facade: WFS Fault
Security Facade->Security Facade: Put WFS response\ninto SOAP envelope
Security Facade->Security Facade: Apply WS-Policy constraints
Security Facade->Client: SOAP response
```

Chapter 7. Advertisement of security at the service side

NOTE	*Client must understand the security model* Interoperability to secured services requires that the calling application, aka the client, understands which security mechanism(s) is(are) implemented at the service and is able to technically "deal" with the advertised requirements. The vehicle to ship the information from the service to the client is capabilities with security annotations.

7.1. How can the service tell the client which security is "setup"?

As described in OGC 15-022, basically two different sets of metadata exist, as below.

- OGC Capabilities: This XML document is used by the client application(s) to bind to an OGC Web Service. It is therefore the appropriate place to include security annotations in machine-readable format that describe the security model implemented at the service instance.
- ISO Service Metadata: This XML document is retrieved from a Catalog and supports end users in searching for applicable services but also developers to craft workflows. An ISO Service Metadata instance document is typically not suitable to actually bind to an OGC Web Services. It should outline procedures that users can follow to obtain required certificates, token, etc. to execute a protected service endpoint.

Different types of OGC Web Services provide different patterns how to receive a capabilities document.

- OWS: The GetCapabilities() operation is common among all services.
- SOAP OWS: For a service which has SOAP security (WS-Security) instrumented, the capabilities document must not be used to provide the security metadata other than pointing (referencing) to a WS-Policy. The GetWSDL operation (if available) must be used to obtain a WSDL document that defined the SOAP message structure and uses WS-Policy to outline the regulations for the SOAP message including crypto and cipher restrictions.

For the use of annotated capabilities to express the security conditions on the service endpoint, it is important that such a capabilities document does not crash existing clients that do not know about the annotation. This is important as rolling-out the security annotations will take place in an existing environment and the ability to update services one by one and use existing clients and new clients that understand the annotated security marking in the capabilities is essential for this concept to have a chance of getting used. This backwards compatibility can be achieved when the security annotation in the capabilities file does not invalidate the structure of the capabilities document. In other words, for each version of a W*S capabilities document, it must be possible to define the security conditions. The proposal, introduced in OGC 15-022, is to leverage an existing element from the OWS common (any version) schema: ows:Constraint. By doing so, the structure of the annotated capabilities document is still valid against the schema.

NOTE

WMS 1.1.1 and 1.3.0 require an individual solution

Because the Web Map Service does not structure the Capabilities document using the OWS Common schema, all versions of WMS require an individual solution. It is important to note that the WMS 1.1.1 structures the capabilities based on a DTD and WMS 1.3.0 uses its own schema.

The following figure illustrates the use of the annotated capabilities document.

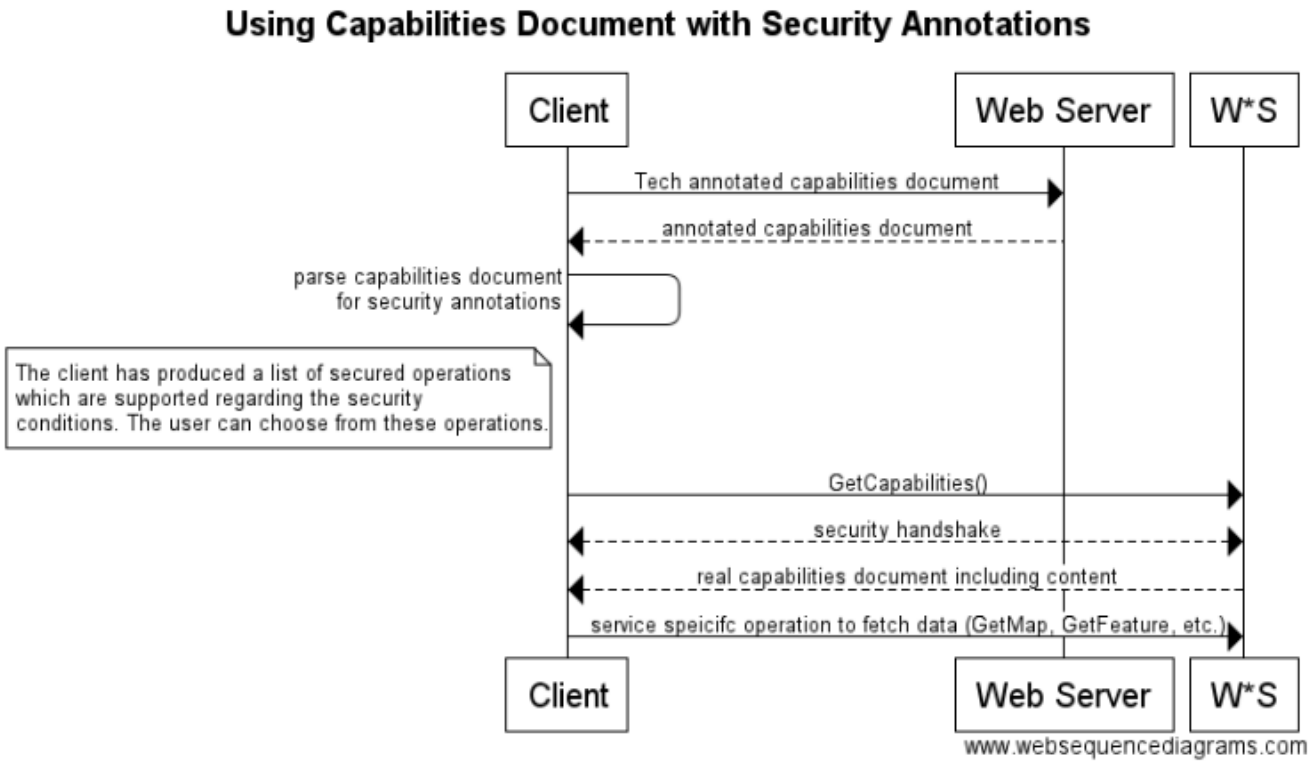

Figure 7. Use of annotated Capabilities document

```
title Using Capabilities Document with Security Annotations
Client->Web Server: Tech annotated capabilities document
Web Server-->Client: annotated capabilities document
Client->Client: parse capabilities document\nfor security annotations
note left of Client: The client has produced a list of secured operations\nwhich are
supported regarding the security\nconditions. The user can choose from these
operations.
Client->W*S: GetCapabilities()
Client<-->W*S: security handshake
W*S-->Client: real capabilities document including content
Client->W*S: service specifc operation to fetch data (GetMap, GetFeature, etc.)
```

7.2. Security annotation in the Capabilities instance document

The approach to annotate the Capabilities document with security information is explained in detail in OGC 15-022. Basically, the approach can be summarized as follows:

- For all W*S, where the capabilities structure is based on OWS Common schema, the ows:Constraint element shall be used to express security constraints;
- For a WMS 1.3, the capabilities document structure is based on O WMS-specific schema, similar to the schema from OWS Common but the ows:Constraint element is missing; and
- For WMS 1.1.1, the capabilities are structured using a DTD.

NOTE

WMS must use ExtendedCapabilities to enable security annotations

For WMS, is necessary to define a mechanism that enables a similar annotation method as provided from OWS Common. For a WMS 1.3.0, the option is to define a specialization of the _ExtendedCapabilitiesElement. For a WMS 1.1.1, the capabilities is structured by a DTD which requires to define a proper extension mechanism via the VendorSpecificCapabilities element.

7.2.1. OWS Common Version 1.0

For OWS Common 1.0.0 schema, it is only possible to list the ows:Value element inside the constraint element. This affects the CSW and the WFS 1.1.0 as these version use the OWS Common 1.0 schema.

The following figure illustrates this limitation

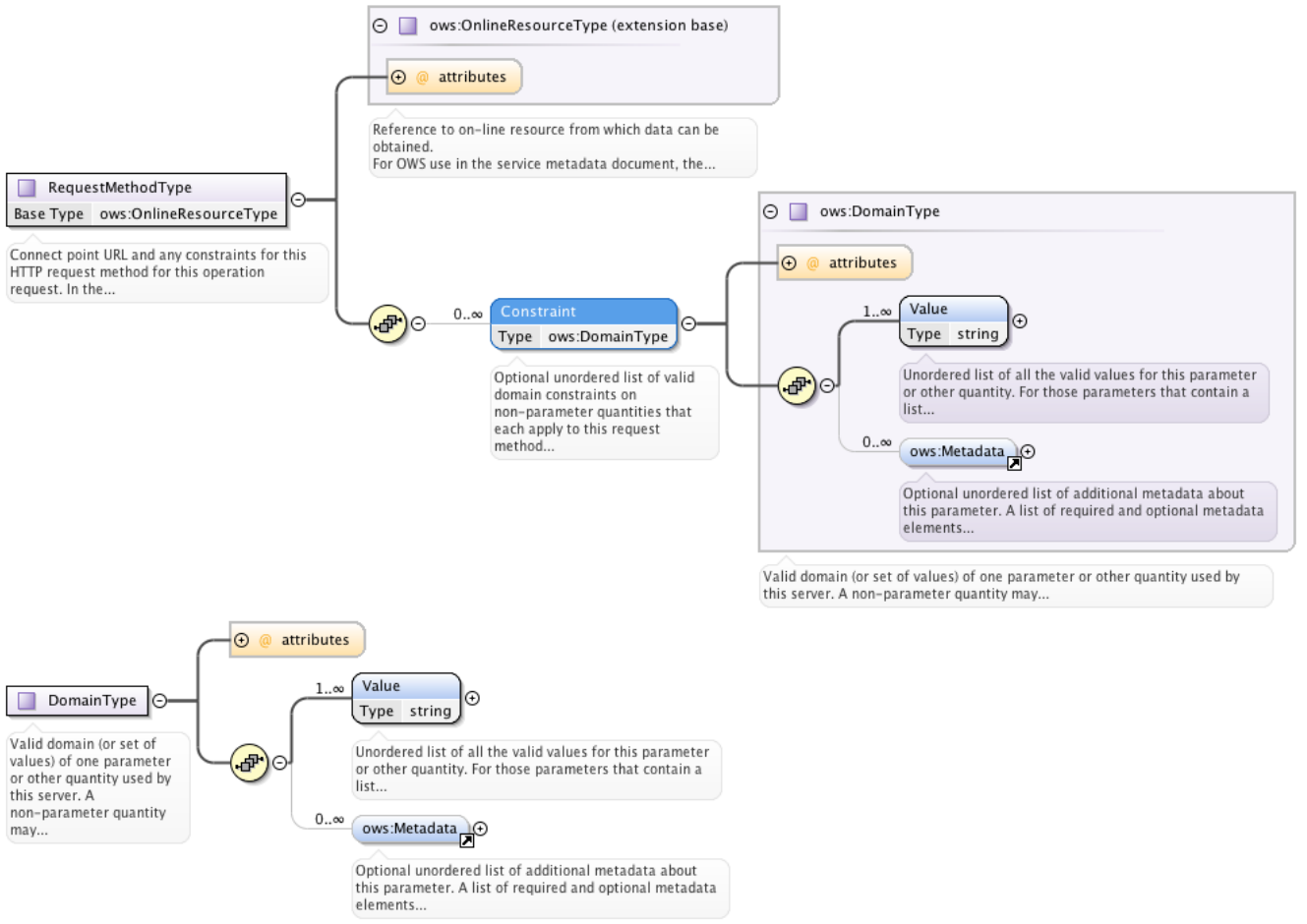

Figure 8. Definition of the Constraint element (DomainType) in OWS Common 1.0

7.2.2. OWS Common Version > 1.0

The options to use the ows:Constraint element(s) is determined by the OWS Common XSD. The ows:OperationsMetadata element from the owsOperationsMetadata-<version>.xsd specifies the Constraint element:

```
<element name="OperationsMetadata">
  <annotation>
    <documentation>Metadata about the operations and related abilities
    specified by this service and implemented by this server, including the
    URLs for operation requests. The basic contents of this section shall be
    the same for all OWS types, but individual services can add elements
    and/or change the optionality of optional elements.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element ref="ows:Operation"
               minOccurs="2"
               maxOccurs="unbounded">
        <annotation>
          <documentation>Metadata for unordered list of all the (requests
          for) operations that this server interface implements. The list of
          required and optional operations implemented shall be specified in
          the Implementation Specification for this service.</documentation>
        </annotation>
      </element>
      <element name="Parameter"
               type="ows:DomainType"
               minOccurs="0"
               maxOccurs="unbounded">
        <annotation>
          <documentation>Optional unordered list of parameter valid domains
          that each apply to one or more operations which this server
          interface implements. The list of required and optional parameter
          domain limitations shall be specified in the Implementation
          Specification for this service.</documentation>
        </annotation>
      </element>
      <element name="Constraint"
               type="ows:DomainType"
               minOccurs="0"
               maxOccurs="unbounded">
        <annotation>
          <documentation>Optional unordered list of valid domain constraints
          on non-parameter quantities that each apply to this server. The
          list of required and optional constraints shall be specified in
          the Implementation Specification for this service.</documentation>
        </annotation>
      </element>
      <element ref="ows:ExtendedCapabilities"
               minOccurs="0" />
    </sequence>
  </complexType>
</element>
```

It is important to point out that the element ows:Constraint is optional but may occur **unbounded**.

The use of multiple ows:Constraint elements within the same OperationsMetadata element shall be interpreted as a logical AND.

In case an operation has a choice of multiple security constraints (OR) the ows:Get or ows:Post element shall be duplicated to reflect the logical OR. The following schema snippet illustrates this rational:

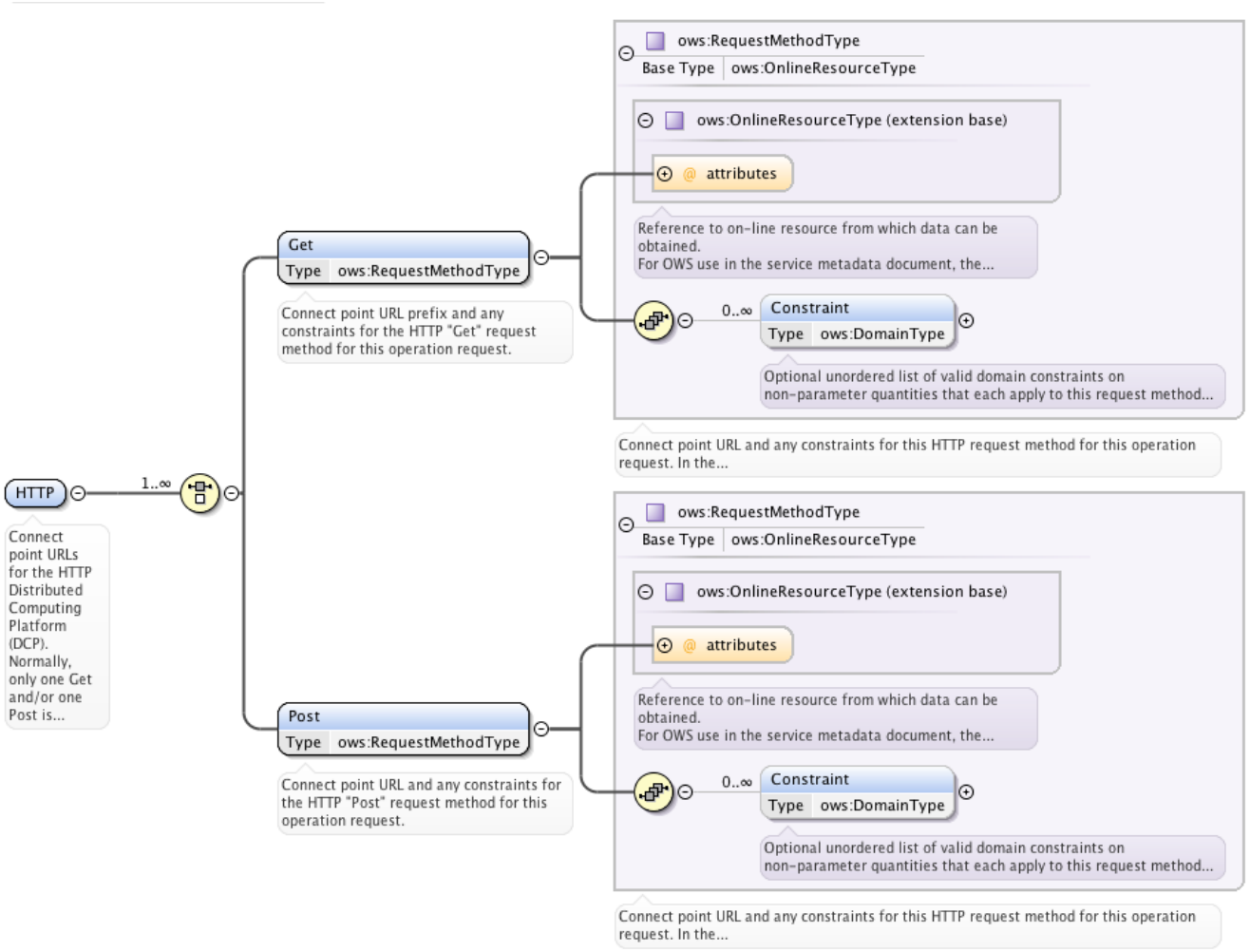

*Figure 9. Definition of the HTTP element in W*S Capabilities*

This procedure, to normalize the logical OR into a DNF of (Get | Post)+ elements is required as we are not allowed to define a new structure because of backwards-compatibility.

An annotation with security shall be done per service operation using the ows:ValuesReference element. The ows:Reference attribute shall point to the authentication codelist, the authorization policy or the WS-Policy.

7.2.3. Different types of security constraints

As outlined in OGC 15-022, it is important to provide information on security constraints like authentication, authorization, integrity and confidentiality. Within TB12, the following URNs have been defined with the following meaning.

- urn:ogc:def:tb12:security:authentication refers to an OGC maintained code list for well defined authentication schemes. For TB12, the following authentication code list is established: http://tb12.opengis.net/security/authCodeList This code list defines some of the most frequently

used authentication schemes like Basic Authentication, Client Certificate, SAML2 Browser POST and ECP. The following example illustrates the definition of an authentication condition:

```
<ows:Constraint name="urn:ogc:def:tb12:security:authentication">
  <ows:ValuesReference
ows:reference="http://tb12.opengis.net/security/authCodeList#CLIENT_CERTIFICATE”>urn:o
gc:def:tb12:ietf:5246:client_certificate</ows:ValuesReference>
</ows:Constraint>
```

- urn:ogc:def:tb12:security:authorization refers to a specification of authorisation rules using a XACML or GeoXACML policy. The following example illustrates the use of this option:

```
<ows:Constraint name="urn:ogc:def:tb12:security:authorization">
  <ows:ValuesReference ows:reference="https://tb12.ogc.secure-
dimensions.com/TB12.geoxacml.xml"/>
</ows:Constraint>
```

- urn:ogc:def:tb12:security:policy refers to a WS-Policy that defines the security constraints for a SOAP envelope of a SOAP security enabled service endpoint.

```
<ows:Constraint name="urn:ogc:def:tb12:security:policy">
  <ows:ValuesReference ows:reference="https://tb12.secure-
dimensions.com/WFSSecurityPolicy.xml"/>
</ows:Constraint>
```

- urn:ogc:def:tb12:security:layer4 refers to SSL or TLS encrypted communication channels used for HTTPS (HTTP over TLS)

```
<ows:Constraint name="urn:ogc:def:tb12:security:layer4”>
  <ows:AllowedValues>
    <ows:Value>urn:ietf:tls:1.2</ows:Value>
  </ows:AllowedValues>
</ows:Constraint>
```

7.2.4. Annotating multiple security constraints for OWS Common based capabilities

From OWS Common XSD, the ows:ServiceMetadata element contains the ows:Constraint element, which can occur unbounded. It is therefore possible to use one or multiple ows:Constraint element(s) to express one or multiple security constraints per service operation. It is important to note that the logical AND must be applied for all ows:Constraint elements.

In case that two or more security constraints exist as a choice (for example BASIC or DIGEST Authentication) it is required to duplicate the ows:Get or ows:Post element accordingly. As an example, the following XML fragment illustrates the choice of BASIC or DIGEST authentication:

```
<ows:Operation name="GetCapabilities">
  <ows:DCP>
    <ows:HTTP>
      <ows:Get xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="https://domain.com/service/wms">
        <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
          <ows:ValuesReference
ows:reference="http://tb12.opengis.net/security/authCodeList#HTTP_BASIC”>urn:ogc:def:t
b12:ietf:2617:basic</ows:ValuesReference>
        </ows:Constraint>
      </ows:Get>
      <ows:Get xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="https://domain.com/service/wms">
        <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
          <ows:ValuesReference
ows:reference="http://tb12.opengis.net/security/authCodeList#HTTP_DIGEST”>urn:ogc:def:
tb12:ietf:2617:digest</ows:ValuesReference>
        </ows:Constraint>
      </ows:Get>
    </ows:HTTP>
  </ows:DCP>
</ows:Operation>
```

7.3. Maintaining the authentication code list

The authentication code list, referenced from the security identifier for authentication (urn:ogc:def:tb12:security:authentication), shall be maintained by the OGC OWS Common - Security SWG. In order to ensure backwards compatibility, the code list must never change for existing items and no items can ever be removed. In other words, the SWG can only add identifiers. The process is that any interested party can issue a CR to the SWG for adding a new identifier. Once the SWG has accepted the request (and clarified the identifier and the text), the OGC naming authority must approve the identifier URN. It will then be populated by OGC staff to become available online. The SWG must also decide the URL via which the normative OGC authentication code list becomes available.

The structure of the OGC authentication code list shall be a ISO format based on the gmx namespace: http://www.isotc211.org/2005/gmx. The namespace and the definition can be retrieved from here: https://cdn.earthdata.nasa.gov/iso/gmx/1.0/. This enables the seamless use of the authentication code identifiers in ISO metadata.

Example of a code list entry that defines the reference from above (HTTP Basic Authentication):

```
  <gmx:codelistItem>
    <gmx:CodeListDictionary gml:id="AuthenticationCode">
      <gml:description>identification of authentication methods</gml:description>
      <gml:identifier
codeSpace="OGC">urn:def:ogc:tb12:security:authentication</gml:identifier>
      <gmx:codeEntry>
        <gmx:CodeDefinition gml:id="HTTP_BASIC">
          <gml:description>
          The "basic" authentication scheme is based on the model that the
          client must authenticate itself with a user-ID and a password for
          each realm.  The realm value should be considered an opaque string
          which can only be compared for equality with other realms on that
          server. The server will service the request only if it can validate
          the user-ID and password for the protection space of the Request-URI.
          There are no optional authentication parameters.</gml:description>
          <gml:identifier codeSpace="OGC
">urn:ogc:def:tb12:ietf:2617:basic</gml:identifier>
        </gmx:CodeDefinition>
      </gmx:codeEntry>
    </gmx:CodeListDictionary>
  </gmx:codelistItem>
```

It is important to note that the GMX code list structure supports multiple code spaces. It is anticipated to use existing identifiers, like OASIS defines URNs for SAML. The namespace OGC shall be used to create unique identifiers for all authentication schemes that do not have a unique urn identifier from another standardization body. In that sense, OGC defines a URN for the HTTP Authentication methods as defined in IETF RFC 2617.

NOTE

More annotation examples

More examples of capabilities including security annotations are available in a later chapter.

7.3.1. Capabilities document must be publicly accessible

In case that a service endpoint has security implemented, the client can only use the Capabilities document and its security annotations to determine compatibility if the instance document could be obtained either (i) publicly or (ii) via a common security mechanism implemented by all clients on the planet. In that sense, chances are that the annotated version of the Capabilities for a protected service must be publicly available. In this case, the client can read and compile the security annotations per operation and filter supported security settings.

But, this approach has an implication if the content served by the protected service is sensitive or information about the data served shall not be given to anonymous users. In this case, the content section of the publicly accessible annotated Capabilities instance document could either:

- contain an empty content section: the exact structure varies for different service types, but it is possible; or

use URL references to the content definition: of course, these links would be protected so that for obtaining a list of offerings, the user client must overcome the security constraint.

In any case, the client must always fetch the full capabilities document via the GetCapabilities operation, outlined in the publicly available capabilities document comprising of the security annotations.

WFS Schema supporting empty content

The following Capabilities document is an example for a WFS 1.1.0 capabilities document that does not contain any FeatureType information; just the Operations Metadata. This approach can be used in case the WFS is serving feature types that are not intended for public awareness.

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:WFS_Capabilities version="1.1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xmlns="http://www.opengis.net/wfs" xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml"
  xmlns:ogc="http://www.opengis.net/ogc" xmlns:xlink="http://www.w3.org/1999/xlink"
  xsi:schemaLocation="http://www.opengis.net/wfs
http://schemas.opengis.net/wfs/1.1.0/wfs.xsd"
  xmlns:xml="http://www.w3.org/XML/1998/namespace"
xmlns:DigitalGlobe="http://www.digitalglobe.com"
  updateSequence="106">
  <ows:ServiceIdentification>
    <ows:Title>DigitalGlobe Web Feature Service</ows:Title>
    <ows:Abstract/>
    <ows:Keywords>
      <ows:Keyword>WFS</ows:Keyword>
      <ows:Keyword>WMS</ows:Keyword>
    </ows:Keywords>
    <ows:ServiceType>WFS</ows:ServiceType>
    <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
    <ows:Fees>NONE</ows:Fees>
    <ows:AccessConstraints>NONE</ows:AccessConstraints>
  </ows:ServiceIdentification>
  <ows:ServiceProvider>
    <ows:ProviderName>DigitalGlobe Inc</ows:ProviderName>
    <ows:ServiceContact>
      <ows:IndividualName>Customer Service Department</ows:IndividualName>
      <ows:PositionName>Customer Service Department</ows:PositionName>
      <ows:ContactInfo>
        <ows:Phone>
          <ows:Voice>800.496.1225</ows:Voice>
          <ows:Facsimile>303.684.4562</ows:Facsimile>
        </ows:Phone>
        <ows:Address>
          <ows:City/>
          <ows:AdministrativeArea/>
          <ows:PostalCode/>
```

```
          <ows:Country/>
        </ows:Address>
      </ows:ContactInfo>
    </ows:ServiceContact>
  </ows:ServiceProvider>
  <ows:OperationsMetadata>
    <ows:Operation name="GetCapabilities">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get
            xlink:href="https://https://tb12.ogc.secure-dimensions.com/service/wfs?">
            <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
              <ows:Value>urn:ogc:def:tb12:ietf:5246:client_certificate</ows:Value>
            </ows:Constraint>
          </ows:Get>
          <ows:Post
            xlink:href="https://https://tb12.ogc.secure-dimensions.com/service/wfs">
            <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
              <ows:Value>urn:ogc:def:tb12:ietf:5246:client_certificate</ows:Value>
            </ows:Constraint>
          </ows:Post>
        </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="AcceptVersions">
        <ows:Value>1.0.0</ows:Value>
        <ows:Value>1.1.0</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="AcceptFormats">
        <ows:Value>text/xml</ows:Value>
      </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="DescribeFeatureType">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get
            xlink:href="https://https://tb12.ogc.secure-dimensions.com/service/wfs?">
            <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
              <ows:Value>urn:ogc:def:tb12:ietf:5246:client_certificate</ows:Value>
            </ows:Constraint>
          </ows:Get>
          <ows:Post
            xlink:href="https://https://tb12.ogc.secure-dimensions.com/service/wfs">
            <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
              <ows:Value>urn:ogc:def:tb12:ietf:5246:client_certificate</ows:Value>
            </ows:Constraint>
          </ows:Post>
        </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="outputFormat">
        <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
      </ows:Parameter>
```

```
    </ows:Operation>
    <ows:Operation name="GetFeature">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get
            xlink:href="https://https://tb12.ogc.secure-dimensions.com/service/wfs?">
            <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
              <ows:Value>urn:ogc:def:tb12:ietf:5246:client_certificate</ows:Value>
            </ows:Constraint>
          </ows:Get>
          <ows:Post
            xlink:href="https://https://tb12.ogc.secure-dimensions.com/service/wfs">
            <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
              <ows:Value>urn:ogc:def:tb12:ietf:5246:client_certificate</ows:Value>
            </ows:Constraint>
          </ows:Post>
        </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="resultType">
        <ows:Value>results</ows:Value>
        <ows:Value>hits</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="outputFormat">
        <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
        <ows:Value>GML2</ows:Value>
        <ows:Value>KML</ows:Value>
        <ows:Value>SHAPE-ZIP</ows:Value>
        <ows:Value>application/atom xml</ows:Value>
        <ows:Value>application/atom+xml</ows:Value>
        <ows:Value>application/gml+xml; version=3.2</ows:Value>
        <ows:Value>application/json</ows:Value>
        <ows:Value>application/rss xml</ows:Value>
        <ows:Value>application/rss+xml</ows:Value>
        <ows:Value>application/vnd.google-earth.kml</ows:Value>
        <ows:Value>atom</ows:Value>
        <ows:Value>csv</ows:Value>
        <ows:Value>gml3</ows:Value>
        <ows:Value>gml32</ows:Value>
        <ows:Value>json</ows:Value>
        <ows:Value>rss</ows:Value>
        <ows:Value>text/xml; subtype=gml/2.1.2</ows:Value>
        <ows:Value>text/xml; subtype=gml/3.2</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="featureProfile">
        <ows:Value>Color_Infrared_Profile</ows:Value>
        <ows:Value>Legacy_Profile</ows:Value>
        <ows:Value>Cloud_Cover_Profile</ows:Value>
        <ows:Value>Accuracy_Profile</ows:Value>
        <ows:Value>Currency_Profile</ows:Value>
        <ows:Value>Currency_RGB_Profile</ows:Value>
        <ows:Value>True_Currency_Profile</ows:Value>
```

```
        <ows:Value>Global_Currency_Profile</ows:Value>
        <ows:Value>Consumer_Profile</ows:Value>
        <ows:Value>Classic_Color_Consumer_Profile</ows:Value>
        <ows:Value>Color_Consumer_Profile</ows:Value>
        <ows:Value>MyDG_Consumer_Profile</ows:Value>
        <ows:Value>MyDG_Color_Consumer_Profile</ows:Value>
        <ows:Value>Default_Profile</ows:Value>
      </ows:Parameter>
      <ows:Constraint name="LocalTraverseXLinkScope">
        <ows:Value>2</ows:Value>
      </ows:Constraint>
    </ows:Operation>
  </ows:OperationsMetadata>
  <ogc:Filter_Capabilities>
    <ogc:Spatial_Capabilities>
      <ogc:GeometryOperands>
        <ogc:GeometryOperand>gml:Polygon</ogc:GeometryOperand>
      </ogc:GeometryOperands>
      <ogc:SpatialOperators>
        <ogc:SpatialOperator name="Intersects"/>
      </ogc:SpatialOperators>
    </ogc:Spatial_Capabilities>
    <ogc:Scalar_Capabilities>
      <ogc:LogicalOperators/>
      <ogc:ComparisonOperators>
        <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>Between</ogc:ComparisonOperator>
      </ogc:ComparisonOperators>
      <ogc:ArithmeticOperators>
        <ogc:SimpleArithmetic/>
      </ogc:ArithmeticOperators>
    </ogc:Scalar_Capabilities>
    <ogc:Id_Capabilities>
      <ogc:FID/>
      <ogc:EID/>
    </ogc:Id_Capabilities>
  </ogc:Filter_Capabilities>
</wfs:WFS_Capabilities>
```

WMS Schema supporting empty content

The following Capabilities document is an example for a WMS 1.3 capabilities document that does not contain any Layer information; just the Operations Metadata. This approach can be used in case the WMS is serving layers that are not intended for public awareness.

```
<?xml version="1.0" encoding="UTF-8"?>
<WMS_Capabilities xmlns:DigitalGlobe="http://www.digitalglobe.com" version="1.3.0"
  updateSequence="5511" xmlns="http://www.opengis.net/wms"
  xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
  xmlns:ows="http://www.opengis.net/ows/1.1"
  xmlns:ows_security="http://wwww.opengis.net/security/1.0"
  xsi:schemaLocation="http://www.opengis.net/wms
http://schemas.opengis.net/wms/1.3.0/capabilities_1_3_0.xsd
  http://www.digitalglobe.com https://tb12.ogc.secure-
dimensions.com/capabilities/dg_wms.xsd
  http://wwww.opengis.net/security/1.0 https://tb12.ogc.secure-
dimensions.com/capabilities/ExtendedSecurityCapabilities.xsd">
  <Service>
    <Name>WMS</Name>
    <Title>DigitalGlobe Web Map Service</Title>
    <Abstract/>
    <KeywordList>
      <Keyword>WFS</Keyword>
      <Keyword>WMS</Keyword>
    </KeywordList>
    <OnlineResource xlink:type="simple" xlink:href="http://www.digitalglobe.com"/>
    <ContactInformation>
      <ContactPersonPrimary>
        <ContactPerson>Customer Service Department</ContactPerson>
        <ContactOrganization>DigitalGlobe Inc</ContactOrganization>
      </ContactPersonPrimary>
      <ContactPosition>Customer Service Department</ContactPosition>
      <ContactAddress>
        <AddressType>Work</AddressType>
        <Address/>
        <City/>
        <StateOrProvince/>
        <PostCode/>
        <Country/>
      </ContactAddress>
      <ContactVoiceTelephone>800.496.1225</ContactVoiceTelephone>
      <ContactFacsimileTelephone>303.684.4562</ContactFacsimileTelephone>

<ContactElectronicMailAddress>info@digitalglobe.com</ContactElectronicMailAddress>
    </ContactInformation>
    <Fees>NONE</Fees>
    <AccessConstraints>NONE</AccessConstraints>
  </Service>
  <Capability>
    <Request>
      <GetCapabilities>
        <Format>text/xml</Format>
        <DCPType>
          <HTTP>
            <Get>
              <OnlineResource xlink:type="simple"
                xlink:href="https://tb12.ogc.secure-dimensions.com/service/wms"
              />
            </Get>
            <Post>
```

```
            <OnlineResource xlink:type="simple"
              xlink:href="https://tb12.ogc.secure-dimensions.com/service/wms"
            />
          </Post>
        </HTTP>
      </DCPType>
    </GetCapabilities>
    <GetMap>
      <Format>image/png</Format>
      <Format>image/geotiff</Format>
      <Format>image/jpeg</Format>
      <DCPType>
        <HTTP>
          <Get>
            <OnlineResource xlink:type="simple"
              xlink:href="https://tb12.ogc.secure-dimensions.com/service/wms"
            />
          </Get>
        </HTTP>
      </DCPType>
    </GetMap>
    <GetFeatureInfo>
      <Format>text/plain</Format>
      <Format>application/vnd.ogc.gml</Format>
      <Format>application/vnd.ogc.gml/3.1.1</Format>
      <Format>text/html</Format>
      <Format>application/json</Format>
      <DCPType>
        <HTTP>
          <Get>
            <OnlineResource xlink:type="simple"
              xlink:href="https://tb12.ogc.secure-dimensions.com/service/wms"
            />
          </Get>
        </HTTP>
      </DCPType>
    </GetFeatureInfo>
  </Request>
  <Exception>
    <Format>XML</Format>
    <Format>INIMAGE</Format>
    <Format>BLANK</Format>
  </Exception>
  <DigitalGlobe:ExtendedCapabilities>
    <DigitalGlobe:Profiles>
      <DigitalGlobe:Profile>Color_Infrared_Profile</DigitalGlobe:Profile>
      <DigitalGlobe:Profile>Legacy_Profile</DigitalGlobe:Profile>
      <DigitalGlobe:Profile>Cloud_Cover_Profile</DigitalGlobe:Profile>
      <DigitalGlobe:Profile>Accuracy_Profile</DigitalGlobe:Profile>
      <DigitalGlobe:Profile>Currency_Profile</DigitalGlobe:Profile>
      <DigitalGlobe:Profile>Currency_RGB_Profile</DigitalGlobe:Profile>
```

```
        <DigitalGlobe:Profile>True_Currency_Profile</DigitalGlobe:Profile>
        <DigitalGlobe:Profile>Global_Currency_Profile</DigitalGlobe:Profile>
        <DigitalGlobe:Profile>Consumer_Profile</DigitalGlobe:Profile>
        <DigitalGlobe:Profile>Classic_Color_Consumer_Profile</DigitalGlobe:Profile>
        <DigitalGlobe:Profile>Color_Consumer_Profile</DigitalGlobe:Profile>
        <DigitalGlobe:Profile>MyDG_Consumer_Profile</DigitalGlobe:Profile>
        <DigitalGlobe:Profile>MyDG_Color_Consumer_Profile</DigitalGlobe:Profile>
        <DigitalGlobe:Profile>Default_Profile</DigitalGlobe:Profile>
      </DigitalGlobe:Profiles>

<DigitalGlobe:FeatureCollection>featureCollection</DigitalGlobe:FeatureCollection>
    </DigitalGlobe:ExtendedCapabilities>
    <ows_security:ExtendedSecurityCapabilities>
      <ows:OperationsMetadata>
        <ows:Operation name="GetCapabilities">
          <ows:DCP>
            <ows:HTTP>
              <ows:Get xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="https://tb12.ogc.secure-dimensions.com/service/wms">
                <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
                  <ows:ValuesReference
ows:reference="http://tb12.opengis.net/security/authCodeList#CLIENT_CERTIFICATE">urn:o
gc:def:tb12:ietf:5246:client_certificate</ows:ValuesReference>
                </ows:Constraint>
              </ows:Get>
            </ows:HTTP>
          </ows:DCP>
        </ows:Operation>
        <ows:Operation name="GetMap">
          <ows:DCP>
            <ows:HTTP>
              <ows:Get xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="https://tb12.ogc.secure-dimensions.com/service/wms">
                <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
                  <ows:ValuesReference
ows:reference="http://tb12.opengis.net/security/authCodeList#CLIENT_CERTIFICATE">urn:o
gc:def:tb12:ietf:5246:client_certificate</ows:ValuesReference>
                </ows:Constraint>
                <ows:Constraint name="urn:ogc:def:tb12:security:authorization">
                  <ows:ValuesReference ows:reference="https://tb12.ogc.secure-
dimensions.com/TB12.geoxacml.xml"/>
                </ows:Constraint>
              </ows:Get>
            </ows:HTTP>
          </ows:DCP>
        </ows:Operation>
      </ows:OperationsMetadata>
    </ows_security:ExtendedSecurityCapabilities>
  </Capability>
</WMS_Capabilities>
```

Chapter 8. Authentication Code List

NOTE	*An Authentication Code List ensures interoperability* A well defined list of authentication methods, defined and maintained by the OGC Naming Authority, guarantees that interoperability can be implemented between a service implementing the method and a client challenging the method.

8.1. Authentication Code List and Conformance Classes

The use of well defined authentication methods, collected in a controlled list, provides the benefit that interested parties can write OGC extensions or profiles that collect relevant methods together for a particular installation.

As an example, a commercial provider likes to ensure that in his Federation, all Web-based and desktop applications are SAML enabled. A profile of the OWS Common Security Extension would reference the authentication methods SAML ECP as mandatory to be implemented. In another community, perhaps the mobile two factor authentication is relevant and therefore that community profile would mandate other entries from the code list.

8.2. Authentication Methods standardized and common practice in mainstream IT

Some standardization organizations have established URNs or other unique identifiers for authentication methods. It is the idea to use all existing authentication identifiers established by other standardization bodies. OGC must only create identifiers for those authentication methods that do no have a unique identifier already.

It is the idea to base the authentication registry on the ISO GMX structure, which enables the use of code spaces. This enables to refer to authentication method definitions from other standardization bodies and separate the OGC code space to create relevant URNs via the OGC Naming Authority.

8.2.1. OASIS

OASIS has well established URNs for their authentication methods associated with SAML. It is proposed to include those URNs using the code space “OASIS.”

- SSO Browser Profile: urn:oasis:names:tc:SAML:2.0:profiles:SSO:browser
- ECP Profile: urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp

8.2.2. IETF

The IETF seems not to have established unique identifiers for the authentication methods defined in RFC 2617: HTTP Basic and Digest Authentication. Also, RFC 5246 does not define an identifier for client side certificate.

It is therefore required that the OGC Authentication Code List include identifiers for these IETF

authentication methods. For TB12, the following URNs were established:

- BASIC: urn:ogc:def:tb12:ietf:2617:basic
- DIGEST: urn:ogc:def:tb12:ietf:2617:digest
- TLS with client side certificate: urn:ogc:def:tb12:ietf:5246:client_certificate

8.3. TB12 Authentication Code List

For TB12, the OGC hosts a code list under this URL: http://tb12.opengis.net/security/authCodeList. It enables the lookup of individual definitions by using the "?" with the identifier. An example URL to lookup the definition of TLS with client side certificate is this: http://tb12.opengis.net/security/authCodeList?CLIENT_CERTIFICATE

NOTE

Authentication Code URNs

The Authentication Codes used in the Authentication Code List are based on URNs that are **not** approved by the OGC Naming Authority! For standardization, the OGC Naming Authority must approve the Authentication codes and the security identifier URNs as well as the official URL for the Authentication Code List. It is expected that the OWS Common Security SWG will request the URNs from the Naming Authority officially.

In order to facilitate the different versions of OWS Common (1.0.0 does only allow string values were as 1.1.0 and 2.0 allow ValuesReference) it is possible to just provide the urn value or a resolvable link.

The following is an example of the TB12 code list. Please note that this list is incomplete for standardization, but it reflects the authentication methods used in TB12 and it supports test of the security annotation approach:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="ogc_gml.xsl"?>
<gmx:CT_CodelistCatalogue xmlns="http://www.secure-dimensions.eu/auth"
  xmlns:gmx="http://www.isotc211.org/2005/gmx"
xmlns:gco="http://www.isotc211.org/2005/gco"
  xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
  http://www.isotc211.org/2005/gmx
http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/gmx/gmx.xsd
  http://www.isotc211.org/2005/gco
http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/gco/gco.xsd
  http://www.opengis.net/gml/3.2
http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19136_Schemas/gml.xsd">
  <!--======Catalogue description=======-->
  <gmx:name>
    <gco:CharacterString>authCodelists</gco:CharacterString>
  </gmx:name>
```

```
  <gmx:scope>
    <gco:CharacterString>Codelists for description of authentication
TB12</gco:CharacterString>
  </gmx:scope>
  <gmx:fieldOfApplication>
    <gco:CharacterString>OGC Testbed 12</gco:CharacterString>
  </gmx:fieldOfApplication>
  <gmx:versionNumber>
    <gco:CharacterString>0.1</gco:CharacterString>
  </gmx:versionNumber>
  <gmx:versionDate>
    <gco:Date>2016-07-14</gco:Date>
  </gmx:versionDate>
  <!--============================================================================-->
  <!--============================================================================-->
  <!--============================== Codelists
=======================================-->
  <!--=== AuthenticationCode ===-->
  <gmx:codelistItem>
    <gmx:CodeListDictionary gml:id="AuthenticationCode">
      <gml:description>identification of authentication methods</gml:description>
      <gml:identifier codeSpace="SD">urn:sd:authentication</gml:identifier>
      <gmx:codeEntry>
        <gmx:CodeDefinition gml:id="SAML2_BROWSER_POST">
          <gml:description>In the scenario supported by the web browser SSO profile, a
web user either accesses a resource at a
                  service provider, or accesses an identity provider such that the
service provider and desired resource are
                  understood or implicit. The web user authenticates (or has already
authenticated) to the identity provider,
                  which then produces an authentication assertion (possibly with input
from the service provider) and the
                  service provider consumes the assertion to establish a security
context for the web user. During this
                  process, a name identifier might also be established between the
providers for the principal, subject to the
                  parameters of the interaction and the consent of the parties.
                  To implement this scenario, a profile of the SAML Authentication
Request protocol is used, in conjunction
                  with the HTTP Redirect, HTTP POST and HTTP Artifact bindings.
                  It is assumed that the user is using a standard commercial browser
and can authenticate to the identity
                  provider by some means outside the scope of SAML.</gml:description>
          <gml:identifier
codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:profiles:SSO:browser</gml:identifier>
        </gmx:CodeDefinition>
      </gmx:codeEntry>
      <gmx:codeEntry>
        <gmx:CodeDefinition gml:id="SAML2_ECP">
          <gml:description>An enhanced client or proxy (ECP) is a system entity that
knows how to contact an appropriate identity
```

```
                provider, possibly in a context-dependent fashion, and also supports
the Reverse SOAP (PAOS) binding
                [SAMLBind].
                An example scenario enabled by this profile is as follows: A
principal, wielding an ECP, uses it to either
                access a resource at a service provider, or access an identity
provider such that the service provider and
                desired resource are understood or implicit. The principal
authenticates (or has already authenticated)
                with the identity provider, which then produces an authentication
assertion (possibly with input from the
                service provider). The service provider then consumes the assertion
and subsequently establishes a
                security context for the principal. During this process, a name
identifier might also be established between
                the providers for the principal, subject to the parameters of the
interaction and the consent of the principal.
                This profile is based on the SAML Authentication Request protocol
[SAMLCore] in conjunction with the
                PAOS binding.</gml:description>
          <gml:identifier
codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp</gml:identifier>
        </gmx:CodeDefinition>
      </gmx:codeEntry>
      <gmx:codeEntry>
        <gmx:CodeDefinition gml:id="HTTP_BASIC">
          <gml:description>The "basic" authentication scheme is based on the model
that the
                    client must authenticate itself with a user-ID and a password for
                    each realm.  The realm value should be considered an opaque string
                    which can only be compared for equality with other realms on that
                    server. The server will service the request only if it can
validate
                    the user-ID and password for the protection space of the Request-
URI.
                    There are no optional authentication parameters.</gml:description>
          <gml:identifier
codeSpace="OGC">urn:ogc:def:tb12:ietf:2617:basic</gml:identifier>
        </gmx:CodeDefinition>
      </gmx:codeEntry>
      <gmx:codeEntry>
        <gmx:CodeDefinition gml:id="HTTP_DIGEST">
          <gml:description>Like Basic Access Authentication, the Digest scheme is
based on a
                simple challenge-response paradigm. The Digest scheme challenges
                using a nonce value. A valid response contains a checksum (by
                default, the MD5 checksum) of the username, the password, the given
                nonce value, the HTTP method, and the requested URI. In this way, the
                password is never sent in the clear. Just as with the Basic scheme,
                the username and password must be prearranged in some fashion not
                addressed by this document.</gml:description>
```

```
          <gml:identifier
codeSpace="OGC">urn:ogc:def:tb12:ietf:2617:digest</gml:identifier>
        </gmx:CodeDefinition>
      </gmx:codeEntry>
      <gmx:codeEntry>
        <gmx:CodeDefinition gml:id="OAUTH2_BEARER_TOKEN">
         <gml:description>In the scenario supported by the OAuth 2.0 SSO profile, a
web user or service either accesses a resource
            at a service provider, or accesses an identity provider such that the
service provider and desired resource are understood
            or implicit. The web user authenticates (or has already authenticated) to
the identity provider, Which then produces an
            authorization grant which was then used by an authorization service to
return an access token.  This access token then
            subtitutes as both authentication and authorization on future
requests.</gml:description>
          <gml:identifier
codeSpace="OGC">urn:ogc:def:tb12:ietf:6750:bearer_token</gml:identifier>
        </gmx:CodeDefinition>
      </gmx:codeEntry>
      <gmx:codeEntry>
        <gmx:CodeDefinition gml:id="CLIENT_CERTIFICATE">
          <gml:description>This type of authentication is an extension to the TLS
handshake as outlined in section 7.4.4:
            "A non-anonymous server can optionally request a certificate from
            the client, if appropriate for the selected cipher suite.  This
            message, if sent, will immediately follow the ServerKeyExchange
            message (if it is sent; otherwise, this message follows the
            server's Certificate message)."
            In case the client cannot provide a suitable and valid certificate, no TLS
            connection gets established</gml:description>
          <gml:identifier
codeSpace="OGC">urn:ogc:def:tb12:ietf:5246:client_certificate</gml:identifier>
        </gmx:CodeDefinition>
      </gmx:codeEntry>
      <gmx:codeEntry>
        <gmx:CodeDefinition gml:id="USERNAME_TOKEN">
          <gml:description>WSSE UsernameToken as specified in https://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf</gml:description>
          <gml:identifier
codeSpace="OGC">urn:ogc:def:tb12:wsse:username_token</gml:identifier>
        </gmx:CodeDefinition>
      </gmx:codeEntry>
    </gmx:CodeListDictionary>
  </gmx:codelistItem>
  <!--=== EOF ===-->
</gmx:CT_CodelistCatalogue>
```

NOTE

Use of GMX codelist

The benefit of using a GMX codelist is that it can also be used to annotate ISO 19139 metadata for services. But evaluating options for annotating ISO metadata is outside the scope of TB12.

Chapter 9. Verification of the Annotation Approach for Capabilities

NOTE

Verify that the annotation approach is comprehensive

The proposed approach is to annotate the service capabilities document using the ows:Constraint element from the OWS Common XSD for any server where the capabilities structure is based on OWS Common. For WMS 1.3, the use of the _ExtendedCapabilities element is proposed. The use of the ows:Constraint element does provide different options on how to annotate. The following examples of service security and the corresponding annotation examples shall help to understand and verify that the proposed approach for security annotation is comprehensive.

9.1. Introduction to the annotation examples

For the security annotation of capabilities documents, TB12 has defined the following urns to identify security constraints:

- urn:ogc:def:tb12:security:authentication is used to define security conditions regarding authentication;
- urn:ogc:def:tb12:security:authorisation is used to express authorisation rules in XACML or GeoXACML;
- urn:ogc:def:tb12:security:policy is used for SOAP security enabled services to reference a WS-Policy; and
- urn:ogc:def:tb12:security:layer4 is used to express TLS or SSL requirements.

The examples refer to security conditions that a service instance exposes to the client. Two possible options exist:

- The service is a SOAP enabled service; or
- The service is following the W*S approach (basically HTTP but no SOAP messages).

9.1.1. SOAP enabled services example

For SOAP security enabled service instances, all the security specific conditions are typically captured in the SOAP message. The SOAP message can either be a service input or output message. As the SOAP message structure is XML, one can apply all security related conditions that can be put on XML: a) Integrity by applying W3C XML Digital Signature; and b) Confidentiality by applying W3C XML Encryption.

The annotation for SOAP enabled services shall be done via a WS-Policy, which can be referenced from the Capabilities document using the OWS:Constraint element with the identifier "security:policy" or be inline to the service instance capabilities file. Both approaches involve a WS-Policy which can be used for creating client side code from the WSDL and at runtime from the capabilities document to apply/verify required security conditions are applied to the SOAP

envelope.

Capabilities referencing a WS-Policy

For annotating a secured SOAP service, only one example is necessary as all the individual conditions are expressed in the associated WS-Policy. The following example illustrates how to reference a WS-Policy from the security annotation inside the Capabilities document.

```
<ows:Constraint name="urn:ogc:def:tb12:security:policy">
  <ows:ValuesReference ows:reference="https://domain.com/WFSPolicy.xml"/>
</ows:Constraint>
```

9.1.2. Standard OGC Web Services annotation examples

The following examples illustrate this procedure.

HTTP over SSL 3.0 or TLS v1.2

The service endpoint uses the protocol scheme “https” and the requirement is that the client supports SSL 3.0 and TLS v1.2.

```
<ows:Operation name="GetCapabilities">
  <ows:DCP>
    <ows:HTTP>
      <ows:Get xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="https://domain.com/service/wms">
        <ows:Constraint name="urn:ogc:def:tb12:security:layer4”>
          <ows:AllowedValues>
            <ows:Value>urn:ietf:ssl:3.0</ows:Value>
            <ows:Value>urn:ietf:tls:1.2</ows:Value>
          </ows:AllowedValues>
        </ows:Constraint>
      </ows:Get>
    </ows:HTTP>
  </ows:DCP>
</ows:Operation>
```

HTTP over TLS 1.2 and Basic Authentication

```
<ows:Operation name="GetCapabilities">
  <ows:DCP>
    <ows:HTTP>
      <ows:Get xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="https://domain.com/service/wms">
        <ows:Constraint name="urn:ogc:def:tb12:security:layer4”>
          <ows:AllowedValues>
            <ows:Value>urn:ietf:ssl:3.0</ows:Value>
            <ows:Value>urn:ietf:tls:1.2</ows:Value>
          </ows:AllowedValues>
        </ows:Constraint>
        <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
          <ows:ValuesReference
ows:reference="http://tb12.opengis.net/security/authCodeList#BASIC_AUTH”>urn:ogc:def:t
b12:ietf:2701:basic_auth</ows:ValuesReference>
        </ows:Constraint>
      </ows:Get>
    </ows:HTTP>
  </ows:DCP>
</ows:Operation>
```

HTTP over TLS 1.2 and (Basic Authentication or Client Certificate) for a WMS only supporting HTTP Get operations

These security constraints must be normalised to a DNF as ((TLS 1.2 AND Authentication) OR (TLS 1.2 AND Client Certificate)) which requires that the ows:Get element is duplicated. Each ows:Get element expresses one possible option for the client to interact with the service endpoint.

```
<ows:Operation name="GetCapabilities">
  <ows:DCP>
    <ows:HTTP>
      <ows:Get xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="https://domain.com/service/wms">
        <ows:Constraint name="urn:ogc:def:tb12:security:layer4”>
          <ows:AllowedValues>
            <ows:Value>urn:ietf:tls:1.2</ows:Value>
          </ows:AllowedValues>
        </ows:Constraint>
        <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
          <ows:ValuesReference
ows:reference="http://tb12.opengis.net/security/authCodeList#HTTP_BASIC”>urn:ogc:def:t
b12:ietf:2617:basic</ows:ValuesReference>
        </ows:Constraint>
      </ows:Get>
      <ows:Get xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="https://domain.com/service/wms">
        <ows:Constraint name="urn:ogc:def:tb12:security:layer4”>
          <ows:AllowedValues>
            <ows:Value>urn:ietf:tls:1.2</ows:Value>
          </ows:AllowedValues>
        </ows:Constraint>
        <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
          <ows:ValuesReference
ows:reference="http://tb12.opengis.net/security/authCodeList#CLIENT_CERTIFICATE”>urn:o
gc:def:tb12:ietf:5246:client_certificate</ows:ValuesReference>
        </ows:Constraint>
      </ows:Get>
    </ows:HTTP>
  </ows:DCP>
</ows:Operation>
```

HTTP over TLS 1.2 and Basic Authentication for HTTP Get and HTTP over TLS 1.2 and Client Certificate for HTTP Post

These security constraints must be normalized to a DNF as ((TLS 1.2 AND Authentication for Get) OR (TLS 1.2 AND Client Certificate of Post)) which requires that the ows:Get and ows:Post elements are used accordingly. Each ows:Get or ows:Post element expresses one possible option for the client to interact with the service endpoint.

```
<ows:Operation name="GetCapabilities">
  <ows:DCP>
    <ows:HTTP>
      <ows:Get xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="https://domain.com/service/wms">
        <ows:Constraint name="urn:ogc:def:tb12:security:layer4”>
          <ows:AllowedValues>
            <ows:Value>urn:ietf:tls:1.2</ows:Value>
          </ows:AllowedValues>
        </ows:Constraint>
        <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
          <ows:ValuesReference
ows:reference="http://tb12.opengis.net/security/authCodeList#BASIC_AUTH”>urn:ogc:def:t
b12:ietf:2701:basic_auth</ows:ValuesReference>
        </ows:Constraint>
      </ows:Get>
      <ows:Post xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="https://domain.com/service/wms">
        <ows:Constraint name="urn:ogc:def:tb12:security:layer4”>
          <ows:AllowedValues>
            <ows:Value>urn:ietf:tls:1.2</ows:Value>
          </ows:AllowedValues>
        </ows:Constraint>
        <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
          <ows:ValuesReference
ows:reference="http://tb12.opengis.net/security/authCodeList#CLIENT_CERTIFICATE”>urn:o
gc:def:tb12:ietf:5246:client_certificate</ows:ValuesReference>
        </ows:Constraint>
      </ows:Post>
    </ows:HTTP>
  </ows:DCP>
</ows:Operation>
```

Annotation example for WMS 1.3

The security annotation for WMS is possible via the definition of a specialization of the _ExtendedCapabilities element, which is abstract in the WMS 1.3 capabilities schema. The following examples illustrates the use of the extension.

```
    <ows_security:ExtendedSecurityCapabilities>
      <ows:OperationsMetadata>
        <ows:Operation name="GetCapabilities">
          <ows:DCP>
            <ows:HTTP>
              <ows:Get xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="https://domain.com/service/wms">
                <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
                  <ows:ValuesReference
ows:reference="http://tb12.opengis.net/security/authCodeList#CLIENT_CERTIFICATE">urn:o
gc:def:tb12:ietf:5246:client_certificate</ows:ValuesReference>
                </ows:Constraint>
              </ows:Get>
            </ows:HTTP>
          </ows:DCP>
        </ows:Operation>
        <ows:Operation name="GetMap">
          <ows:DCP>
            <ows:HTTP>
              <ows:Get xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="https://domain.com/service/wms">
                <ows:Constraint name="urn:ogc:def:tb12:security:authentication">
                  <ows:ValuesReference
ows:reference="http://tb12.opengis.net/security/authCodeList#CLIENT_CERTIFICATE">urn:o
gc:def:tb12:ietf:5246:client_certificate</ows:ValuesReference>
                </ows:Constraint>
                <ows:Constraint name="urn:ogc:def:tb12:security:authorization">
                  <ows:ValuesReference
ows:reference="https://domain.com/TB12.geoxacml.xml"/>
                </ows:Constraint>
              </ows:Get>
            </ows:HTTP>
          </ows:DCP>
        </ows:Operation>
      </ows:OperationsMetadata>
    </ows_security:ExtendedSecurityCapabilities>
```

Chapter 10. Change Requests and Requirements for Clients

This section summarizes topics that are required to be submitted as Change Requests (CRs) to OGC, requirements for future work and best practices discussions relevant for progressing with security for OGC Web Services.

10.1. Change Requests

The identification of CRs on one hand is based on the disconnect of OGC Web Service specifications from mainstream IT as outlined in OGC 15-022 which limits the options to be interoperable for secured services. To close the gap is the main objective behind the identified CRs.

10.1.1. W*S must support content-type application/www-form-urlencoded for POST requests

This CR is captured under OGC CR #388: http://ogc.standardstracker.org/show_request.cgi?id=388

As outlined in the CR, the main problem is that mainstream IT uses content-type application/www-form-urlencoded for HTML + FORM elements to post a request from the client application to the service, but W*S specifications do not. For HTML5, a FORM element can only have one of the following content-types [source: http://www.w3schools.com/tags/tag_form.asp]:

- application/x-www-form-urlencoded: Default. All characters are encoded before sent (spaces are converted to "+" symbols, and special characters are converted to ASCII HEX values)
- multipart/form-data: No characters are encoded. This value is required when you are using forms that have a file upload control
- text/plain: Spaces are converted to "+" symbols, but no special characters are encoded

In OGC W*S specifications, if they support HTTP POST, the content-type is “text/xml” or “application/xml” or similar. This does prevent a simple HTML5 document including a FORM element to submit a XML encoded request to a W*S.

The CR does identity the main limitation regarding security as follows: For a secured W*S endpoint, some authentication methods (e.g. SAML) that store a security context at the service side require a two-stage communication from the client to the service to establish the session. In stage one, the service returns to the client a temporary session which gets completed during stage two. It is therefore important that the client can point to the previous (incomplete) session. The SAML implementation Shibboleth triggers the second stage by returning an HTTP status 302 (for a HTTP Get request) or 307 (for a HTTP Post request) to the client. The reference to the preliminary session is done via extra parameters.

But, the current way OGC POST requests work, it is not possible for the service to add specific parameters, as the structure is defined by an OGC maintained XSD and the content-type must be text/xml or application/xml.

Due to that limitation, it is not possible to establish a session with W*S using a POST request, which is a shortcoming that introduces a disconnect from main stream IT, where almost all client/service interactions via POST use the content-type application/www-form-urlencoded.

10.1.2. The implementation of a secured W*S shall support IETF RFC 2818 (HTTPS)

This CR is captured under OGC CR #417: http://ogc.standardstracker.org/show_request.cgi?id=417

This requirement is important to enable that a security facade can be setup in front of any W*S to require the use of HTTPS.

Also, the support of HTTPS, as standardized in the IETF RFC 2818 (HTTP over TLS) must become a normative reference.

10.1.3. Improved DCP Description

This CR is captured under OGC CR #418: http://ogc.standardstracker.org/show_request.cgi?id=418

The use of OWS Common limits the binding of OGC Web Services to HTTP. However, secured endpoints that require confidential transport of request and responses are leveraging HTTP over TLS (HTTPS). But the OWS Common schema limits the use of the DCP subtype HTTP which indicates an unsecured communication channel. The OWS Common Schema must be extended to support HTTPS to indicate a secure use of HTTP per TLS.

10.1.4. The implementation of a secured W*S shall be able to support HTTP Cookies

This CR is captured under OGC CR #419: http://ogc.standardstracker.org/show_request.cgi?id=419

The ability - in general - to support HTTP Cookies ensures that it is possible to store state on the client side, including a reference to a security context stored at the server side.

NOTE	*Support HTTP Cookies* The support for HTTP Cookies in a service instance is optional and mainly depends on the security policy for your site.

10.1.5. The implementation of a secured W*S must be able to fully support of IETF RFC 2616 (HTTP/1.1)

This CR is captured under OGC CR #420: http://ogc.standardstracker.org/show_request.cgi?id=420

Permit the use of all HTTP status codes as it fits…

For REST, we need the use of HEAD, OPTIONS and all status codes…

10.1.6. WFS and the use of DescribeFeatureType in response XML documents

This CR is captured under OGC CR #421: http://ogc.standardstracker.org/show_request.cgi?id=421

The WFS response for a GetFeature operation is a FeatureCollection, which can be encoded in XML. For an XML encoded response, it is common practice that the WFS DescribeFeature operation is used to reference to the schema fragment that defines the features types included in the feature collection. As this schema portion is required to validate the XML document, it is problematic if the service endpoint is protected and so the DescribeFeatureType operation. In this case, it is recommended to use a URL that is fully comprised with security relevant information to enable the XML parser to gain access to the schema fragment. One approach could be to issue an OAuth access token with a short lifetime and attach that to the URL for obtaining the schema fragment that defines the involved feature types. In the case where the DescribeFeatureType operation is protected by HTTP Basic Authentication for example, the client implementation must add the proper user credentials with the request, which is difficult to implement in a safe way without exploiting the user credentials. Ergo, a non solution would be to create a DescribeFeatureType URL inside the FeatureCollection XML of the structure http://<user>:<password>@<host>/DescribeFeatureType? ...

10.1.7. WPS Execute Request using POST data

This CR is captured against the WPS 2.0 under OGC CR #422: http://ogc.standardstracker.org/show_request.cgi?id=422

A WPS can be deployed to operate as a facade (or proxy) in front of backend services. According to the W*S type of the backend service, a request can be either HTTP Get or Post. In cases where the WPS process has received an XML encoded request for the backend service, it is important that the request is **not** processed but just relayed to the backend service. This is important in cases where the backend service is a SOAP + security enabled service and the client submits the request to that service via the WPS Execute request. In case where a digital signature is on that request, any processing at the WPS would invalidate the request and the backend service would return an error. An example for this constellation is captured in chapter 11.

10.1.8. Secure implementation of a WMS 1.1.1 using security annotated capabilities

This CR is captured under OGC CR #423: http://ogc.standardstracker.org/show_request.cgi?id=423

As outlined in the ER in an earlier section, the WMS 1.1.1 capabilities are not based on OWS Common schema but on a WMS version specific DTD. In order to annotate the security conditions according to the proposed formalism, the following DTD structure must be included inside the VendorSpecificCapabilities element.

NOTE	*WMS 1.1.1 security annotation in capabilities document* For WMS 1.1.1, Only DCPType HTTP is allowed and because the proposed solution below is meant to be backwards-compatible, the proposed DTD element is X_HTTP and not HTTPS.

In case where the service endpoint must not include a vendor specific section, then the following structure shall be used:

```
<!DOCTYPE WMT_MS_Capabilities SYSTEM "WMS_MS_Capabilities.dtd"[

<!--
============================================================================
OWS Common Security Extension to annotate security
requires to add element X_OperationsMetadata to
your Vendor Specific Capabilities definition
If you do not define your own VendorSpecificCapabiltiies
then use this element
<!ELEMENT VendorSpecificCapabilities (X_OperationsMetadata) >
============================================================================
-->

<!ELEMENT VendorSpecificCapabilities (X_OperationsMetadata) >

<!--
============================================================================
OWS Common Security Extension to annotate security
Definition of element X_OperationsMetadata replicating the
definition from the OWS Common Schema to become available as DTD
============================================================================
-->

<!ELEMENT X_OperationsMetadata (X_Operation*) >
<!ELEMENT X_Operation (X_DCP+) >
<!ATTLIST X_Operation
  name CDATA #REQUIRED
>

<!ELEMENT X_DCP (X_HTTP) >
<!ELEMENT X_HTTP (X_Get | X_Post)+ >

<!ELEMENT X_Get (X_Constraint+)>
<!ATTLIST X_Get
xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
xlink:type CDATA #FIXED "simple"
xlink:href CDATA #REQUIRED >

<!ELEMENT X_Post (X_Constraint+)>
<!ATTLIST X_Post
xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
xlink:type CDATA #FIXED "simple"
xlink:href CDATA #REQUIRED >

<!ELEMENT X_Constraint (X_AllowedValues | X_ValuesReference)+>
<!ATTLIST X_Constraint
  name CDATA #REQUIRED
```

```
>

<!ELEMENT X_AllowedValues (Value)+>
<!ELEMENT X_Value (#PCDATA)>

<!ELEMENT X_ValuesReference (#PCDATA)>
<!ATTLIST X_ValuesReference
  reference CDATA #REQUIRED
>


]>
```

In case the use of VendorSpecificCapabilities is required to define vendor specific portions in the capabilities, the security extension must be integrated into the vendor's definition. The following example illustrates that approach:

```
<!DOCTYPE WMT_MS_Capabilities SYSTEM "WMS_MS_Capabilities.dtd"[

<!--
================================================================================
OWS Common Security Extension to annotate security
requires to add element X_OperationsMetadata to
your Vendor Specific Capabilities definition
If you do not define your own VendorSpecificCapabiltiies
then use this element
<!ELEMENT VendorSpecificCapabilities (X_OperationsMetadata) >
================================================================================
-->

<!ELEMENT VendorSpecificCapabilities (Profiles, FeatureCollection,
X_OperationsMetadata) >
<!ELEMENT Profiles (Profile*) >
<!ELEMENT Profile (#PCDATA) >
<!ELEMENT FeatureCollection (#PCDATA) >


<!--
================================================================================
OWS Common Security Extension to annotate security
Definition of element X_OperationsMetadata replicating the
definition from the OWS Common Schema to become available as DTD
================================================================================
-->

<!ELEMENT X_OperationsMetadata (X_Operation*) >
<!ELEMENT X_Operation (X_DCP+) >
<!ATTLIST X_Operation
  name CDATA #REQUIRED
>
```

```
<!ELEMENT X_DCP (X_HTTP) >
<!ELEMENT X_HTTP (X_Get | X_Post)+ >

<!ELEMENT X_Get (X_Constraint+)>
<!ATTLIST X_Get
xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
xlink:type CDATA #FIXED "simple"
xlink:href CDATA #REQUIRED >

<!ELEMENT X_Post (X_Constraint+)>
<!ATTLIST X_Post
xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
xlink:type CDATA #FIXED "simple"
xlink:href CDATA #REQUIRED >

<!ELEMENT X_Constraint (X_AllowedValues | X_ValuesReference)+>
<!ATTLIST X_Constraint
  name CDATA #REQUIRED
>

<!ELEMENT X_AllowedValues (Value)+>
<!ELEMENT X_Value (#PCDATA)>

<!ELEMENT X_ValuesReference (#PCDATA)>
<!ATTLIST X_ValuesReference
  reference CDATA #REQUIRED
>


]>
```

10.1.9. Secure implementation of a WMS 1.3.0 using security annotated capabilities

This CR is captured under OGC CR #424: http://ogc.standardstracker.org/show_request.cgi?id=424

For annotating WMS 1.3.0 capabilities with security annotations, the following XSD structure shall be used. The following schema defines the ExtendedSecurityCapabilities element which is in the substitution group _ExtendedCapabilties and can therefore be used in the capabilities document either exclusively or in combination with other extended capabilities elements. The example in a previous section of this ER outlines an example for a DigitalGlobe WMS 1.3.0, but the approach would also be valid for an INSPIRE download service.

NOTE	*WMS 1.3.0 security annotation of capabilities document* The proposed extended security type is compatible with WMS 1.3.0 and therefore leverages the DCPType HTTP even for service endpoints that are actually HTTPS.

The following ExetendedSecurityCapabilities type definition shall be used:

```
<schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:ows_security="http://wwww.opengis.net/security/1.0"
    xmlns:ows="http://www.opengis.net/ows/1.1"
    xmlns:wms="http://www.opengis.net/wms"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    targetNamespace="http://wwww.opengis.net/security/1.0"
    elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0.0">
    <import namespace="http://www.opengis.net/wms"
schemaLocation="http://schemas.opengis.net/wms/1.3.0/capabilities_1_3_0.xsd"/>
    <import namespace="http://www.opengis.net/ows/1.1"
schemaLocation="http://schemas.opengis.net/ows/1.1.0/owsOperationsMetadata.xsd"/>
    <xs:complexType name="ExtendedSecurityCapabilitiesType">
      <sequence>
        <element ref="ows:OperationsMetadata"/>
      </sequence>
    </xs:complexType>
    <element name="ExtendedSecurityCapabilities"
type="ows_security:ExtendedSecurityCapabilitiesType"
substitutionGroup="wms:_ExtendedCapabilities"/>
</schema>
```

10.1.10. Security Annotations in Capabilities Documents require URNs

This CR is captured under OGC CR #425: http://ogc.standardstracker.org/show_request.cgi?id=425

The use of security annotations in the Capabilities document (response) is based on well defined URNs with pre-defined meaning. In order to achieve an interoperable approach, certain URNs must be approved by the OGC Naming Authority.

NOTE

URNs must get OGC scope before being submitted to the OGC Naming Authority

URNs must be re-defined by the OWS Common - Security SWG before submitted to the OGC Naming Authority.

The following URNs have being used for Testbed 12:

- Authentication: urn:ogc:def:tb12:security:authentication;
- Authorization: urn:ogc:def:tb12:security:authorization;
- Integrity: urn:ogc:def:tb12:security:integrity;
- Confidentiality: urn:ogc:def:tb12:security:confidentiality; and
- Security Policy: urn:ogc:def:tb12:security:policy.

Also, the URL location of the Authentication Codelist must be specified by the OGC Naming Authority. The URL is important to allow clients the resolving of the identifiers used in the Authentication Codelist.

NOTE	*Use of Integrity and Confidentiality* Ensuring backwards compatibility, it is not possible to use another DCP type than HTTP. In order to outline the use of HTTPS, the allowed codes shall be listed as possible allowed values for Integrity and Confidentiality. For example SSL3.0, TLS1.2, etc.

NOTE	*Authentication Codelist URL* URL must be re-defined by the OWS Common - Security SWG before submitted to the OGC Naming Authority.

For Testbed 12, the following URL has being used for the AuthenticationCodeList: http://tb12.opengis.net/security/authCodeList

The actual identifiers for the different authentication schemes that comprise the authentication codelist must be approved by the OGC Naming Authority.

NOTE	*URN values for Authentication Codes must be reviewed before submitted to the OGC Naming Authority* The actual values of the URNs used in Testbed 12 cannot be used for an Implementation Specification. The OWS Common - Security SWG must define the URNs and submit them to the OGC Naming Authority.

10.1.11. Cross-Origin-Resource-Sharing (CORS)

This CR is captured under OGC CR #426: http://ogc.standardstracker.org/show_request.cgi?id=426

When a secured OGC Web Service (e.g. using HTTPS) is called from a Web Application via JavaScript code, the Web Browser applies certain sand-boxes to the code to ensure privacy and security and to prevent Cross-Site-Scripting.

The W3C CORS recommendation shall be adopted on the server side to ensure that OGC Web Service requests, where XML encoded responses are expected (e.g. GetCapabilities, DescribeFeatureType, GetFeature, etc.) are not blocked by the Web Browser due to a different Origin.

The full conditions when CORS applies can be found in the W3C CORS recommendation: https://www.w3.org/TR/cors/

10.2. Requirements for Clients

The topics listed here are important to be considered when implementing clients that shall be able to work with secured services. The list of topics is not meant to be complete, because only topics arising from Testbed 12 are captured.

Because OGC has not released Client Implementation Specifications, the topics listed here cannot be captured as a Change Request!

10.2.1. Requirement 1 - Client applications shall support HTTPS

For any client application that shall be used on a secured service, it is important that the implementation supports HTTP over TLS (IETF RFC 2818). This support requires the use of modern and strong ciphers (TLS 1.2) and refrain from using weak ciphers such as SSL 3.0. Further more, it is important that server certificates are properly introspected and certificate revocation lists are used when validating certificates.

10.2.2. Requirement 2 - Client applications shall support HTTP Cookies

The use of OWS Common limits a W*S to be stateless. This is acceptable for unsecured services, but when applying a particular security technology or standard, it must be possible that a client refers to a security context via an HTTP Cookie.

10.2.3. Requirement 3 - Client applications must support full implementation of IETF RFC 2616 (HTTP/1.1)

OWS Common has a mandatory reference to IETF RFC 2616, which is HTTP 1.1. Even though client specifications do not exist, all client implementations must support proper functions for all HTTP status codes, including 302 and 307. This is important when enterprise security authentication is implemented at the service, as it is defined in e.g. SAML2.

Also, the appropriate use of **all** HTTP verbs is encouraged, including the use of HTTP OPTIONS and HEAD operations.

10.3. Best Practices Discussion

The topics listed here cannot be submitted as a CR and also not as a client requirement. Instead, the topics identify general problems to be solved in the OGC

10.3.1. Best Practices Discussion 1 - How to validate XML encoded responses for W*S when the schema is not publicly accessible?

The WFS returns an XML encoded response for the GetFeature operation; the FeatureCollection. In order for the receiving application to validate the XML, a schema is required. General practice is to refer to the schema of the FeatureCollection using the WFS DescribeFeatureType operation. But for a secured WFS where the schema is sensitive and not publicly accessible, the client side XML parser cannot simply use the DescribeFeatureType operation, as this is a secured operation. The general question is how can the XML parse execute the secured operation DescribeFeatureType without unveiling the credentials required for this operation?

10.3.2. Best Practices Discussion 2 - How to validate XML encoded responses for a SOAP enabled service with WS-Security

For any W*S instance that supports SOAP and enforces security conditions on the SOAP envelope, no HTTP GET operations must be used. The main reason is that the security conditions for the SOAP envelope (typically described in a WS-Policy) can **not** be modeled in a HTTP GET request. It is therefore not possible that e.g. a WFS supports a DescribeFeatureType via HTTP GET in cases where

this operation is also protected via SOAP security.

Therefore, the general question is how to enable the client side parser to obtain the schema without following the DescribeFeatureType link outlined in the XML response; the FeatureCollection?

Chapter 11. Results from securing Services in Testbed 12

NOTE	*Technical Experiment and Verification* The main contribution from each TB12 participant that puts security on a service is to verify if the deployment is OGC standard-compliant and if not, what the delta is with respect to compliance.

This section is separated into contributions related to server-side or client-side activities. Overall, contributions for this section have been received from the following Testbed 12 CMD Thread participants (in alphabetic order):

- 52North
- DigitalGlobe
- Cubewerx
- Compusult
- Esri
- GMU
- Luciad
- Secure Dimensions
- Rasdaman

11.1. Server Side Contributions from Secure Dimensions, DigitalGlobe, Rasdaman, 52North and Cubewerx

For TB12, Secure Dimensions has deployed various security facades to secure otherwise unsecured services from different TB12 participating organizations. The following sub-sections summarize the secured services contribution from Secure Dimensions. The TIE results from these endpoints using annotated public capabilities is documented in the following chapter.

11.1.1. Secured Services for DigitalGlobe

For securing services from DigitalGlobe, the security architecture is very much identical to the security model from DHS (Department of Homeland Security). In a nutshell, the architecture involves user authentication via personal certificates, issued for TB12 from OGC and Secure Dimensions. The service endpoints are secured with client side TLS which requires the user to select his TB12 certificate to establish a TLS connection. With the CN from the certificate, the security facade requests user attributes from a SAML Attribute Authority, which is operated by OGC for TB12. From that user repository, a set of different user attributes becomes available for undertaking the authorization. In particular, the security implements the OASIS SAML Profile

“SAML Attribute Sharing Profile for X.509 Authentication-Based Systems”.

The access control itself is based on GeoXACML policy based authorization which involves geographic conditions for military installations and otherwise sensitive areas.

The following W*S are available for the DigitalGlobe services stack:

- WMS 1.1.1: https://tb12.ogc.secure-dimensions.com/service/wms?request=GetCapabilities&service=WMS&version=1.1.1
- WMS 1.3.0: https://tb12.ogc.secure-dimensions.com/service/wms?request=GetCapabilities&service=WMS&version=1.3.0
- WMTS 1.0.0: https://tb12.ogc.secure-dimensions.com/service/wms?request=GetCapabilities&service=WMTS&version=1.0.0
- WFS 1.1.0: https://tb12.ogc.secure-dimensions.com/service/wfs?request=GetCapabilities&service=WFS&version=1.1.0
- WCS 1.1.1: https://tb12.ogc.secure-dimensions.com/service/wcs?request=GetCapabilities&service=WCS&version=1.1.0

For these services, annotated capabilities and all required schema materials are available from this publicly and open URL: https://tb12.ogc.secure-dimensions.com/capabilities

In terms of access control and authorization, the WMS and WMTS endpoints operate on a GeoXACML policy that enforces redaction of sensitive areas for all users that do not have sufficient rights based on their role and (TB12) clearance. For the WMS response, the entire image is used for redaction. For the WMTS endpoint, the applicable tile that intersects the given sensitive area is redacted. Depending on the zoom level, it is possible that multiple tiles get redacted before returned to the client:

- Alcatraz: This island is chosen as a sensitive area close to San Francisco;
- Armory: This military base is chosen as a larger installation; and
- Nike: This is a small military installation.

11.1.2. Secured Service for Cubewerx

For Cubewerx, a WFS 2.0 with SOAP support is secured using a SOAP+WS-Security as described in a WS-Policy. In order to execute the secured service, the client must send a particular security header inside the SOAP envelope which matches the requirements from the WS-Policy.

This SOAP + WS-Security enabled WFS is available here: https://tb12.secure-dimensions.com/soap/services

For this service, a static capabilities document with public access exists, as well as the WSDL and the WS-Policy:

- Capabilities: https://tb12.secure-dimensions.com/capabilities/WFS-2.0.0.xml
- WSDL: https://tb12.secure-dimensions.com/WFS.wsdl
- WS-Policy: https://tb12.secure-dimensions.com/WFSSecurityPolicy.xml

11.1.3. Secured Service for rasdaman

For rasdaman, a WCPS security implementation exists which connects to a rasdaman WCS 2.0.1. The security is based on HTTPS and Basic Authentication. Based on the user's identifier, the security proxy does rewrite POST requests from a client with the following effects.

- Alcatraz: For all users that are not allowed to see Alcatraz, a water color blend-over is applied.
- Armory: For all users that are not allowed to see Armory, the military installation is covered by a football field.
- Nike: For all users that are not allowed to see the Nike installation, other coverages showing wood are blend-over the actual launch pad locations.

For this service, a static capabilities document including security annotations is available here: https://tb12.secure-dimensions.com/capabilities/WCPS-2.0.1.xml

The following URL can be used to execute the secured service via a Web Browser based client application: http://rasdaman.org/tb12

11.1.4. Verify the use of security standards

HTTPS

The service instances are hosted on HTTPS endpoints. Therefore, the client must have implemented support for IETF RFC 2818.

The use of HTTPS is a problem as not many GIS desktop clients support this. This is caused by the fact that the OGC Web Service suite of standards does not mandate the implementation of HTTPS. A change request to OWS Common is required to ensure interoperability.

The use of client side certificate, as one option in the TLS RFC to establish a secure connection, reduces the stack of available clients dramatically. For connecting to WMS and WMTS, ArcMap can be used. But QGIS does not support HTTPS + client side certificate.

HTTP Cookies

The service endpoints create (secure) cookies to enable the client to reference a security context, stored at the service side.

The OGC Suite of Web Service standards does not mandate the client to support HTTP cookies - actually, an OGC Web Services is supposed to be stateless. A Change Request to OWS Common is required to enable interoperability for HTTP cookies.

ArcMap successfully supported the use of HTTP cookies.

HTTP BASIC Authentication

The use of HTTP Basic Authentication - whether over HTTP or HTTPS - typically does work with clients, as the network libraries typically support this feature. However, the Suite of OGC Web Services standards does not mandate support for HTTP Authentication.

The use of HTTP Basic Authentication was successfully demonstrated with QGIS and ArcMap.

11.1.5. Contribution from 52North and Secure Dimensions

52North has deployed a Web Processing Service (WPS) with a process "testbed12.cmd.AsyncFacadeProcess" which is acting as a proxy in front of any Web Service which accepts a POST request using XML encoding and content-type "text/xml" or "application/xml". The WPS endpoint is here: http://tb12.dev.52north.org/wps/WebProcessingService

The description of the process can be obtained via this URL: http://tb12.dev.52north.org/wps/WebProcessingService?Request=DescribeProcess&Service=WPS&version=2.0.0&identifier=testbed12.cmd.AsyncFacadeProcess

This WPS process was leveraged to demonstrate the execution of a SOAP WFS with WS-Security, as illustrated in the following sequence diagram. The SOAP + WS-Security enabled WFS is provided by Secure Dimensions via this URL: https://tb12.secure-dimensions.com/soap/services

The security requirement is described in the annotated capabilities available here: https://tb12.secure-dimensions.com/capabilities/WFS-SOAP-2.0.0.xml

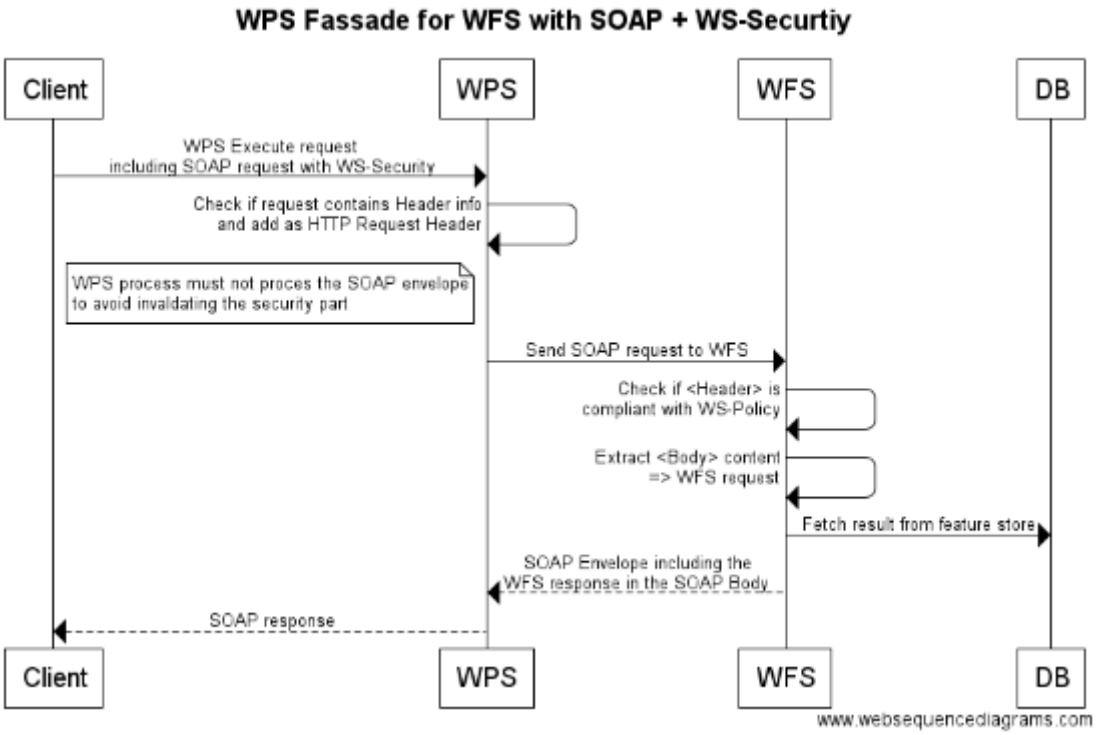

Figure 10. Executing a SOAP + WS-Security WFS 2.0 via a WPS process

```
title WPS Fassade for WFS with SOAP + WS-Securtiy

Client->WPS: WPS Execute request\nincluding SOAP request with WS-Security
WPS->WPS: Check if request contains Header info\nand add as HTTP Request Header
note left of WPS: WPS process must not process the SOAP envelope\nto avoid
invalidating the security part
WPS->WFS: Send SOAP request to WFS
WFS->WFS: Check if <Header> is\ncompliant with WS-Policy
WFS->WFS: Extract <Body> content\n=> WFS request
WFS->DB: Fetch result from feature store
WFS-->WPS: SOAP Envelope including the\nWFS response in the SOAP Body
WPS-->Client: SOAP response
```

The following listing contains the XML encoded WPS Execute request that can be sent to the WPS process to execute the secured WFS. Please note the WS-Security specific elements in the Header element of the SOAP request.

```
<wps:Execute xmlns:wps="http://www.opengis.net/wps/2.0"
xmlns:ows="http://www.opengis.net/ows/2.0"
  xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0
http://schemas.opengis.net/wps/2.0/wps.xsd"
  service="WPS" version="2.0.0" response="document" mode="async">
  <ows:Identifier>testbed12.cmd.AsyncFacadeProcess</ows:Identifier>
  <wps:Input id="request">
    <wps:Data mimeType="application/soap%2Bxml">
      <soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
        <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
          <wsa:To>https://tb12.secure-dimensions.com/soap/services/ows_proxy</wsa:To>
          <wsa:Action>http://www.opengis.net/wfs/requests#GetFeature</wsa:Action>
          <wsa:MessageID>urn:uuid:ae6c2f16-c6a1-1e51-226b-002522163135</wsa:MessageID>
          <wsse:Security
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
            soapenv:mustUnderstand="1">
            <wsse:BinarySecurityToken
              EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary"
              xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
              wsu:Id="CertID-ae6d196c-c6a1-1e51-226c"
              ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-
token-profile-1.0#X509v3"

>MIIDDDCCAfSgAwIBAgIQM6YEf7FVYx/tZyEXgVComTANBgkqhkiG9w0BAQUFADAwMQ4wDAYDVQQKDAVPQVNJU
zEeMBwGA1UEAwwVT0FTSVMgSW50ZXJvcCBUZXN0IENBMB4XDTA1MDMxOTAwMDAwMFoXDTE4MDMxOTIzNTk1OVo
wQjEOMAwGA1UECgwFT0FTSVMxIDAeBgNVBAsMF09BU0lTIEludGVyb3AgVGVzdCBDZXJ0MQ4wDAYDVQQDDAVBb
GljZTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAoqi99By1VYo0aHrkKCNT4DkIgPL/SgahbeKdGhrbu3K
```

```
2XG7arfD9tqIBIKMfrX4Gp90NJa85AV1yiNsEyvq%2BmUnMpNcKnLXLOjkTmMCqDYbbkehJlXPnaWLzve%2BmW
0pJdPxtf3rbD4PS/cBQIvtpjmrDAU8VsZKT8DN5Kyz%2BEZsCAwEAAaOBkzCBkDAJBgNVHRMEAjAAMDMGA1UdH
wQsMCowKKImhiRodHRwOi8vaW50ZXJvcC5iYnRlc3QubmV0L2NybC9jYS5jcmwwDgYDVR0PAQH/BAQDAgSwMB0
GA1UdDgQWBBQK4l0TUHZ1QV3V2QtlLNDm%2BPoxiDAfBgNVHSMEGDAWgBTAnSj8wes1oR3WqqqgHBpNwkkPDzA
NBgkqhkiG9w0BAQUFAAOCAQEABTqpOpvW%2B6yrLXyUlP2xJbEkohXHI5OWwKWleOb9hlkhWntUalfcFOJAgUy
H30TTpHldzx1%2BvK2LPzhoUFKYHE1IyQvokBN2JjFO64BQukCKnZhldLRPxGhfkTdxQgdf5rCK/wh3xVsZCNT
fuMNmlAM6lOAg8QduDah3WFZpEA0s2nwQaCNQTNMjJC8tav1CBr6%2BE5FAmwPXP7pJxn9Fw9OXRyqbRA4v2y7
YpbGkG2GI9UvOHw6SGvf4FRSthMMO35YbpikGsLix3vAsXWWi4rwfVOYzQK0OFPNi9RMCUdSH06m9uLWckiCxj
os0FQODZE9l4ATGy9s9hNVwryOJTw==</wsse:BinarySecurityToken>
            <wsse:UsernameToken
              xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
              wsu:Id="SigID-ae6d8e9c-c6a1-1e51-226d">
              <wsse:Username>Alice</wsse:Username>
              <wsse:Password
                Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordDigest"
                >%2B43faXkM59Wvc/1EqkhqFYGJwOs=</wsse:Password>
              <wsse:Nonce>p3aGzO%2ByN4I8oTehOJTY3ZrIcYsM6XJ8</wsse:Nonce>
              <wsu:Created>2016-01-29T16:02:22.307Z</wsu:Created>
            </wsse:UsernameToken>
            <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
              Id="SigID-ae6db5ac-c6a1-1e51-226e">
              <ds:SignedInfo>
                <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
                <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/>
                <ds:Reference URI="#SigID-ae6d8e9c-c6a1-1e51-226d">
                  <ds:Transforms>
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                  <ds:DigestValue>sk06OjMEwRt9U0PZYPTR3CVycL4=</ds:DigestValue>
                </ds:Reference>
              </ds:SignedInfo>

<ds:SignatureValue>bcI24Ds8MvM7dvgAC3WeO%2Buv9zKAq6yB3qdI02t8I1h7W8GVBrPYs4f7a9JdELtV0
UsA1NUOp4nbPwtPUdw%2BnVURHQUHr%2BmZ879HZuzD3daW5Z3kTk9i1%2B6ceCbZh/K/SRCFOQvDpKRC74W5i
Tkik7TBALgPl7sJxVt7nY6GBhs=</ds:SignatureValue>
              <ds:KeyInfo>
                <wsse:SecurityTokenReference>
                  <wsse:Reference URI="#CertID-ae6d196c-c6a1-1e51-226c"
                    ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3"
                  />
                </wsse:SecurityTokenReference>
              </ds:KeyInfo>
            </ds:Signature>
```

```
          </wsse:Security>
        </soapenv:Header>
        <soapenv:Body>
          <GetCapabilities service="WFS" version="2.0.0"
xmlns="http://www.opengis.net/wfs/2.0"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.opengis.net/wfs/2.0
            http://schemas.opengis.net/wfs/2.0.0/wfs.xsd"
          />
        </soapenv:Body>
      </soapenv:Envelope>
    </wps:Data>
  </wps:Input>
  <wps:Input id="endpoint-url">
    <wps:Data>
      <wps:LiteralValue>https://tb12.secure-
dimensions.com/soap/services</wps:LiteralValue>
    </wps:Data>
  </wps:Input>
  <wps:Output id="response" transmission="value" mimeType="application/soap%2Bxml"/>
</wps:Execute>
```

11.1.6. Contribution from Cubewerx

For TB12, Cubewerx has provided a set of secured services via the Cubewerx CwAuth and HTTP Basic Authentication.

For the secured endpoints, Cubewerx provides freely accessible Capabilities with security annotation:

- https://tb12.cubewerx.com/a007/cubeserv?datastore=USGS&SERVICE=WFS&REQUEST=GetCapabilities
- https://tb12.cubewerx.com/a011/cubeserv?datastore=USGS&SERVICE=WFS&REQUEST=GetCapabilities
- https://tb12.cubewerx.com/a037/cubeserv?datastore=USGS&SERVICE=WFS&REQUEST=GetCapabilities
- https://tb12.cubewerx.com/a040/wmsCapabilities.xml
- https://tb12.cubewerx.com/a041/wmsCapabilities.xml
- https://tb12.cubewerx.com/a042/wmtsCapabilities.xml
- https://tb12.cubewerx.com/a045/wcsCapabilties_Flood_sims_AND_surge_H.xml
- https://tb12.cubewerx.com/a045/wcsCapabilities_Flooding_MODIS.xml
- https://tb12.cubewerx.com/a045/wcsCapabilities_Floodzone_sealevel_rise.xml
- https://tb12.cubewerx.com/a045/wcsCapabilities_GPM_Imerge_20160201.xml
- https://tb12.cubewerx.com/a045/wcsCapabilities_GPM_Imerge_20160209.xml
- https://tb12.cubewerx.com/a045/wcsCapabilities_LiDAR_2m_SFSU.xml

- https://tb12.cubewerx.com/a045/wcsCapabilities_NED_elevation.xml

The endpoints got challenged in different TIEs to evaluate the security annotation.

Cubewerx did also participate with their Web Browser based client to conduct TIEs. The result of the TIEs is captured in a later subsection.

11.2. Client Side Contributions from Compusult, Cubewerx, GMU, Luciad and Esri

11.2.1. Contribution from Compusult

For TB12, Compusult has not provided secured OGC Web Services. Their security related contribution for TB12 is conducting TIEs with their client product.

WMS + user certificate – Secure Dimensions

Client: Client (desktop) GIS Based Capabilities URL: https://tb12.ogc.secure-dimensions.com/capabilities/WMS-1.1.1.xml Result: The client was able to successfully load the annotated WMS public capabilities, and add the layer. DigitalGlobe:CitySphere to the map. The certificate for user “dbrown” was chosen when prompted by the browser.

WMTS + user certificate – Secure Dimensions

Client: Client (desktop) GIS Based Capabilities URL: https://tb12.ogc.secure-dimensions.com/capabilities/WMTS-1.0.0.xml Result The client was able to successfully load the annotated WMTS public capabilities, and add the layer. DigitalGlobe:CitySphereTileService to the map. The certificate for user “dbrown” was chosen when prompted by the browser.

WMTS – CubeWerx

Client: Client (desktop) GIS Based Capabilities URL: https://tb12.cubewerx.com/a042/wmtsCapabilities.xml Result: The client was confused by the presence of both SOAP and KVP encodings, and initiated a SOAP request, which failed. Investigating possible ways for the client to mitigate this situation.

The following screenshot illustrates successful TIE with the secured WMS from Digital Globe.

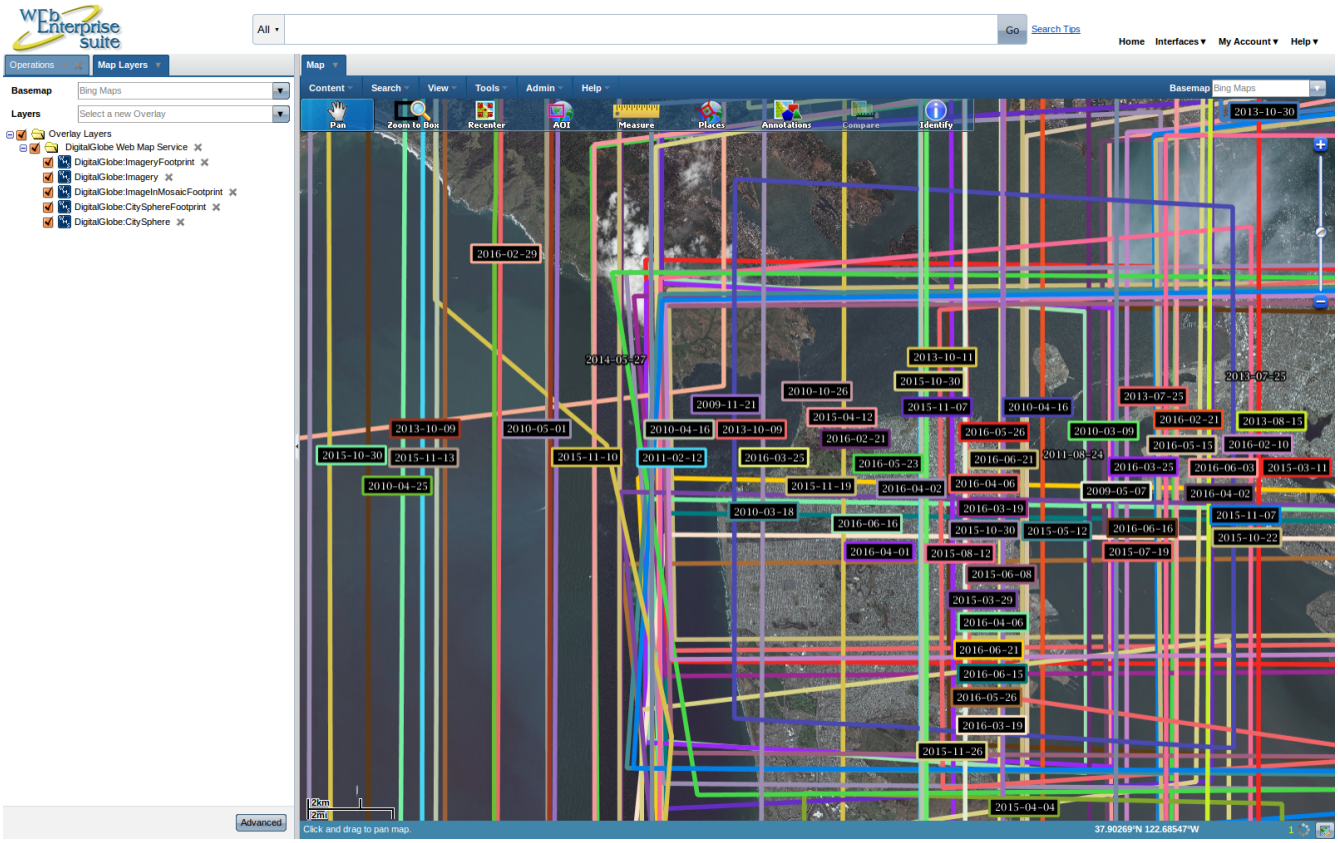

Figure 11. TIE result for Compusult WIS and DigitalGlobe WMS 1.1.1 via Secure Dimensions security proxy

Items for Further Discussion

- User credentials in server-side requests: Because the requests for map layers are invoked by the web client, the browser is able to prompt the user for a certificate and retrieve the map layers.

However, there is still the situation where server-side code may want to submit a request to the WMS/WMTS. Because this request is not made from the browser, the server-side code will need to inject user credentials into the request in order to retrieve data as that user. The alternative is for the server to use its own credentials (e.g. certificate in the server keystore) when making the request.

This may be a suitable workaround for a testbed event, but a production-level system will need to resolve the issue of reconciling user credentials with server-side requests.

- Negotiating between multiple bindings: In the case where multiple bindings (encodings) are supported by a service (e.g., SOAP and KVP), the client code must decide which one to use. If the client makes the wrong choice, it may lead to errors in the client. (e.g., if a client supports SOAP only for a subset of operations.) Clients will need to be coded to automatically negotiate an encoding, or pass the decision back to the user in the form of a prompt.

One solution is to allow the client to decide, but provide a means for the user to override the decision. In the case of adding a layer to a map, this could be presented as a drop-down list of available encodings.

11.2.2. Contribution from Esri

For TB12, Esri conducted TIEs for the secured WMTS https://tb12.ogc.secure-dimensions.com/capabilities/WMTS-1.0.0.xml using their desktop product ArcMap 10.4.1. The results from the successful TIE is captured in the following screenshot:

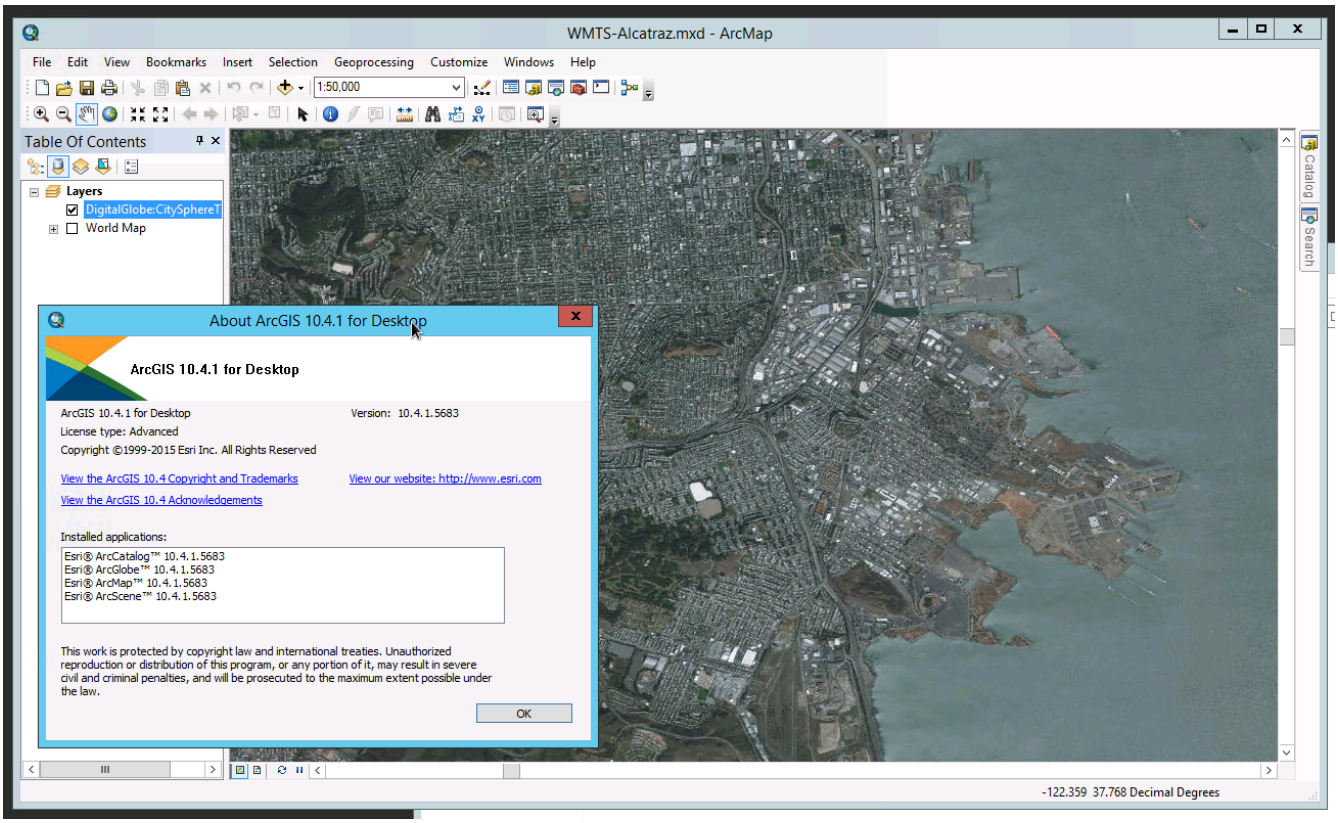

Figure 12. TIE result for ArcMap 10.4.1 and DigitalGlobe WMTS via Secure Dimensions security proxy

Table 2. WMTS

Tested Component	**ArcGIS Desktop 10.4.1**
Tested Against	WMTS Server
Tester Contact	Marten Hogeweg, Andreas Matheus
Date of TIE	26/10/2016
Request	GetCapabilities and GetMap requests using client-side certificate
Response	WMTS usable in ArcMap
Issues	None

11.2.3. Contribution from GMU

For TB12, GMU has conducted a TIE with their Web Browser based client and the WFS 2.0.0 with SOAP + WS-Security https://tb12.secure-dimensions.com/capabilities/WFS-SOAP-2.0.0.xml

The TIE was successful which is illustrated in the following screenshot:

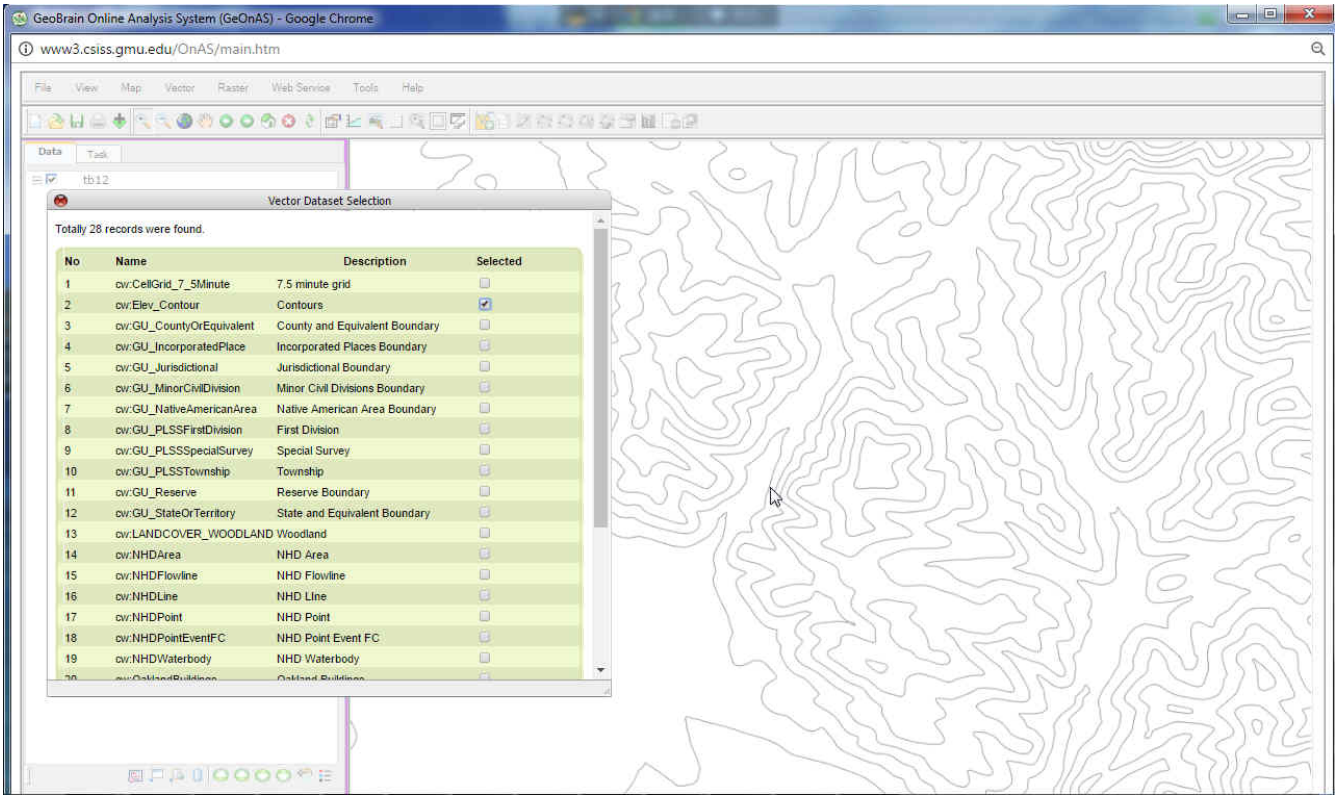

Figure 13. TIE result for Client Browser Based GMU and WFS 2.0.0 with SOAP + WS-Security

Table 3. SOAP + WS-Security

Tested Component	**Client Browser Based GMU**
Tested Against	Eval Security & SOAP, Common Security Extension WFS server
Tester Contact	Ziheng Sun
Date of TIE	25/10/2016
Request	GetCapabilities and GetFeature requests with WS-Security header
Response	WFS 2.0 Capability document and GML feature data with WS-Security headers
Issues	None

11.2.4. Contribution from Luciad

For TB12, Luciad conducted TIEs with their client product and different secured services. All TIEs were successful reading the annotated capabilities and displaying the service response. The following screenshot illustrated the TIE result with a WMS 1.1.1 from DigitalGlobe via a Secure Dimensions security proxy.

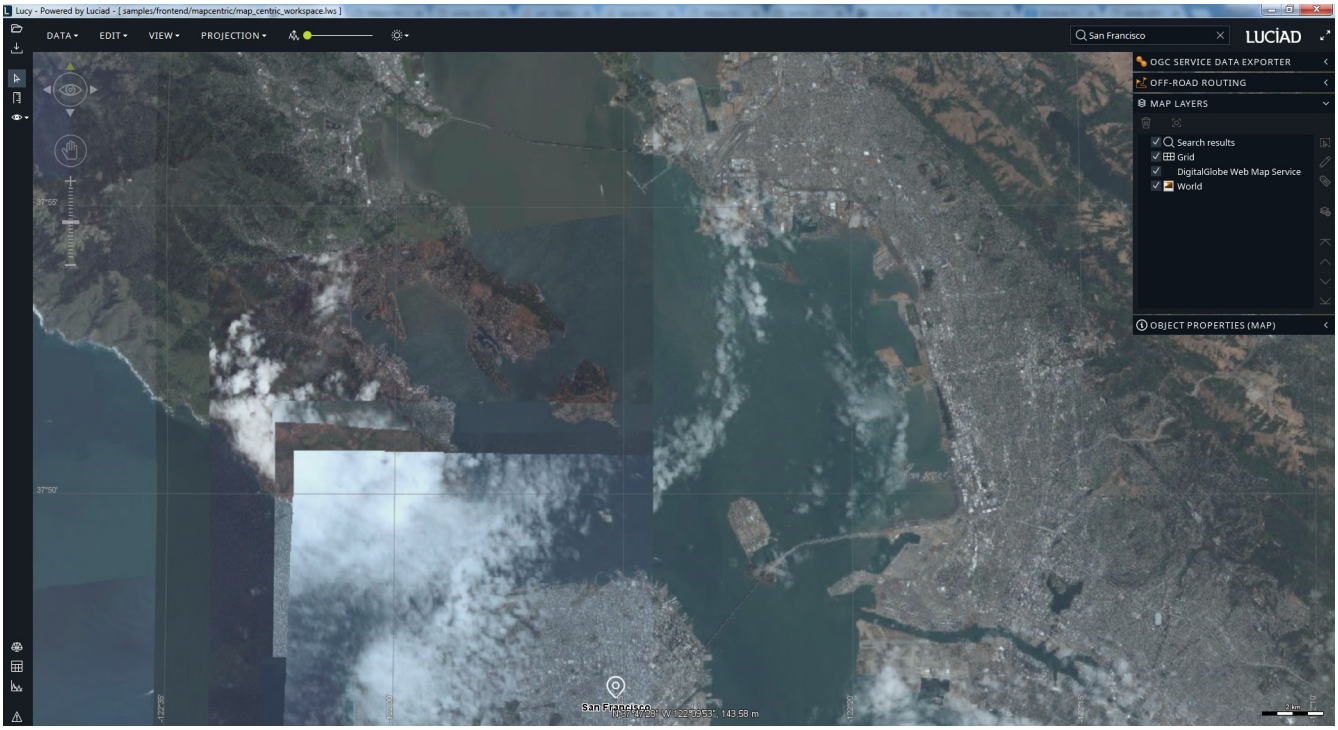

Figure 14. TIE result for Client Desktop GIS and DigitalGlobe WMS 1.1.1 via Secure Dimensions security proxy

Luciad performed a set of security TIEs using its Java-based Desktop Client, which has been developed to support the Field Operations thread. Security-wise, the client supports HTTP Basic Authentication and HTTPS SSL/TLS authentication using certificates. The client tested against WMS, WFS and WMTS services with security-annotated capabilities and using KVP/XML bindings. A test with a SOAP-based WFS was not performed, due to the lack of proper WS Security support in the client. The tests showed that the client was able to decode the capabilities without any issue, demonstrating the backwards compatibility of the security annotations. Querying and visualizing the data was also successful, apart from one interoperability issue related to the handling of a "not authorized" response encoded as an image (served by a WMS); the client was not able to display this image, due to a missing & appropriate mime type in the Content-Type header.

11.3. TIE Results

In order to determine backwards compatibility with existing clients for the annotated capabilities, TIEs where conducted. The objective of the TIEs is to determine whether an existing client - which is naturally not able to understand the security annotations in the capabilities document - does (i) crash when loading the annotated capabilities or (ii) connect to the protected service, ignoring the security annotation(s).

As a summary, it can be concluded that ALL tested clients did NOT crash when processing the annotated capabilities! This proves the proposed use of annotated capabilities as backwards-compatible, with at least the tested clients.

11.3.1. TIE results for WMS endpoints

The ESRI ArcMap 10.2.2 and 10.4.1 desktop applications; Luciad desktop GIS; Compusult WES were used to load the annotated capabilities for WMS 1.1.1 and 1.3.0:

- https://tb12.ogc.secure-dimensions.com/capabilities/WMS-1.1.1.xml
- https://tb12.ogc.secure-dimensions.com/capabilities/WMS-1.3.0.xml

The result of loading the annotated capabilities did not result in a crash of the client applications. After selecting a layer, the client application did load the map and displayed it with no error. In order to have the client execute the GetMap request for the first time, the user must select the client certificate to establish a TLS connection.

The result of this TIE is identical as if the client had loaded the capabilities via the protected service endpoint using the GetCapabilities operation.

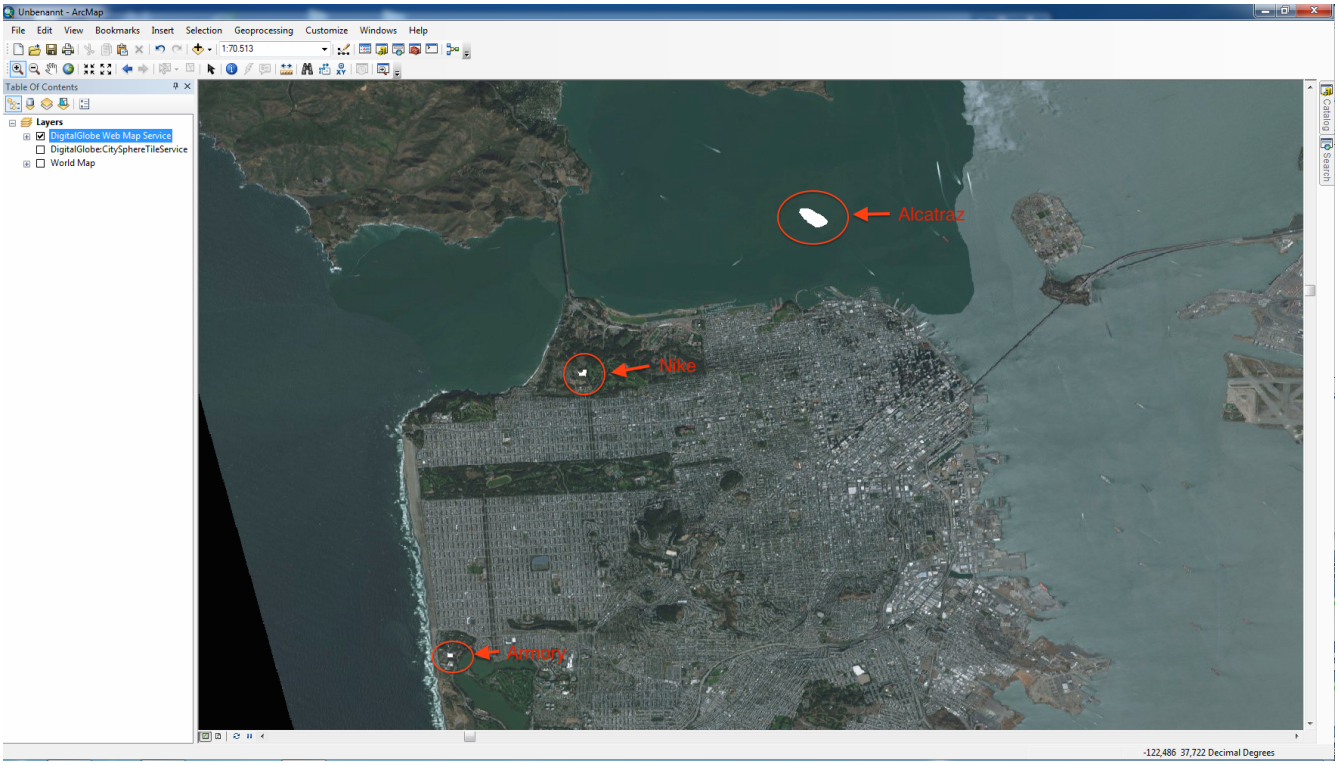

Figure 15. Map displayed after connecting to the WMS via unprotected and annotated capabilities

11.3.2. TIE results WMTS endpoint

The ESRI ArcMap 10.2.2 and 10.4.1 desktop applications; Luciad desktop GIS; Compusult WIS were used to load the annotated capabilities for WMTS 1.0.0:

- https://tb12.ogc.secure-dimensions.com/capabilities/WMTS-1.0.0.xml

The result of loading the annotated capabilities did not result in a crash any of the client applications.

However, after selecting a layer, the ArcMap 10.2.2 and 10.4.1 applications open multiple network connections in parallel to load the tiles via the GetTile request. At that point, the underlying network library causes popup boxes to appear asking the user to select a client certificate to establish the TLS connection. After the second popup box is processed by the user - selecting the certificate - the application crashes. This problem was reported to ESRI via the proper channels. The crash is not related to the use of annotated capabilities, as the same crash occurs at random (3 of 5 attempts) if a project file /.mxd file for the WMTS is used to connect to the service. The cause for the crash is the parallel connections for loading tiles. WMS works, because only one single GetMap

request is issued and therefore only one popup box occurs.

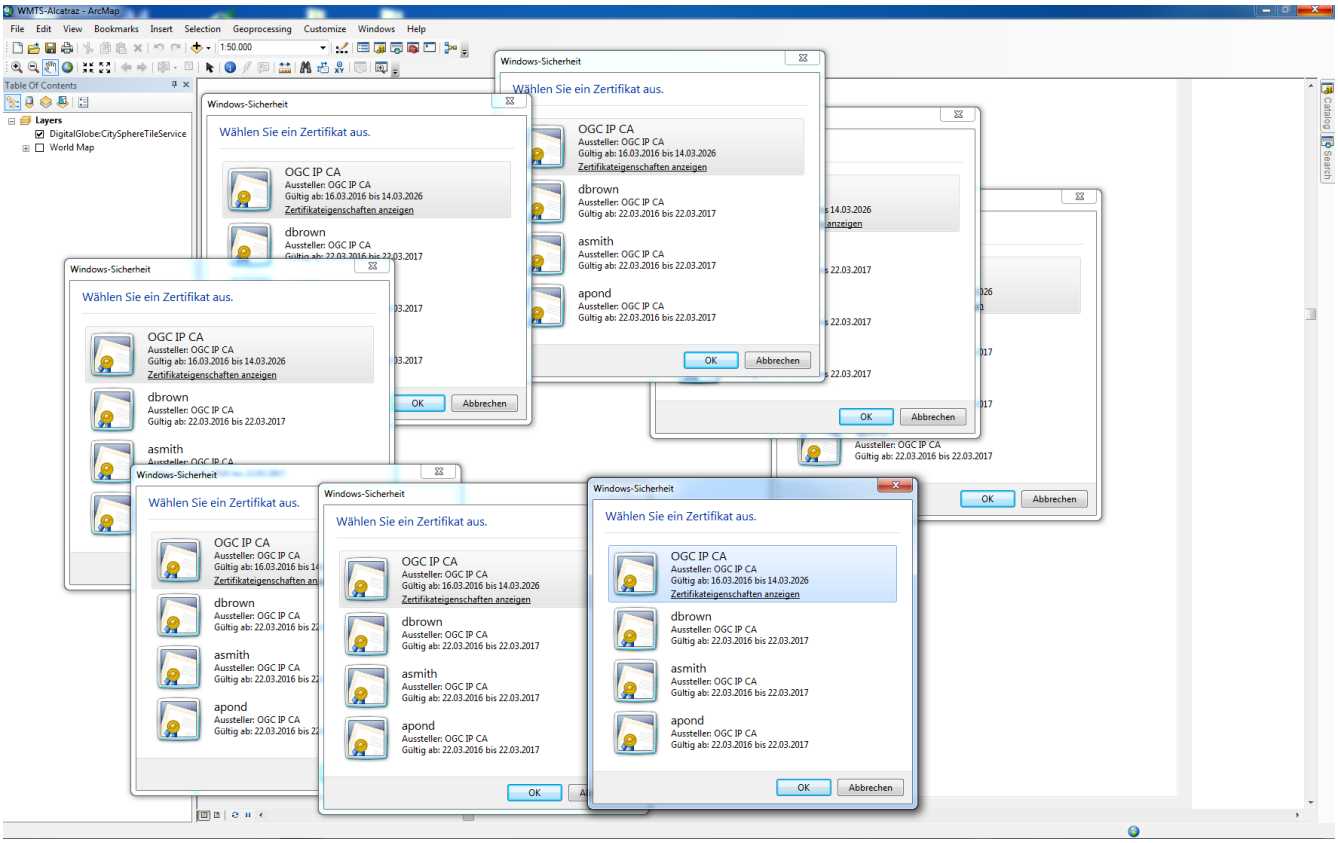

Figure 16. Popup boxes displayed after connecting ArcMap 10.2.2 to the WMTS via unsecured capabilities or .mxd project file

When using the secured endpoint of the service (https://tb12.ogc.secure-dimensions.com/service/wmts) ArcMap connects to the service via the GetCapabilties request which causes the popup of the certificate selection box. Once that single request has resulted in a TLS connection, the parallel requests for GetTile do not require a certificate selection any more. So, the application works as expected.

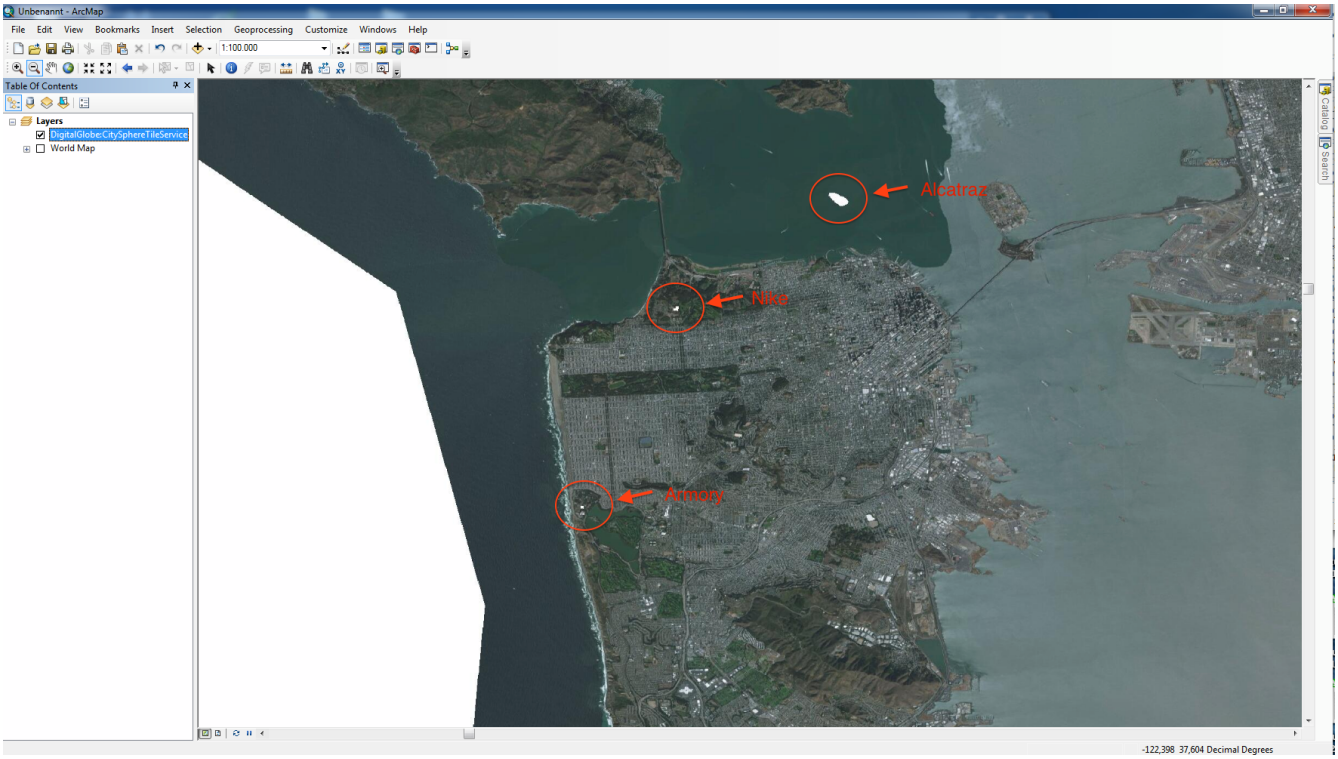

Figure 17. Map displayed after connecting to the WMTS via secured GetCapabilties request

11.3.3. TIE results WFS endpoint

The ESRI ArcCatalog 10.2.2 desktop application; Luciad desktop GIS; Compusult WES were used to load the annotated capabilities for WFS 1.1.0:

- https://tb12.ogc.secure-dimensions.com/capabilities/WFS-1.1.0.xml

It was possible to create an Interoperability Extension for the WFS using ArcCatalog. The annotated capabilities document was loaded with no crash. It was also possible to select a feature type and save the configuration in ArcCatalog.

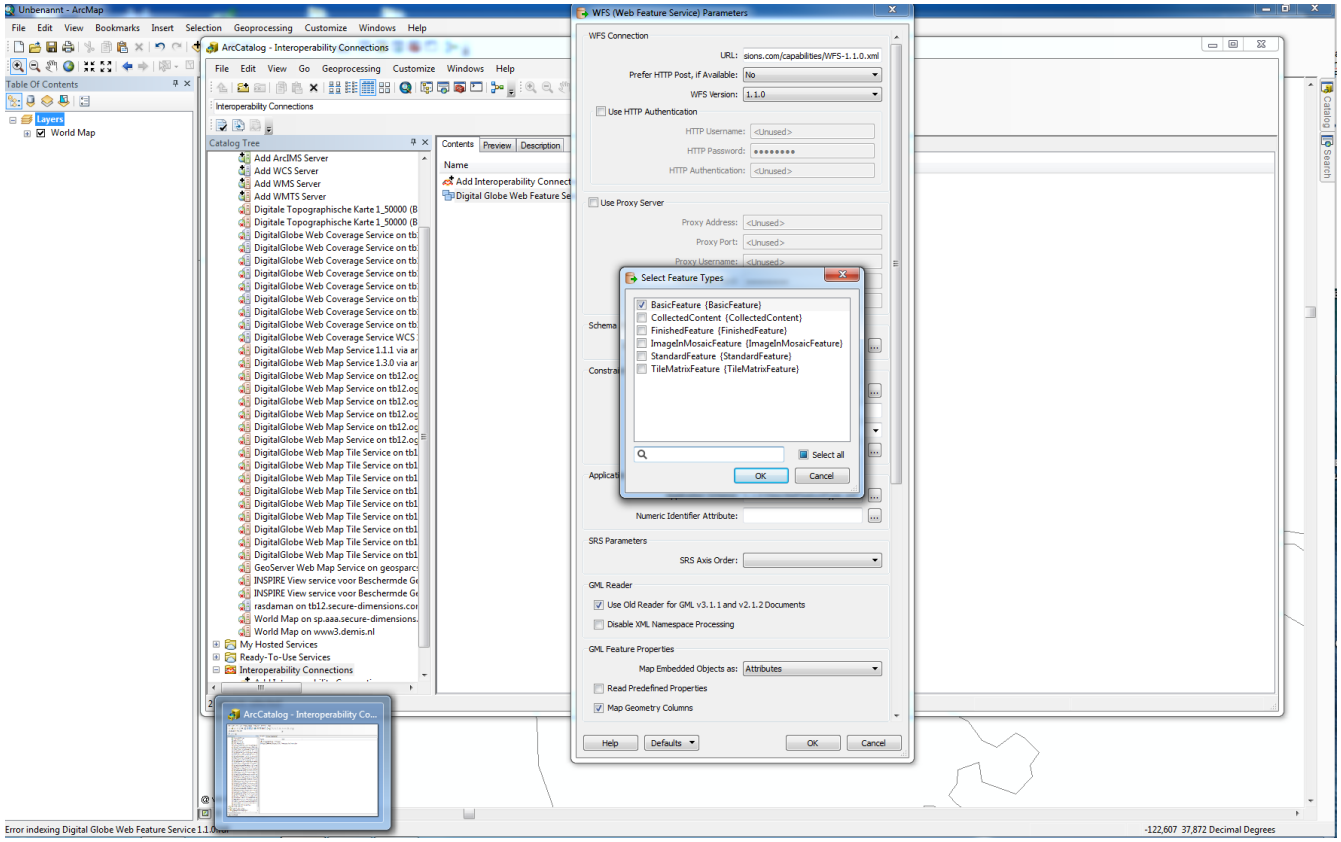

Figure 18. Creation of the WFS instance from loading the annotated capabilities document

The loading of features from this interoperability connection in ArcMap 10.2.2 failed.

The loading of the annotated capabilities and displaying of the result did work for the Luciad desktop GIS and the Compusult WIS.

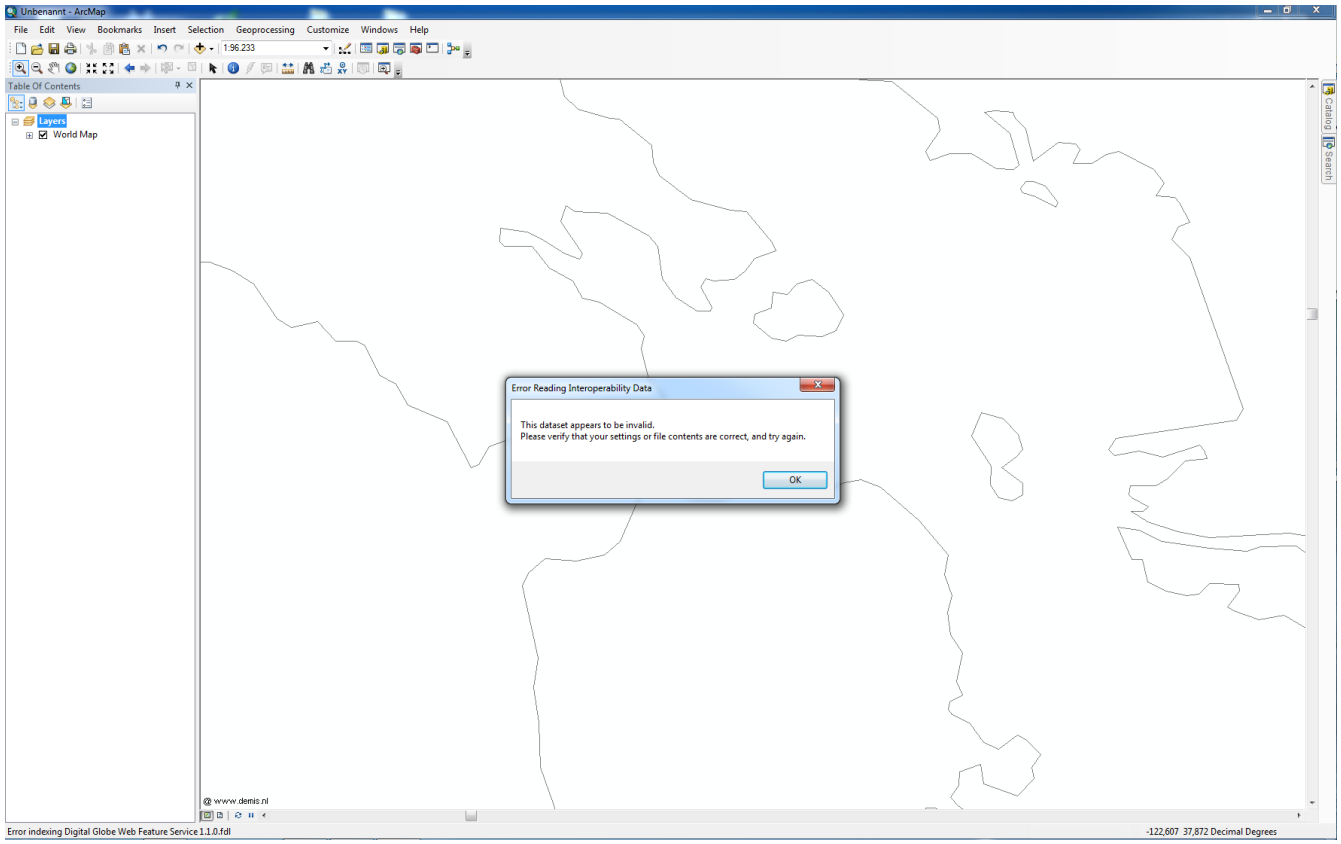

Figure 19. Loading of features from the protected WFS failed in ArcMap 10.2.2

The result of the TIE is that the use of the annotated capabilities did not crash the client application. But the lack of support for the client side TLS caused the ArcMap client not to load any features.

11.3.4. TIE results WCS endpoint

The ESRI ArcMap 10.2.2 desktop application was used to load the annotated capabilities for WCS 1.1.1:

- https://tb12.ogc.secure-dimensions.com/capabilities/WCS-1.1.1.xml

It was possible to load the annotated capabilities files without crashing the client. But the client did not issue a DescribeCoverage request and the selection box remained empty.

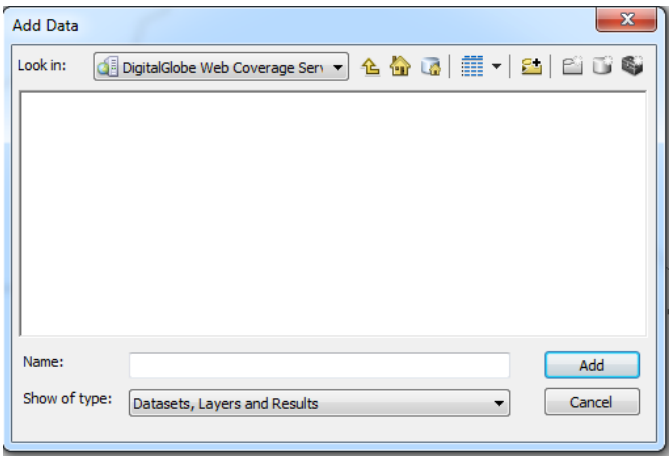

Figure 20. Loading of Coverages fails

The result of the TIE is that the use of the annotated capabilities did not crash the client application. But for an error in the capabilities document, it was not possible to select a coverage.

11.3.5. TIE results Summary for secured W*S endpoints

The following TIE matrix documents the result from the testing of secured W*S and annotated capabilities

*Table 4. W*Services (Evaluate Security and SOAP, Common Security Extension WFS server)*

Annotated public capabilities	Client Application	Crash?	Function?	Test performed by
https://tb12.ogc.secure-dimensions.com/capabilities/WMS-1.1.1.xml	ArcMap 10.2	No	Yes	Andreas Matheus
https://tb12.ogc.secure-dimensions.com/capabilities/WMS-1.3.0.xml	ArcMap 10.2	No	Yes	Andreas Matheus

https://tb12.ogc.secure-dimensions.com/capabilities/WMTS-1.0.0.xml	ArcMap 10.2	No	No*1	Andreas Matheus
https://tb12.ogc.secure-dimensions.com/capabilities/WFS-1.1.0.xml	ArcCatalog 10.2 / ArcMap 10.2	No / No	Yes	Andreas Matheus
https://tb12.ogc.secure-dimensions.com/capabilities/WCS-1.1.1.xml	ArcMap 10.2	No	No *2	Andreas Matheus
https://tb12.secure-dimensions.com/capabilities/WCPS-2.0.1.xml	ArcMap 10.2	No	No *3	Andreas Matheus
https://tb12.ogc.secure-dimensions.com/capabilities/WMTS-1.0.0.xml	ArcMapcMap 10.4.1	No	Yes	Marten Hogeweg
https://tb12.ogc.secure-dimensions.com/capabilities/WMS-1.1.1.xml	ArcMap 10.2.2 / QGIS 2.16	No	No *5	Andreas Matheus
https://tb12.ogc.secure-dimensions.com/capabilities/WMS-1.3.0.xml	ArcMap 10.2.2 / QGIS 2.16	No	No *5	Andreas Matheus
https://tb12.ogc.secure-dimensions.com/capabilities/WMTS-1.0.0.xml	ArcMap 10.2.2 / QGIS 2.16	No	No *5	Andreas Matheus
https://tb12.ogc.secure-dimensions.com/capabilities/WFS-1.1.0.xml	ArcMap 10.2.2 / QGIS 2.16	N/A *6	No *5	Andreas Matheus

https://tb12.ogc.secure-dimensions.com/capabilities/WCS-1.1.1.xml	ArcMap 10.2.2 / QGIS 2.16	No	No *5	Andreas Matheus
https://tb12.secure-dimensions.com/capabilities/WCPS-2.0.1.xml	ArcMap 10.2.2 / QGIS 2.16	N/A *6	No *5	Andreas Matheus
https://tb12.ogc.secure-dimensions.com/capabilities/WMS-1.1.1.xml	Client (desktop) GIS Based	No	Yes	Craig Coffin
https://tb12.ogc.secure-dimensions.com/capabilities/WMTS-1.0.0.xml	Client (desktop) GIS Based	No	Yes	Craig Coffin
http://tb12.cubewerx.com/a042/wmtsCapabilities.xml	Client (desktop) GIS Based	No	No *4	Craig Coffin
https://tb12.cubewerx.com/a007/cubeserv?datastore=USGS&SERVICE=WFS&REQUEST=GetCapabilities	Google Chrome Version 54.0.2840.71 (64-bit), QGIS 2.0.1-Dufour	No	Yes	Panagiotis Vretanos
https://tb12.cubewerx.com/a011/cubeserv?datastore=USGS&SERVICE=WFS&REQUEST=GetCapabilities	Google Chrome Version 54.0.2840.71 (64-bit), QGIS 2.0.1-Dufour	No	Yes	Panagiotis Vretanos
https://tb12.cubewerx.com/a037/cubeserv?datastore=USGS&SERVICE=WFS&REQUEST=GetCapabilities	Google Chrome Version 54.0.2840.71 (64-bit), QGIS 2.0.1-Dufour	No	Yes	Panagiotis Vretanos
https://tb12.cubewerx.com/a040/wmsCapabilities.xml	Google Chrome Version 54.0.2840.71 (64-bit), QGIS 2.0.1-Dufour	No	Yes	Panagiotis Vretanos

https://tb12.cubewerx.com/a041/wmsCapabilities.xml	Google Chrome Version 54.0.2840.71 (64-bit), QGIS 2.0.1-Dufour	No	Yes	Panagiotis Vretanos
https://tb12.cubewerx.com/a042/wmtsCapabilities.xml	Google Chrome Version 54.0.2840.71 (64-bit), QGIS 2.0.1-Dufour	No	Yes	Panagiotis Vretanos
https://tb12.cubewerx.com/a045/wcsCapabilties_Flood_sims_AND_surge_H.xml	Google Chrome Version 54.0.2840.71 (64-bit), QGIS 2.0.1-Dufour	No	Yes	Panagiotis Vretanos
https://tb12.cubewerx.com/a045/wcsCapabilities_Flooding_MODIS.xml	Google Chrome Version 54.0.2840.71 (64-bit), QGIS 2.0.1-Dufour	No	Yes	Panagiotis Vretanos
https://tb12.cubewerx.com/a045/wcsCapabilities_Floodzone_sealevel_rise.xml	Google Chrome Version 54.0.2840.71 (64-bit), QGIS 2.0.1-Dufour	No	Yes	Panagiotis Vretanos
https://tb12.cubewerx.com/a045/wcsCapabilities_GPM_Imerge_20160201.xml	Google Chrome Version 54.0.2840.71 (64-bit), QGIS 2.0.1-Dufour	No	Yes	Panagiotis Vretanos
https://tb12.cubewerx.com/a045/wcsCapabilities_GPM_Imerge_20160209.xml	Google Chrome Version 54.0.2840.71 (64-bit), QGIS 2.0.1-Dufour	No	Yes	Panagiotis Vretanos
https://tb12.cubewerx.com/a045/wcsCapabilities_LiDAR_2m_SFSU.xml	Google Chrome Version 54.0.2840.71 (64-bit), QGIS 2.0.1-Dufour	No	Yes	Panagiotis Vretanos
https://tb12.cubewerx.com/a045/wcsCapabilities_NED_elevation.xml	Google Chrome Version 54.0.2840.71 (64-bit), QGIS 2.0.1-Dufour	No	Yes	Panagiotis Vretanos

https://tb12.ogc.secure-dimensions.com/capabilities/WMS-1.1.1.xml	Client (desktop) GIS Based	No	Yes	Robin Houtmeyers
https://tb12.ogc.secure-dimensions.com/capabilities/WMS-1.3.0.xml	Client (desktop) GIS Based	No	Yes	Robin Houtmeyers
https://tb12.ogc.secure-dimensions.com/capabilities/WMTS-1.0.0.xm	Client (desktop) GIS Based	No	Yes	Robin Houtmeyers
https://tb12.ogc.secure-dimensions.com/capabilities/WFS-1.1.0.xml	Client (desktop) GIS Based	No	Yes	Robin Houtmeyers

Notes

- Authentication codes in the annotated capabilities document are from the authentication code list @ http://tb12.opengis.net/security/authCodeList
- 1: ArcMap crashes - to the opinion of the tester - due to a bug in ArcMap. The bug was reported to ESRI thru the proper channel: Crash Reporter.
- 2: The selection of Coverages is empty.
- 3: The selection of Coverages is empty.
- 4: The client was confused by the presence of both SOAP and KVP encodings, and initiated a SOAP request, which failed. Investigating possible ways for the client to mitigate this situation.
- 5: QGIS 2.16 does not support TLS with client side certificate
- 6: QGIS 2.16 does not support WFS 1.1.0 and WCS 2.0

11.3.6. TIE results Summary for SOAP + WS-Security endpoint

The following TIE matrix documents the result from testing the WFS 2.0.0 with SOAP + WS-Security

Table 5. SOAP + WS-Security

Annotated public capabilities	Client Application	Crash?	Function?	Test performed by
https://tb12.secure-dimensions.com/capabilities/WFS-SOAP-2.0.0.xml	No client available	N/A	N/A	Andreas Matheus
https://tb12.secure-dimensions.com/capabilities/WFS-SOAP-2.0.0.xml	Client (desktop) GIS Based	N	N (because of the client's inability to correctly use WS Security)	Robin Houtmeyers
https://tb12.secure-dimensions.com/capabilities/WFS-SOAP-2.0.0.xml	Client Browser Based	No	Yes	Ziheng Sun

Appendix A: TB12 Protected Services and Annotated Capabilities Documents

This section lists all protected services from TB12 that have public capabilities with security annotations and are accessible online.

A.1. Secure Dimensions

A.1.1. WFS 1.1.0

- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.ogc.secure-dimensions.com/capabilities/WFS-1.1.0.xml

A.1.2. WCS 1.1.1

- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.ogc.secure-dimensions.com/capabilities/WCS-1.1.1.xml

A.1.3. WCPS 2.0.1

- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.secure-dimensions.com/capabilities/WCPS-2.0.1.xml

A.1.4. WMS 1.1.1

- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.ogc.secure-dimensions.com/capabilities/WMS-1.1.1.xml

A.1.5. WMS 1.3.0

- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.ogc.secure-dimensions.com/capabilities/WMS-1.3.0.xml

A.1.6. WMTS 1.0.0

- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.ogc.secure-dimensions.com/capabilities/WMTS-1.0.0.xml

A.1.7. WFS 2.0 + SOAP WS-Security

- Service endpoint: https://tb12.secure-dimensions.com/soap/services/ows_proxy
- Public Capabilities: https://tb12.secure-dimensions.com/capabilities/WFS-SOAP-2.0.0.xml

A.2. Cubewerx

A.2.1. WFS 2.0.0

- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.cubewerx.com/a007/cubeserv?datastore=USGS&SERVICE=WFS&REQUEST=GetCapabilities
- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.cubewerx.com/a011/cubeserv?datastore=USGS&SERVICE=WFS&REQUEST=GetCapabilities
- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.cubewerx.com/a037/cubeserv?datastore=USGS&SERVICE=WFS&REQUEST=GetCapabilities

A.2.2. WMS 1.3.0

- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.cubewerx.com/a040/wmsCapabilities.xml
- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.cubewerx.com/a041/wmsCapabilities.xml
- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.cubewerx.com/a042/wmsCapabilities.xml

A.2.3. WCS 2.0.1

- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.cubewerx.com/a045/wcsCapabilties_Flood_sims_AND_surge_H.xml
- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.cubewerx.com/a045/wcsCapabilities_GPM_Imerge_20160201.xml
- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.cubewerx.com/a045/wcsCapabilities_GPM_Imerge_20160209.xml
- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.cubewerx.com/a045/wcsCapabilities_LiDAR_2m_SFSU.xml
- Service endpoint: GetCapabilities
- Public Capabilities: https://tb12.cubewerx.com/a045/wcsCapabilities_NED_elevation.xml

A.2.4. WMTS 1.0.0

- Service endpoint: GetCapabilities
- Public Capabilities: http://tb12.cubewerx.com/a042/wmtsCapabilities.xml

A.3. GMU

A.3.1. WCS 2.0.1 with SOAP + WS-Security

- Service endpoint: GetCapabilities
- Public Capabilities: ?

Appendix B: Revision History

Table 6. Revision History

Date	Release	Editor	Primary clauses modified	Descriptions
2016-04-15	A. Matheus	.1	all	initial version
2016-09-30	A. Matheus, Chuck Heazel	.2	all	draft version
2016-10-07	A. Matheus	.3	all	version for SWG to review
2016-10-17	A. Matheus	.4	all	incorporating feedback from SWG review
2016-10-31	A. Matheus	.5	all	incorporating feedback from TB12 review
2016-11-09	A. Matheus	.6	CR	incorporating feedback from OWS Common - Security SWG
2016-11-15	A. Matheus	.7	CR	final version for public release 16-048r1
2017-03-07	S. Simmons	.8	all	final edits for publication